Infrastructure: Secure Core Services *Lecture Notes, Review Questions, Problems and Lab Tutorials*

Erik Hjelmås

April 17, 2026

About this Compendium

This compendium serves as the main learning resource for the second-semester course Infrastructure: Secure Core Services. While it is not a fully polished textbook, it contains the essential material you need for the course. Some sections may still be incomplete or contain typos, so your feedback is highly appreciated. If you notice anything that could be improved, please email your suggestions to `erik.hjelmas@ntnu.no`.

The goal of this course is to help you develop knowledge, practical skills, and general competence that will be valuable both in real-world work and in further studies. For professional roles, we focus on preparing you for positions such as *Junior Security Analyst* and *Junior Infrastructure Engineer*. For academic progression, this course provides a strong foundation for advanced topics like ethical hacking, risk analysis, and infrastructure as code.

Some practical information about this compendium:

- Start with Chapters 1 and 2, as they provide the foundation for the entire course. After completing these, most of the remaining chapters can be studied independently. However, some chapters have minor dependencies, so we recommend following the sequence outlined below for the best learning experience:
  - Chapter 3.
  - Chapter 4.
  - Chapter 5, 6, 7, 8.
  - Chapter 9.
  - Chapter 10, 11.
  - Chapter 12.
- This compendium is automatically generated from lecture slides and notes, which is why figure references appear at the beginning of many paragraphs. *All the slides used in lectures are included here as figures*, so you have a complete visual reference alongside the text.
- Each chapter in this compendium includes a main text section, followed by review questions and problems, and in some cases, one or more lab tutorials. To get the most out of the material, we recommend the following approach for each chapter:
  1. Read the chapter text carefully.
  2. Complete any lab tutorials provided.
  3. Work through the review questions and problems to reinforce your understanding.

- PowerShell is a central theme throughout this compendium. Our goal is for you to develop strong and lasting skills in PowerShell, enabling you to automate tasks, manage systems efficiently, and apply scripting to security and infrastructure challenges. By the end of the course, you should feel confident using PowerShell as a practical tool in real-world scenarios.
- Exam questions will be based on the review questions, problems, and lab tutorials included in this compendium. Completing these exercises is essential for both learning and exam preparation. Some review questions are marked as *Key Problems* to highlight their importance – these are especially relevant for the exam and should be prioritized.
- We aim to use consistent syntax throughout the compendium. PowerShell code examples that include output start with `PS>`. Examples without `PS>` are intended for you to copy, paste, and run yourself. Occasionally, you may need to merge lines for the code to work – this will always be indicated in a preceding comment (`#`). For example:

```
# the following shows a PowerShell command and its output:
PS> Get-Date
Wednesday, December 17, 2025 11:30:13 AM

# the following is a PowerShell command you can copy, paste,
# and run yourself:
Get-Date

# the following should be on one line:
curl -O https://download.microsoft.com/download/8/5/C/
  85C25433-A1B0-4FFA-9429-7E023E7DA8D8/
  Windows%2010%20version%2022H2%20Security%20Baseline.zip
```

I would like to thank everyone who contributed valuable input to this compendium:

- My great colleagues Tor Ivar Melling, Ernst Gunnar Gran, Lars Erik Pedersen and Eigil Obrestad.
- My teaching assistants Jan van Ngo and Peder Andreas Stuen.
- My students Daniel Hinderaker, Sara Lehne Engesvik, Eline Lædre and Magnus Næs.

Your feedback has been instrumental in shaping this material.

This compendium, including its introduction, structure, language improvements, and many of the exercise suggestions, was developed with assistance from Microsoft Copilot – an AI tool used

to refine text, propose structural enhancements, and improve clarity and student engagement throughout the material.

Erik Hjelmås, April 17, 2026

Contents

Glossary

Infrastructure In our course context, a collection of servers, software, and networking components that provide core computing services, similar to Infrastructure as a Service (IaaS).

1

NSM, Cloud Computing and PowerShell

1.1 NSM "Grunnprinsipper for IKT-sikkerhet 2.1"

NSM Grunnprinsipper for IKT-sikkerhet 2.1 in figure 1.1. The main goal of this course is to help you master two key areas: (1) Windows Infrastructure – both theory and practical implementation, and (2) the fundamentals of secure system management. While most Internet-facing servers run Linux, the majority of critical internal servers in organizations typically run Windows, often with Active Directory at the core. Modern IT Infrastructure are increasingly deployed in private or public clouds, so we will also introduce cloud computing concepts early in the course. Our approach is guided by NSM's (Nasjonal Sikkerhetsmyndighet) "Grunnprinsipper for IKT-sikkerhet 2.1" [1], the framework widely used by Norwegian organizations to structure their security practices.

Mapping NSM to Chapters in this Compendium in figure 1.2. Once we've covered the basics, we'll dive into what we call "getting the basics right." In practice, this means learning how to handle core security tasks in a cloud-based Windows environment. These include things like:

- Backup and restore (NSM 2.7 and 2.9, Chapter 3)
- Identity management and access control (NSM 2.6, Chapter 6 and 7)
- Configuration management (NSM 2.3, Chapter 7)
- Software package management (NSM 1.2, 2.1 and 2.3, Chapter 8)

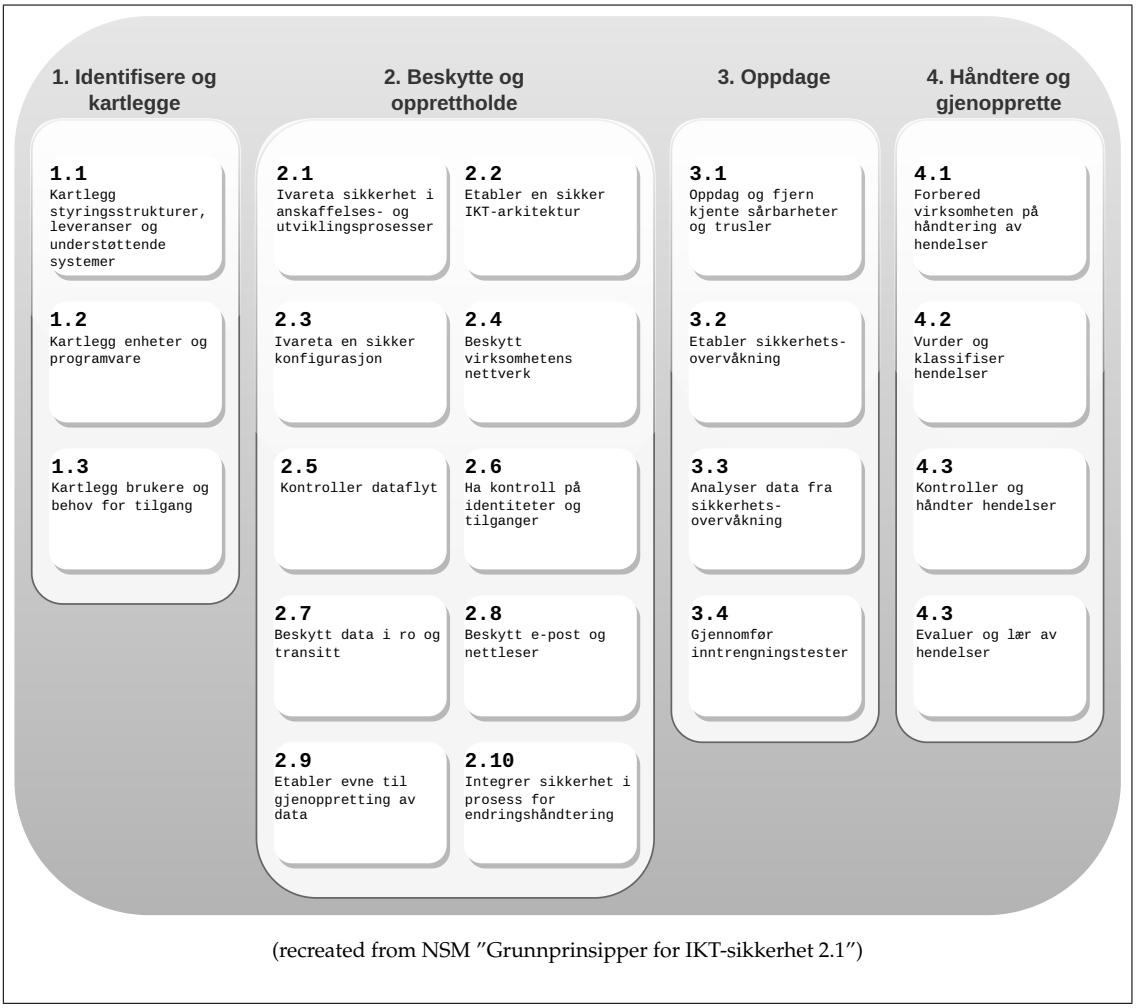


Figure 1.1: NSM Grunnprinsipper for IKT-sikkerhet 2.1 .

- Logging and monitoring (NSM 3.2, Chapter 9)
- Fast (automatic) redeploy/installation/provisioning (NSM 4.3, Chapter 12)

1.2 IT Landscape of Today

DevSecOps: IT today in figure 1.3. Before approximately 2010, IT work was typically divided into two distinct areas: development and operations. Developers – such as programmers, software engineers, and project managers – focused solely on creating software. A few times per year, they would release new versions and "hand them over" to the operations team. IT operations, consisting of system and network administrators, were responsible for deploying and maintaining these systems in production, often using manual processes and specialized tools for installation, upgrades, and monitoring.

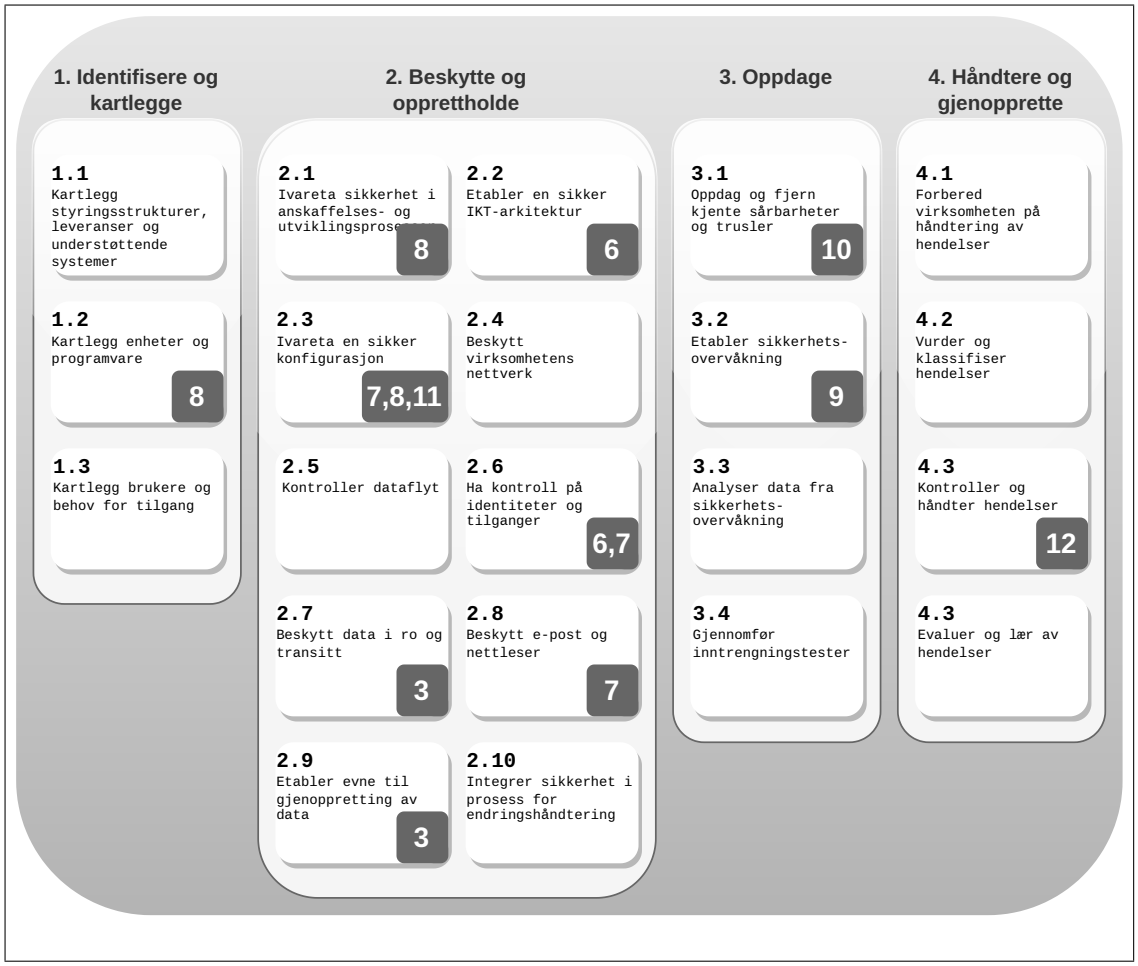


Figure 1.2: Mapping NSM to Chapters in this Compendium.

Meanwhile, a separate security team attempted to enforce security policies, which were frequently perceived as an obstacle by the operations team.

Today, the approach is very different. The adoption of agile development methods, combined with technological advances such as virtualization and containers, has enabled much more frequent software releases and eliminated the traditional barrier between developers and IT operations. At the same time, the growing number of security incidents and attacks has led to security becoming an integrated part of both development and operations. This modern way of working is known as *DevOps*, or *DevSecOps* when security is explicitly emphasized. Occasionally, the term *DevOPS* is used – with OPS in uppercase – to highlight a stronger focus on operations and security rather than development.

At the core of this collaboration is the version control system Git, which we will explore in Chapter 4. Git has become an essential part of the modern IT landscape because nearly all technical tasks – whether in development, operations, or security –

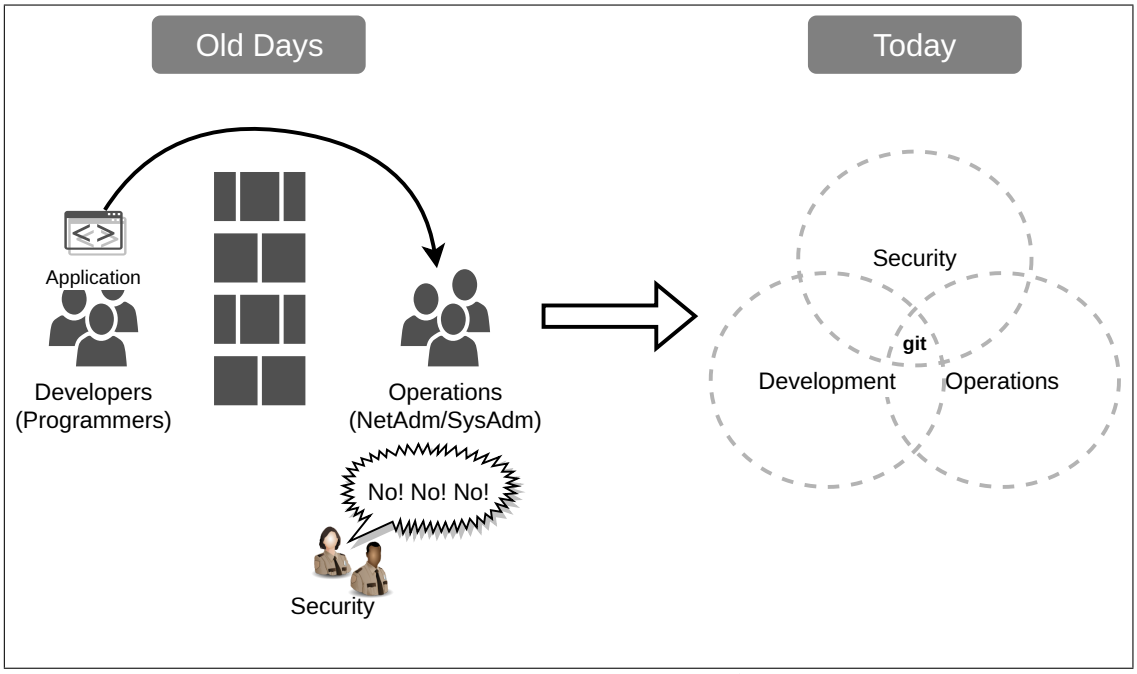


Figure 1.3: DevSecOps: IT today.

can now be expressed as code. These tasks should therefore be stored in a version-controlled repository such as Git. Patrick Debois provides an excellent overview of this concept in his blog "In-depth research and trends analyzed from 50+ different concepts as code" [17], where he discusses examples like "configuration as code", "threat modeling as code", "UI as code", and "contracts as code". His work illustrates how virtually everything can be treated as code. In our context, this reinforces *the importance of using the command line for most tasks*, since commands themselves are code and can easily be stored in a version control system like Git.

An excellent, easy-to-read book that explores this transformation in the IT landscape is "The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win" [27]. Unlike a traditional textbook, it presents these concepts through a narrative format, making complex ideas accessible and engaging.

1.3 Cloud Computing

IaaS, PaaS, SaaS in figure 1.4. Cloud computing delivers computing services over the Internet without requiring users to know the exact physical location of the servers providing those services – hence the term "the cloud." The first widely adopted cloud services were introduced by Amazon Web Services in 2006 (IaaS), followed by Google App Engine in 2008 (PaaS) and Microsoft Azure in 2010. Today, these services are generally categorized into three main models:

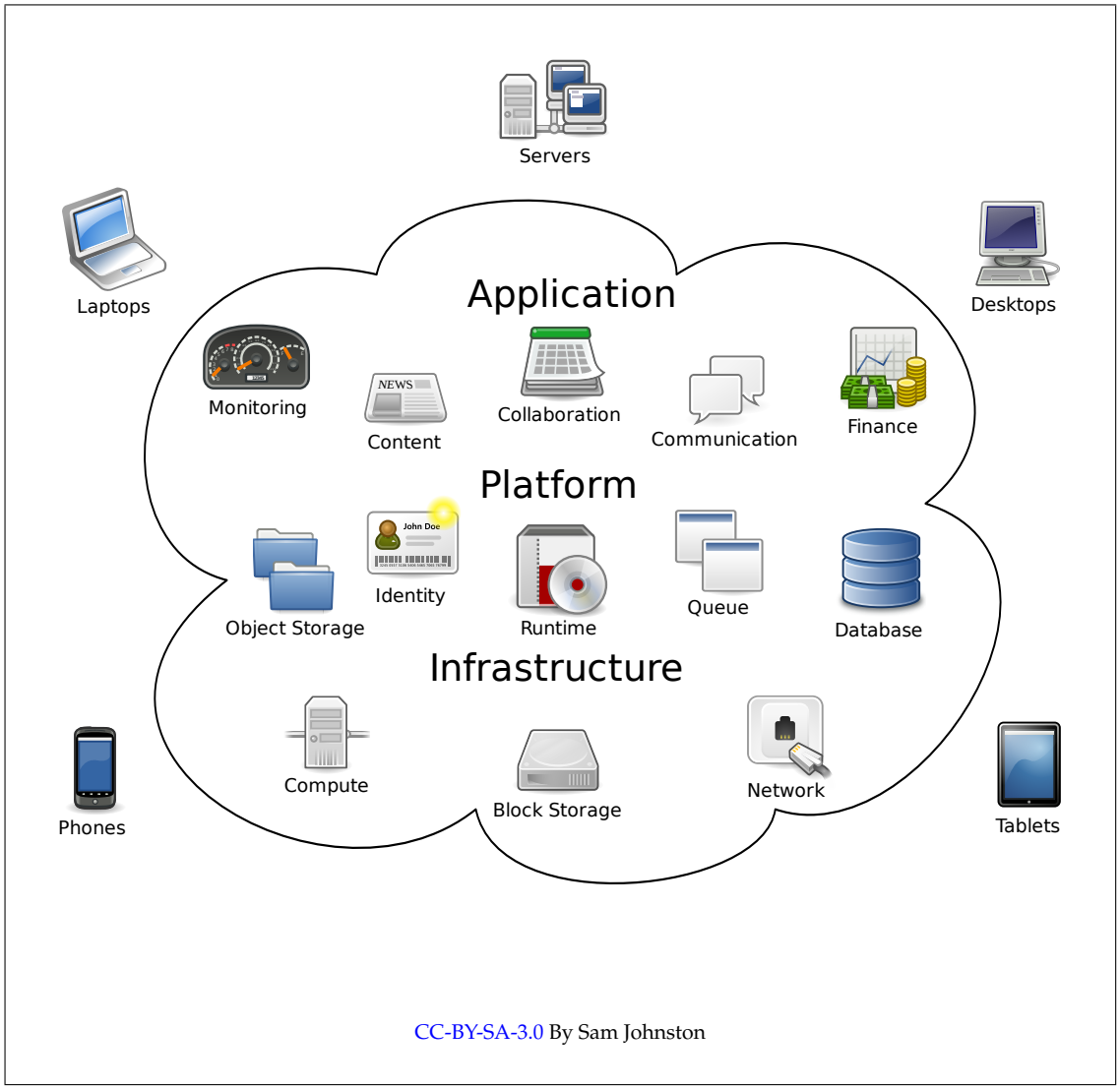


Figure 1.4: IaaS, PaaS, SaaS.

Dynamic you can create and destroy/delete infrastructures as needed
Self-service you manage everything yourself in software (instead of buying hardware)
Pay-as-you-go you only pay for the resources you need and use

Figure 1.5: Cloud Computing Characteristics.

IaaS Infrastructure as a Service: compute (virtual machines), network and storage

PaaS Platform as a Service: run-time environments (e.g. Kubernetes), databases, web servers, object storage (files access over http, not "virtual hard disk" like in IaaS)

SaaS Software as a Service: e.g. email (Gmail), monitoring (Grafana cloud), LMS (Blackboard Learn SaaS)

Occasionally, you may encounter the term *serverless computing*, which describes a model where customers neither manage nor interact directly with virtual machines. Instead, they consume services that are typically categorized as PaaS. When an organization fully embraces modern cloud capabilities – along with contemporary development and operational practices such as Agile, DevOps, and DevSecOps – it is considered *cloud native* [2]. In contrast, simply replicating an on-premise data center in the cloud without leveraging these advantages is often described as the opposite of being cloud native.

1.3.1 Characteristics

Cloud Computing Characteristics in figure 1.5. Cloud computing is built on virtualization and shared resources, enabling multiple users to access computing power, storage, and networking from common physical infrastructure. Its defining characteristics – *dynamic* scalability, *self-service* provisioning, and *pay-as-you-go* pricing – have driven a fundamental shift in how IT services are delivered. These features allow organizations to rapidly create or remove resources as needed, manage infrastructure entirely through software, and pay only for what they consume, making cloud computing both flexible and cost-efficient.

1.3.2 Security and Privacy

When an organization maintains its own server room with power supplies, racks, cables, network devices, and servers, this setup is referred to as *on-premise infrastructure*. Some companies operate entirely on-premise, others rely fully on public cloud providers such as Amazon Web Services or Microsoft Azure, and many adopt a hybrid approach combining both. From a security and privacy perspective, moving to the cloud does not introduce fundamentally new concerns; most issues remain similar to

- Privacy: storing data on someone else's computers
- All issues related to outsourcing (possibly loosing control)
- Maintaining an accurate inventory of assets can be challenging in highly dynamic environments
- Fast DevOps-style replacement of services can be beneficial for patching
- Otherwise mostly same issues as on-premise (patch/update, backups, access control, logs, monitoring)

Figure 1.6: Cloud Computing: Security and Privacy.

those faced on-premise. However, special attention must be paid to data storage locations due to legal and regulatory requirements. For example, are you permitted to store data in the country where the cloud provider's servers are located? You will explore these topics in more detail in later courses, but for now, make a note to consult Datatilsynet's website for guidance on such questions.

Cloud Computing: Security and Privacy in figure 1.6. One essential skill – worth revisiting – is the practical use of cryptography to protect files. This technique applies not only to individual files but also to entire directory structures, which can be compressed into a single archive before encryption. Knowing how to encrypt files and folders is invaluable: if you encrypt your data before uploading it to the cloud, it remains secure even in the event of a provider breach. Two widely used, cross-platform tools that implement open cryptographic standards are 7-Zip[1] and OpenSSL[2]. Below are examples demonstrating AES-256 encryption and decryption of a file named `s.txt`:

```
# The following is based on the PowerShell environment
# you will set up in the lab.

# 7-zip:
choco install -y 7zip     # install 7-zip
7z a -p s.txt.7z s.txt    # encrypt
7z e s.txt.7z             # decrypt

# OpenSSL:
choco install -y openssl  # install OpenSSL
# Open a new PowerShell window (to reload environment variables
# such that PowerShell will find the newly installed openssl)
openssl enc -aes-256-cbc -a -pbkdf2 -in s.txt -out s.txt.enc    # encrypt
openssl enc -aes-256-cbc -a -pbkdf2 -d -in s.txt.enc -out s.txt # decrypt
```

[1] www.7-zip.org
[2] www.openssl.org

In general, the same considerations that apply when outsourcing a service – such as contracts, trust, and reliability – also apply when outsourcing infrastructure to the cloud. Maintaining an accurate inventory of assets can become more challenging, though not impossible. In fact, moving to the cloud often requires organizations to strengthen their asset management practices to ensure visibility before migrating specific components. However, cloud environments are typically far more dynamic than traditional on-premise setups, which makes maintaining a clear and up-to-date inventory significantly harder.

Migrating to a cloud environment can also provide significant security benefits. Typically, cloud infrastructures are implemented using the concept of *infrastructure as code* (IaC). This approach involves describing your infrastructure in a specialized language and deploying it by executing code that the system interprets. IaC enables rapid and consistent provisioning, making it easier to replace or restore services quickly in the event of an incident.

1.4 OpenStack

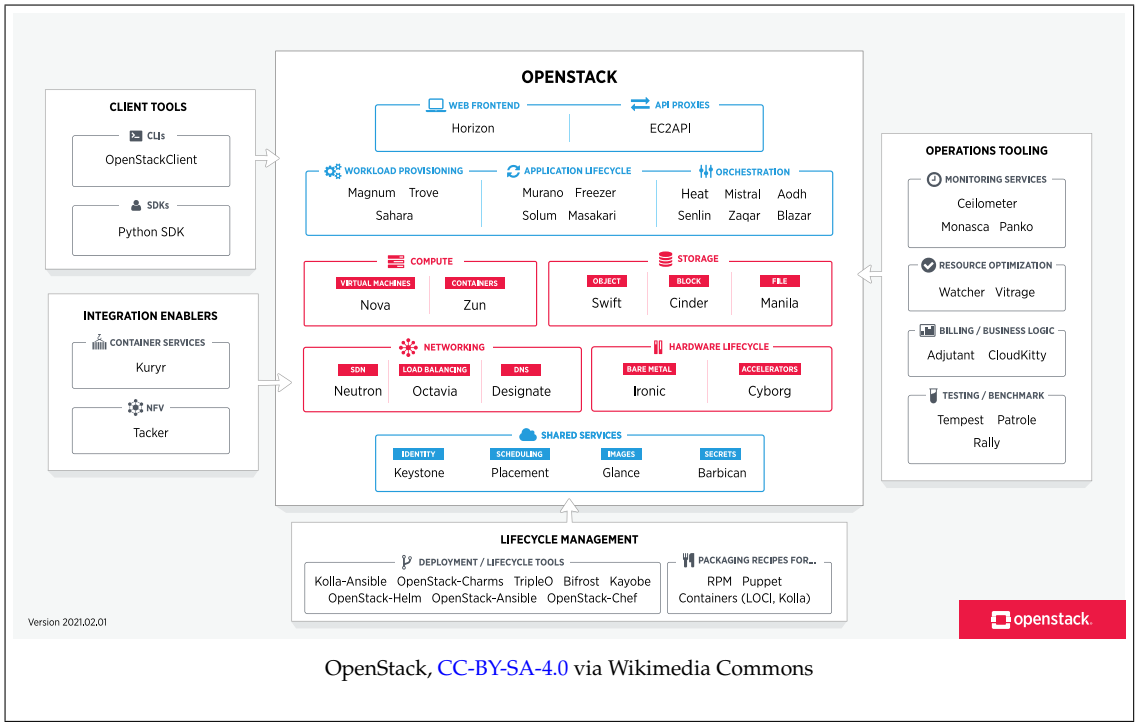
The original announcement of OpenStack[14] included the following mission statement:

> To produce the ubiquitous Open Source Cloud Computing platform that will meet the needs of public and private clouds regardless of size, by being simple to implement and massively scalable.

OpenStack is an open-source project consisting of modular software components that can be combined to build public or private cloud environments. Organizations can choose to implement only the components they require. At NTNU, several OpenStack deployments exist; in this course, we will use the largest implementation, known as SkyHiGh[3]

OpenStack Components in figure 1.7. SkyHiGh currently implements several OpenStack components, including Horizon, Magnum, Heat, Nova, Swift, Cinder, Neutron, Keystone, and Glance (this list may expand over time). Each component can run on its own set of physical servers. We use the term ”set of physical servers” because a secure core infrastructure requires redundancy – meaning there are N installations of the same component, where $N > 1$. Each component provides one or more services, accessible via a specific port number and IP address. For example, when you create a virtual disk (to attach to a virtual machine) in SkyHiGh by clicking ”Volumes” and ”Create Volume”, the Horizon web application sends a command (”create a virtual hard disk of size ...”) to port 8776 [62] on one of the servers running the appropriate service. The

[3]SkyHiGh was established in 2011 as a cloud platform (“Sky”) at Høgskolen i Gjøvik (HiG), which inspired its name, SkyHiGh.



OpenStack, CC-BY-SA-4.0 via Wikimedia Commons

Figure 1.7: OpenStack Components.

commands available for a service are defined by its API (Application Programming Interface), which specifies the operations you can perform.

1.4.1 SkyHiGh

SkyHiGh consists of three racks of physical servers, collectively hosting the various OpenStack components. While each component runs on its own dedicated servers, redundancy is achieved by deploying multiple instances of the same component across different servers, ensuring high availability and fault tolerance.

The SkyHiGh racks at NTNU campus Gjøvik in figure 1.8. When working with SkyHiGh in this course, you will primarily interact with the following components:

Horizon The web-based dashboard used to log in at `https://skyhigh.iik.ntnu.no`. It provides a graphical interface for managing cloud resources.

Heat The *orchestration* service responsible for creating and managing stacks, which are collections of resources such as virtual machines, networks, and security groups. A stack allows you to treat multiple resources as a single unit for deployment and deletion. You will deploy your first stack in the lab exercise.

Cinder The *block storage* service (IaaS) used to create virtual disks that can be attached to virtual machines.

Figure 1.8: The SkyHiGh racks at NTNU campus Gjøvik.

Swift The *object storage* service (PaaS) for storing files, either publicly or privately.

Although you will not manually create virtual machines or networks, you will interact with these components indirectly through orchestration with Heat. Other components, such as Nova and Neutron, will also be involved behind the scenes – for example, when Horizon retrieves information about running instances from Nova.

SkyHiGh is available exclusively to NTNU employees and students, making it a *private cloud*. OpenStack is also widely used to build *public clouds* [26], similar to platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud. Notably, one of the largest OpenStack deployments is the private cloud at CERN, which boasts more than 300,000 CPU cores [68].

1.5 From C to PowerShell

In your programming course, you were introduced to the concept of a data structure known as a *struct* in the C programming language. A struct:

```
// Define the data type Tidspunkt:
struct Tidspunkt  {
  int time,
      minutt,
      sekund;
};
// Create an instance (a variable called tid)
// of the data type Tidspunkt:
struct Tidspunkt tid;
// Assign values to its members:
tid.time   = 17;
tid.minutt = 23;
tid.sekund = 57;
```

PowerShell is built around the concept of *objects*, which are similar to the struct data structures you know from C. It functions both as a programming language and a command-line interface. While it resembles the Linux command line, the key difference is that PowerShell operates on objects (including piping between them) rather than simple streams of bytes or characters, as in Linux. You will encounter PowerShell objects representing files, directories, processes, users, groups, packages, log entries, and more. For example, executing the command `Get-Date` returns an object of type `DateTime`, whose members – called *properties* in PowerShell – can be accessed directly:

```
PS> Get-Date
  Sunday, December 17, 2023 5:38:32 PM
```

```
PS> (Get-Date).DayOfWeek
  Sunday

PS> $tid = Get-Date

PS> $tid.DayOfWeek
  Sunday
```

Note: The following PowerShell tutorial is designed to cover all the essential concepts you will need throughout the semester. It includes more material than can realistically be mastered in just two weeks. For now, focus on reading the section "PowerShell tutorial 1: Standalone Host" 1.6 to gain an initial overview, and prioritize the lab tutorial along with the review questions and problems. At this stage, you only need to practice on a single host. Later, when we reach the chapters on Active Directory, you will begin studying the second part, "PowerShell tutorial 2: Domain-Joined Hosts and Remoting" 1.7.

The tutorial includes links (highlighted in blue) to official Microsoft documentation for various PowerShell concepts. You are not required to read all of them, but they are available if you want to explore a topic in greater depth. Additionally, the tutorial provides exercises that you should attempt as you progress through the material.

1.6 PowerShell tutorial 1: Standalone Host

This first part of the tutorial assumes you are working on a standalone Windows Server 2025. While most tasks can also be performed on Windows 11, we will primarily use server environments throughout the course, so it's beneficial to become familiar with them early on.

1.6.1 Windows Introduction and Background

Windows have GUI (the Windows Shell), the old command line interface `cmd` and the preferred command line interface PowerShell (`cmd` and PowerShell can also both be used in Windows Terminal).

To copy and paste on the command line:

- copy: mark the text and hit Enter
- paste: right click

1.6.2 Getting Started with PowerShell

Note: In the future, we may switch to using winget for installing PowerShell Core and other software. For now, however, we will continue using Chocolatey as described below

By default, Windows includes the older Windows PowerShell (version 5.1). For this course, we will use the newer, cross-platform version called PowerShell Core. Note that the executable for Windows PowerShell is `powershell.exe`, while PowerShell Core uses `pwsh.exe`. PowerShell Core can also be installed on Linux and macOS, where it is launched using the same `pwsh` command. To install PowerShell Core, follow these steps:

```
# search your machine for powershell
# start PowerShell with "Run as administrator"
# run the following commands:
# (you can copy and paste all lines at once)

Set-ExecutionPolicy Bypass -Scope Process -Force
[System.Net.ServicePointManager]::SecurityProtocol = `
  [System.Net.ServicePointManager]::SecurityProtocol -bor 3072
iex ((New-Object System.Net.WebClient).DownloadString(`
  'https://chocolatey.org/install.ps1'))
choco install -y powershell-core
exit
```

PowerShell uses ExecutionPolicies as a safety feature to help prevent accidental execution of scripts, especially those downloaded from the Internet. This is designed to protect you from running potentially harmful code unintentionally.

Sometimes, you may need to change the execution policy to allow your own scripts to run.

- Use `Get-ExecutionPolicy` to check the current policy on your system.
- Use `Set-ExecutionPolicy` to change it when necessary.

Execution Policy is not a strong security boundary. It is a very weak protection because any user with sufficient permissions can easily change the policy using `Set-ExecutionPolicy`. Therefore, it should not be relied upon as a primary security measure.

A commonly used setting is `RemoteSigned`, which requires that any script downloaded from the Internet must be digitally signed by a trusted publisher before it can run.

By default:

- Windows client systems running Windows PowerShell use `Restricted`, which blocks all scripts. Windows client systems running PowerShell Core use `RemoteSigned`.

- Windows servers use `RemoteSigned`.
- On non-Windows systems (such as Linux or macOS), the default policy is `Unrestricted`, meaning scripts can run without signature requirements.

If you download a script that is not digitally signed, you can still run it by manually unblocking the file using `Unblock-File`.

What is a digital signature? A digital signature is a cryptographic mechanism that verifies the authenticity and integrity of a file. It ensures that the script was created by a trusted publisher and has not been altered since it was signed. This helps protect against tampered or malicious code.

Chocolatey is a powerful package manager for Windows that simplifies installing and updating software. However, it's important to understand the risks involved – particularly around *supply chain security*. When you install software from community repositories, you are trusting that the packages have not been tampered with.

Before using Chocolatey extensively, it is advised to read about Chocolatey's Rigorous Moderation Process for Community Packages to understand how packages are reviewed and approved before you rely on them.

What is supply chain security? Supply chain security refers to the practice of ensuring that software and its dependencies are obtained from trusted sources and have not been compromised during distribution. A malicious actor could inject harmful code into a package or its dependencies, so verifying integrity and authenticity is critical when using community-driven repositories. You will learn more about supply chain security in chapter 8.

You can use Chocolatey to install most of the software you need (as we did with PowerShell Core). It also allows you to manage updates easily:

- Check for outdated packages with `choco outdated`
- Upgrade installed packages with `choco upgrade all`

Important: Do Not Run PowerShell as Administrator Unless Required

Before we continue, make sure you are not running PowerShell as Administrator unless explicitly instructed to do so. Elevated privileges should only be used for system administration tasks such as installing software, adding users, stopping critical services, and similar operations.

Why is this important? Humans make mistakes – computers do not. If you accidentally run a harmful command with administrative privileges, the consequences can be severe. By operating as a standard user with limited permissions, you greatly reduce the risk of causing system-wide damage.

1.6.3 Cmdlets, Parameters, Aliases and Help

PowerShell's commands are called *cmdlets* (pronounced "commandlets") and follow the syntax `verb-noun`, for example `Write-Output`. Cmdlets are often used together with *parameters* (note: in Linux man pages, parameters are called *options*), and parameters can have values (also known as *arguments*).

It is advised to read the documentation About command syntax for a deeper understanding.

Long Command Lines Sometimes PowerShell command lines can become too long to fit on a single line. Similar to Linux's line continuation character backslash (\), PowerShell uses a line continuation character called the *backtick* (`).

The backtick is not needed if the line ends with a pipe character (|). Whenever you see a line ending with a backtick or a pipe, it means the command continues on the next line.

In the beginning it can be hard to remember all the different cmdlets. Fortunately most of the cmdlets have aliases that correspond to commands you might already know from DOS (cmd.exe) or Unix/Linux. In addition, PowerShell provides short aliases for many cmdlets to make typing faster. To find the cmdlet associated with a command you know, you can use the `Get-Alias` cmdlet:

```
# is there a cmdlet corresponding to Unix/Linux ls?
Get-Alias ls
# are there many aliases to Get-ChildItem?
Get-Alias -Definition Get-ChildItem
# list all aliases
Get-Alias
# or list them from the "alias drive" (similar to your C:\ drive)
Get-ChildItem Alias:\
```

Aliases can be convenient for quick typing, but it is recommended to use the full cmdlet names, especially in your scripts. Using full names makes your scripts more readable, easier to maintain, and simpler for others to understand.

Each cmdlet has several parameters that are specific to its functionality, but there is also a set of common parameters available across most cmdlets. Some particularly useful common parameters include `-Verbose`, `-ErrorAction SilentlyContinue` and `-WhatIf`.

> **DO THIS**: Try `Get-Service | Stop-Service -WhatIf`.
> Try `Get-ChildItem -Recurse C:\Windows\System32\wbem\MO*` with and without the option `-ErrorAction SilentlyContinue`.

A running program in Windows is called a *process*, and most processes contain one or more *threads* that execute instructions concurrently. A service is a special type of process that runs in the background, often without any user logged in. Services typically handle system-level tasks such as networking, printing, or security. Sometimes, a process or a group of processes can be wrapped in a job. Jobs allow commands or scripts to run asynchronously in the background, but this is less common than services.

To get help with cmdlets, use the `Get-Help` cmdlet. For example:

- `Get-Help Write-Output | more` displays the help text one page at a time.

A useful feature is the ability to view the help page online by adding the `-Online` parameter:

- `Get-Help Write-Output -Online` opens the documentation in your browser.

If you prefer offline help, you can use:

- `Get-Help Write-Output -Examples` to see practical examples.
- `Get-Help Write-Output -Full` for the complete help content.

PowerShell supports *TAB-completion* for both cmdlets and parameters. This feature allows you to type part of a command or parameter and press the TAB-key to automatically complete it. TAB-completion saves time, reduces typing errors, and helps you discover available commands and options.

> **DO THIS**: Use `Get-Help Select-String -online` (note: `Select-String` is similar to Linux `grep`) and find the example code for "Find a string in subdirectories". Try it. Did you get an access error (red text)? If so, repeat the command but add the parameter `-ErrorAction SilentlyContinue` to the `Get-ChildItem` cmdlet. This will suppress error messages and allow the command to continue. Searching file systems can be slow. How slow? Repeat the command, but wrap it in braces (`{ }`) and prepend it with the `Measure-Command` cmdlet. This will measure the time it takes to execute the command.

To find which cmdlets you can use, the primary tool is the `Get-Command` cmdlet, e.g. Try `Get-Command *DNS*`. (For a bit of fun, try `Get-Command | Get-Random | Get-Help`. Repeat this a few times to explore random cmdlets and learn what they do!)

1.6.4 Variable, Array and Hashtable

A variable in PowerShell can represent any kind of object. Variables can also be grouped into collections such as *arrays* or *hashtables*:

Array: A collection where each element is accessed by a numeric index (e.g., `$myArray[0]`).

Hashtable: A collection of key-value pairs where each element is accessed by a key instead of a numeric index (e.g., `$myHash["Name"]`).

```
# A variable can hold any object
$a = 'hei'
$a.GetType().FullName          # System.String (string object)

# Use a reliable file path for demonstration
# $PSHOME points to the PowerShell installation directory
$a = Get-Item "$PSHOME\powershell.exe"
$a.GetType().FullName          # System.IO.FileInfo (file object)

# Arrays: 0-based numeric index
$a = @('frodeh','kelly')
$a.GetType().FullName          # System.Object[] (array)
$a[0]                          # 'frodeh'
$a[1]                          # 'kelly'
$a                             # Prints the two elements

# Add elements to an array (creates a new array under the hood)
$a += 'erik'
$a                             # Now contains 'frodeh','kelly','erik'
$a.Count                       # 3

# Hashtables: key-value pairs, indexed by key (not by numeric index)
$a = @{'frodeh'='Frode Haug'; 'kelly'='Jia-Chun Lin'}
$a.GetType().FullName          # System.Collections.Hashtable (hashtable)
$a['kelly']                    # 'Jia-Chun Lin'
$a[1]                          # Usually $null | hashtables use keys,
                               # not numeric index
$a                             # Prints the key-value pairs
$a.Count                       # Number of entries (2)
$a.Keys                        # Lists keys: 'frodeh','kelly'

# Add or update entries in a hashtable
$a['erik'] = 'Erik Hjelmås'    # Add new key-value
$a['kelly'] = 'Kelly Lin'      # Update value for existing key
```

```
$a.Keys                          # 'frodeh','kelly','erik'
```

DO THIS: 1. Using the cmdlet `Write-Output` and the hashtable `$a` above (the last example), write a command line that outputs "User frodeh has real name Frode Haug". 2. Repeat this exercise, but this time use the `-f` format operator (see Format operator -f). (hint: in the first exercise you must use `$(...)` to insert the value from the hashtable into the string, while in the second exercise you do not need this.)

1.6.5 Objects and `Get-Member`

The "Power" in PowerShell comes from working with *objects* rather than just plain text or byte streams. This means that when you pipe data between cmdlets, you are passing rich objects with properties and methods – not just strings.

Example: Resolving a DNS Name If you want to look up the IP address of a Fully Qualified Domain Name (FQDN), such as `ftp.uninett.no`, you can use the old-style `nslookup` program or the modern PowerShell cmdlet `Resolve-DnsName`:

```
# Using nslookup (legacy tool)
nslookup.exe ftp.uninett.no
nslookup.exe ftp.uninett.no | Get-Member
# Notice: TypeName is System.String - nslookup outputs plain text only.

# Using Resolve-DnsName (PowerShell cmdlet)
Resolve-DnsName ftp.uninett.no
Resolve-DnsName ftp.uninett.no | Get-Member
# Notice: TypeName is Microsoft.DnsClient.Commands.DnsResponseRecord
# This is an object with properties like IP4Address.
```

Accessing Properties Because `Resolve-DnsName` returns an object, you can easily access its properties:

```
(Resolve-DnsName ftp.uninett.no).IP4Address
```

Doing the same with `nslookup` requires complex text parsing:

```
(nslookup.exe ftp.uninett.no 2>&1 | `
  Select-String '\d+\.\d+\.\d+\.\d+').Matches[-1].Value
```

Why `Get-Member`? When piping objects, use `Get-Member` to inspect which properties and methods the object contains. To access a property or method, use:

```
Object.Property
(Cmdlet-that-returns-object).Property
```

> **DO THIS**: Do `$name = 'mysil'`. Use the properties and methods of the `$name`-object to
>
> - Find out how many characters the string contains.
> - Print the string in upper case.

1.6.6 Select-Object

The `Select-Object` cmdlet is useful in several contexts:

1. Inspecting Properties You can use `Select-Object` to display the values of all properties of an object. This is sometimes more practical than `Get-Member` when learning about an object's properties:

```
Resolve-DnsName ftp.uninett.no | Select-Object -Property *
```

2. Creating a Reduced Object You can create a new object in the pipeline with a reduced set of properties. Compare the output of these two commands:

```
# Create a file
Write-Output mysil > abc.txt

# Full object with all properties
Get-ChildItem .\abc.txt | Get-Member

# Reduced object with selected properties
Get-ChildItem .\abc.txt | `
  Select-Object -Property Name, LastWriteTime | Get-Member
```

3. Acting Like Head/Tail in Unix/Linux `Select-Object` can also be used to select the first or last items in a collection, similar to `head` and `tail` in Unix/Linux:

```
Get-Process | Select-Object -Last 5
Get-ChildItem | Select-Object -First 3
```

> **DO THIS:** Run `Get-Content C:\Windows\System32\drivers\etc\services` and pipe it to `Select-Object` so that you only output lines 30 to 50 (meaning: from line number 30 to line number 50).

4. Advanced: Creating New Properties on the Fly Sometimes we use `Select-Object` to create new properties dynamically. For example, if we want to show the owner of files (which is not directly available as a property), we can use a calculated property with a hash table:

```
Get-ChildItem |
  Select-Object Name, Directory, @{Name="Owner"; `
    Expression={(Get-ACL $_.FullName).Owner}}, `
    CreationTime, LastAccessTime | Format-Table
```

Notice the syntax for calculated properties:

- **Name**: The header.
- **Expression**: A script block that computes the value for each pipeline object (`$_`).

1.6.7 Filtering with `Where-Object`

Use the automatic variable `$_` to reference the current pipeline object. `Where-Object` filters objects similarly to `grep` in Unix/Linux.

```
# List all services:
Get-Service

# Services with Status exactly equal to 'Running':
Get-Service | Where-Object { $_.Status -eq 'Running' }

# Combine conditions (-eq for equality, -like for wildcard):
Get-Service | Where-Object { $_.Status -eq 'Running' -and `
  $_.StartupType -like '*auto*' }

# Use -match for regular expressions:
Get-Service | Where-Object { $_.Status -eq 'Running' -and `
  $_.StartupType -match '.*auto.*' }
```

Browse the documentation for all comparison operators: About Comparison Operators.

> **DO THIS:** Start another instance of PowerShell Core so that you have at least two running. Use `Get-Process` and `Where-Object`:
>
> - List all processes where `Name` equals `pwsh`.
> - List all processes where `WorkingSet` is greater than 10MB.

- Use `Get-Member` or `Select-Object -Property *` to find the property showing the full path to `pwsh.exe`.

Tip: For efficiency, filter as far left as possible:
`Get-Process -Name notepad` is better than
`Get-Process | Where-Object { $_.Name -eq 'notepad' }`.

1.6.8 Formatting and Output Cmdlets: `Format-*` and `Out-*`

These cmdlets are used to *format or redirect the final output* of a pipeline. **Important rule:** *filter to the left, format to the right*. Apply filtering and selection before formatting.

Format-* Cmdlets Examples: `Format-Table`, `Format-List`. Purpose: Control how data is displayed on the screen. *Use them only for display*, not for saving data or further processing. If you need to select properties for saving or further processing, use `Select-Object` instead.

```
Get-ChildItem -File C:\Windows\ |
  Format-List -Property Name, Length, LastAccessTime
Get-ChildItem -File C:\Windows\ |
  Format-Table -Property Name, Length, LastAccessTime
```

Out-* Cmdlets Examples: `Out-File`, `Out-GridView`, `Out-Printer`. Purpose: Send output to a destination other than the console.

Common uses:

```
# Save to a file
Get-ChildItem -File b* | Out-File b.dat    # Same as > b.dat

# Send to a graphical grid view
Get-ChildItem -File b* | Out-GridView

# Copy to clipboard
Get-ChildItem -File b* | clip

# Export to CSV
Get-ChildItem -File b* | Export-Csv $home\a.csv
```

DO THIS: TAB through all `ConvertTo-` cmdlets to see available options for converting objects to different formats.

Tip: PowerShell automatically adds `Out-Default` at the end of every pipeline, which can sometimes cause unexpected behavior. See example when using semicolons (the same thing can happen inside scripts/functions).

1.6.9 Sorting Objects with `Sort-Object`

The cmdlet `Sort-Object` allows you to sort objects based on any property. This is extremely useful when working with large sets of data or when you need to organize output for easier analysis.

Below are two practical examples, including the use of `Get-Date` to filter recently accessed files. Note that PowerShell commands can span multiple lines:

```
# Sort files accessed in the last 30 minutes by size (descending)
Get-ChildItem C:\Windows\System32\ |
  Where-Object { $_.LastAccessTime -gt (Get-Date).AddMinutes(-30) } |
    Sort-Object -Property Length -Descending

# Same filter, but select the 10 smallest files
Get-ChildItem C:\Windows\System32\ |
  Where-Object { $_.LastAccessTime -gt (Get-Date).AddMinutes(-30) } |
    Sort-Object -Property Length |
      Select-Object -Last 10
```

Why Time Stamps Matter

Time stamps are critical in many contexts:

- Log analysis and forensic investigations (building timelines).
- Backup and restore operations.
- Everyday tasks, such as finding which file you edited or downloaded last Thursday.

  DO THIS: List all processes where the name matches the wildcard expression `'*host*'` (meaning the process name contains the word `host`). Sort the output by the property `CPU`, then pipe to `Format-Table` and display only the properties `Name`, `Id`, and `CPU`.

1.6.10 Measuring Objects with `Measure-Object`

You have already used `Measure-Command` to measure execution time. If you want to count objects, lines, words, characters, or compute statistics from object properties, use `Measure-Object`.

```
# Count how many objects were returned in the DNS example
Resolve-DnsName ftp.uninett.no | Measure-Object

# Count lines, words, and characters in a text file
Get-Content C:\Windows\System32\drivers\etc\services |
  Measure-Object -Line -Word -Character

# Count files and calculate total size in System32
Get-ChildItem -File C:\Windows\System32\ | Measure-Object

# Compute sum and average of file sizes
Get-ChildItem -File C:\Windows\System32\ |
  Measure-Object -Property Length -Sum -Average
```

This cmdlet is useful for:

- Counting objects in a pipeline.
- Computing statistics such as sum, average, minimum, and maximum.
- Analyzing text files for lines, words, and characters.

  DO THIS: Repeat the previous exercise (processes named `*host*`), and use `Measure-Object` to:

  - Count the number of processes.
  - Compute the average of the property `WorkingSet` (memory usage).
  - Store the output in a variable `$avghostproc`.
  - Output the average value in megabytes (see example in the hashtable section).
  - Format the value with two decimals using: `{0:N2} -f (expression)` (see also Format operator -f).

1.6.11 Iterating with `ForEach-Object`

While `Where-Object` is used for filtering objects in the pipeline, `ForEach-Object` is used when you want to *perform an action on each object* in the pipeline. This provides flexibility for custom operations.

You can also use standard loops and conditions such as `foreach` or `if`, which are typically used in scripts (collections of commands stored in a file).

```
# Simple foreach loop: open Notepad 4 times
foreach ($i in 1..4) { notepad }

# Terminate all Notepad processes with a custom message
Get-Process notepad | ForEach-Object {
    Write-Output "Terminating Notepad with id $($_.Id)"
    $_.Kill()
}
```

Note: If you only want to stop all Notepad processes, you can simply use:

```
Get-Process notepad | Stop-Process
```

> **DO THIS:** Start a pipeline with the strings
> `'Set-Location', 'Get-Location', 'Push-Location'`
> and use `ForEach-Object` to iterate over these cmdlet names. For each name, call: `Get-Alias -Definition $_`

1.6.12 Conditional Logic: `if`, Comparisons, and Branching

PowerShell supports conditional logic using the `if`-statement. This allows you to execute code only when a condition evaluates to `True`. Conditions often involve comparison operators such as `-eq` (equals), `-gt` (greater than), and logical operators like `-and` or `-or`.

```
# Check if the Administrator account is enabled
if ( (Get-LocalUser -Name Administrator).Enabled ) {
    Write-Output "It's enabled!"
}

# Check if a string is not empty
if ( $name.Length -gt 0 ) {
    Write-Output "Name has characters"
```

```
}

# Check if a file exists
if ( Test-Path C:\Windows\System32\ntoskrnl.exe ) {
    Write-Output "OS exists!"
}
```

Key Points to Remember:

- Use parentheses `(...)` around the condition.
- Enclose the code block in curly braces `{ ... }`.
- Combine conditions with logical operators (`-and`, `-or`).
- Explore comparison operators in About Comparison Operators.

Sometimes it is easier to use `Compare-Object`, for example, when you need to copy only new or updated files between two directories.

```
# Define source and destination directories
$src = Get-ChildItem -Recurse "C:\Users\Admin\test\"
$dst = Get-ChildItem -Recurse "C:\Backup\"

# Handle empty collections to avoid errors
if ($src -eq $null -or $dst -eq $null) {
    Write-Output "Compare-Object does not like empty objects :)"
    return
}

# Compare based on LastWriteTime and copy newer files
Compare-Object -ReferenceObject $dst -DifferenceObject $src `
    -Property LastWriteTime -PassThru |
        Where-Object { $_.SideIndicator -eq '=>' } |
            Copy-Item -Recurse -Force -Destination C:\Backup\
```

When using `Compare-Object` with the `-PassThru` option, PowerShell adds a property called `SideIndicator` to each object in the pipeline. This property indicates the difference:

- `=>` means the object exists only in the `DifferenceObject` (source).
- `<=` means the object exists only in the `ReferenceObject` (destination).

You can use `SideIndicator` to filter and select objects based on how they differ.

1.6.13 Extending PowerShell with Modules

PowerShell can be extended with modules to add new functionality. For example, to manage Windows Update through PowerShell, you can install the `PSWindowsUpdate` module.

```
# Find available modules related to Windows Update
Find-Module *windowsupdate*

# Install the PSWindowsUpdate module (run as Administrator)
Install-Module -Name PSWindowsUpdate

# Check for updates
Get-WindowsUpdate

# Download updates
Get-WindowsUpdate -Download

# Install updates (optional, may take time)
# Get-WindowsUpdate -Install
```

1.6.14 Creating and Using PowerShell Scripts

A PowerShell script is simply a collection of commands stored in a file (usually with the extension `.ps1`). There are several ways to create a script file:

- Use a text editor (e.g., VS Code, nano, vim, Notepad).
- Use `Out-File` (`>`) or append with `Out-File -Append` (`>>`).
- Use a here document (called a here-string in PowerShell).

For small scripts, a here-string is practical. The example below creates a script `myscript.ps1` that accepts a parameter and displays the file size:

```
@'
param($Name)
"Hello $Name, the size of this file is in Bytes:"
(Get-ChildItem .\myscript.ps1).Length
'@ > myscript.ps1
```

DO THIS: Create the script above and execute it:

- With and without providing the parameter `Name`.
- With and without assigning a value to `Name`.

Sometimes you will see `&` as a prefix when executing scripts or commands, especially when using TAB-completion. For example, rename `myscript.ps1` to `my script.ps1` and try TAB-completion. `&` is the *call operator*. Learn more in About Operators.

Checking Script Quality Always check your scripts for code quality. PowerShell provides tools for this:

- PSScriptAnalyzer: A static code analyzer for PowerShell.
- Pester: A testing framework for larger projects.

```
# Install PSScriptAnalyzer
Install-Module -Name PSScriptAnalyzer

# Analyze script quality
Invoke-ScriptAnalyzer myscript.ps1
# No output means no issues found :)
```

1.6.15 Drives, Profiles, Variables, and Namespaces

PowerShell introduces the concept of *drives*, which are data stores that you can access like a file system. This might seem unusual at first, but it is very practical because it allows you to reuse the same set of commands for accessing different types of data.

Use `Get-PSDrive` to list all available drives. Navigate (`cd`) into drives you did not know about and explore their contents using `Get-ChildItem` or `Get-ChildItem -Recurse`. For example:

```
Get-ChildItem -Path Cert:\LocalMachine\CA
```

This command lists the Certificate Authorities on the local machine.

Variables in PowerShell are stored in the `variable` namespace by default. PowerShell supports multiple namespaces, and sometimes you need to specify them explicitly.

```
# Working with variables
$a = 'heisan'
Write-Output $a
Write-Output $variable:a
```

```
# Environment variables
cd $env:homepath
$env:homepath
$homepath
$home
$env:home

# Profile script location
$profile
Get-Content $profile   # Similar to 'cat' in Linux
```

- *Environment variables* are stored in the `env` namespace. Learn more: About Environment Variables.
- A *Profile* is a script that runs when PowerShell starts. It can be used to customize your environment (e.g., aliases, functions, settings). Learn more: About Profiles.

1.7 PowerShell tutorial 2: Domain-Joined Hosts and Remoting

Note: The remainder of this chapter will only be relevant when we cover Active Directory. You may skip it for now.

In this section, we focus on managing remote hosts without relying on a full Remote Desktop Protocol (RDP) GUI session. RDP (port 3389) has been associated with security vulnerabilities in recent years, and automating administrative tasks through a GUI is often inefficient.

PowerShell Remoting allows you to execute commands or establish sessions on remote hosts using either the WSMan or SSH protocols. In this tutorial, we will focus on WSMan for communication between Windows hosts. We will explore several scenarios where the client is a Windows 11 machine (the host initiating the connection) and the server is Windows Server 2025 (the host receiving the connection).

*Note: We are now going to create a **Windows Domain**. A domain includes at least one domain controller running Active Directory services, along with client and server computers that join the domain by registering in Active Directory. This setup allows users to log in to any domain-joined computer using their Active Directory account (instead of a local account) and enables centralized management of groups of computers.*

Active Directory is a core component of Windows-based infrastructures, and almost every organization uses it. Because of its critical role, Active Directory is a highly attractive target for attackers. A well-known example is the ransomware attack on Norsk Hydro in 2019, which had an estimated cost of roughly NOK 800 million.

Why Active Directory Knowledge Matters for Azure and Microsoft Entra ID Understanding Active Directory (which is an on-premises directory service) is useful for working with Microsoft Entra ID (formerly Azure AD, a cloud-based identity and access management solution). Many concepts—such as users, groups, and authentication—are similar, and this knowledge helps when configuring hybrid identity solutions and secure access in cloud environments.

This lab assumes you have two Windows 11 hosts called `mgr` and `cl1`, and two Windows Server 2025 called `dc1` and `srv1`. You can create this environment by following the instructions in the lab tutorials towards the end of this chapter, and use the Heat template cl_dc_srv_basic.yaml. We install the services Active Directory and DNS on `dc1`, join `cl1`, `mgr` and `srv1` to the domain. We have chosen `sec.core` as our domain name. Before we create a domain, you need to set a password for the local Administrator user (who will become domain administrator) and you need to have a password for safe mode. For simplicity, we let these be the same password (in production environments these should be unique of course). Choose a password before moving on (by "Choose a password" we mean either to use your password manager to generate a password for you or create a password following best practice). On `dc1` do (notice that since `Install-ADDSForest` have many parameters, we use splatting)

1.7.1 Install the Domain Controller

```
# run as administrator
$Password = Read-Host -Prompt 'Enter Password' -AsSecureString
Set-LocalUser -Password $Password Administrator
$Params = @{
    DomainMode                    = 'WinThreshold'
    DomainName                    = 'sec.core'
    DomainNetbiosName             = 'SEC'
    ForestMode                    = 'WinThreshold'
    InstallDns                    = $true
    NoRebootOnCompletion          = $true
    SafeModeAdministratorPassword = $Password
    Force                         = $true
}
Install-WindowsFeature AD-Domain-Services, DNS -IncludeManagementTools
Install-ADDSForest @Params
Restart-Computer
# Log in as SEC\Administrator with password from above, test our domain
Get-ADRootDSE
Get-ADForest
Get-ADDomain
# Any computers joined the domain?
Get-ADComputer -Filter *
```

1.7.2 Join hosts to the domain

On `cl1`, `mgr` and `srv1` (who will join the domain) the following *needs to be done in Windows PowerShell* (NOT in PowerShell Core) since `Add-Computer` is not supported in PowerShell Core

```
# run as administrator
Get-NetAdapter | Set-DnsClientServerAddress `
  -ServerAddresses IP_ADDRESS_OF_DC1
$cred = Get-Credential -UserName 'SEC\Administrator' -Message 'Cred'
Add-Computer -Credential $cred -DomainName sec.core -PassThru -Verbose
Restart-Computer
```

(Btw, it is possible to do this in PowerShell core by doing something like `Import-Module Microsoft.PowerShell.Management -UseWindowsPowerShell` first, see about_Windows_PowerShell_Compatibility)

1.7.3 Remoting

By default, when you establish a remote connection you will enter into Windows PowerShell on the remote host. If you want to enter into PowerShell core on the remote host you have to do

```
# on the host you are connecting to
Install-PowerShellRemoting.ps1
# on the host you are connecting from
Enter-PSSession mgr -ConfigurationName PowerShell.7
```

By default, remoting is only enabled on servers, so if you want to connect to a client host (e.g. Windows 11) you have to do

```
# on client hosts (e.g. Windows 11)
Enable-PSRemoting
```

Learning PowerShell Remoting through Practice

> Log in to `cl1` as `CL1\Admin`, read and do all the examples in the article About Remote. After doing this, you should be comfortable with `Enter-PSSession`, `New-PSSession` and `Invoke-Command`.

1.7.4 Generating Passwords

Best Practice: Never create passwords manually unless you absolutely need to type them repeatedly. If you do, follow modern password recommendations: use a long, hard-to-guess password. Note that current guidelines no longer require complex composition rules.

For most cases, always use a secure random password generator for services, databases, accounts, and cryptographic keys. Avoid prompting for passwords with `Read-Host` unless necessary.

Windows PowerShell (5.1): You can use the built-in .NET method:

```
Add-Type -AssemblyName 'System.Web'
[System.Web.Security.Membership]::GeneratePassword(20,0)
```

PowerShell Core: Since .NET Core lacks this functionality, use `Get-Random` (a strong random number generator (RNG) since PowerShell 5.0):

```
# store possible characters in a string variable
$chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPRSTUVWXYZ' +
  '0123456789!#$%&()*+,-./:<=>?@[\]^_{|}~'
# select 20 random characters and join them into a string
-join ($chars.ToCharArray() | Get-Random -Count 20)
```

Certain characters such as backticks (`), quotes (" and '), and semicolons (;) are excluded because they can cause parsing issues or break scripts when used in PowerShell commands.

Validating Credentials: To verify a username/password combination (Windows PowerShell):

```
$username = 'SEC\Administrator'
$password = Read-Host -MaskInput -Prompt 'Password'
$computer = $env:COMPUTERNAME

Add-Type -AssemblyName System.DirectoryServices.AccountManagement
$obj = New-Object `
 System.DirectoryServices.AccountManagement.PrincipalContext('machine',$computer)
$obj.ValidateCredentials($username, $password)  # Returns $true if valid

Clear-Variable password
```

This approach can be useful for troubleshooting authentication issues or confirming that credentials work as expected.

1.7.5 Hint: removing autocomplete suggestions

If you don't like the autocomplete suggestions in PowerShell Core, you can disable them with

```
New-Item -ItemType Directory -Path (Split-Path $PROFILE)
Set-PSReadLineOption -PredictionSource None
# To make this permanent, add the line below to your profile script
Write-Output 'Set-PSReadLineOption -PredictionSource None' >> $profile
```

1.8 Lab tutorials

1. **Creating Infrastructure in SkyHiGh Using Heat Templates** In this lab, you will create your own infrastructure in *SkyHiGh* by using predefined monolithic Heat Orchestration Templates (HOT templates), which contain all configuration details in a single file. You will access the SkyHiGh environment through the web interface at https://skyhigh.iik.ntnu.no (note: this site is only accessible from within NTNU's network or via VPN). The interface you log in to is *Horizon*, the OpenStack dashboard that acts as a graphical front-end communicating with the OpenStack API on your behalf. When you select *"Orchestration"* in Horizon, you are using Horizon to interact with *Heat*, the orchestration component of OpenStack, which is responsible for deploying and managing infrastructure based on the templates you provide.

   (a) Log in (NTNU credentials) at https://skyhigh.iik.ntnu.no

   (b) Create a keypair unless you already have one:

      i. Compute
      ii. Key Pairs
      iii. Create Key Pair
      iv. Key Pair Name `mykey` (or whatever name you prefer)
      v. Create Key Pair. *This command will generate a file named `mykey.pem`, which contains **your private key**. This file is critical for secure access, so make sure you control where it is stored and keep it safe*. The corresponding public key is automatically stored in OpenStack. It is used to generate passwords for Windows instances and injected into Linux instances to enable key-based authentication.

   (c) Create your infrastructure:

      i. Orchestration
      ii. Stacks
      iii. Launch Stack
      iv. Template Source: Direct Input
      v. Template Data: copy the RAW file (middle button on the right side) of single_windows_server.yaml.
      vi. (You do not have to choose "Environment Source" or "Environment File")
      vii. Stack Name: `Mysil` (or whatever name you prefer)
      viii. Enter your NTNU-password
      ix. key_name: `mykey` (or whatever you chose as the name in the step earlier)
      x. Launch
      xi. Wait 5-15 minutes for Windows instances, 1-5 minutes for Linux instances.

xii. If you would like, you can monitor progress for each instance by clicking Network, Network Toplogy, choose an instance, Console

xiii. Click on Compute, Instances and make a note of the IP addresses starting with `10.212`. Those are the floatingIPs which are publically available (but restricted to access only from NTNU networks) and you use when accessing your instances with RDP or SSH.

(a) Your username is `Admin`. Find your password:

i. Compute

ii. Instances

iii. Retrieve Password (right side drop-down menu)

iv. Private Key File: upload `mykey.pem` (or copy and paste its contents)

v. Decrypt Password

vi. Your password will be displayed

(b) To connect to a Windows instance, use a remote desktop client application (`mstsc` on Windows, `Windows App` from App store on Mac, or `xfreerdp` on Linux)

Always be prepared for quickly deleting and recreating your stack, that's how we take advantage of what the cloud offers us, and helps us get used to modern DevOps thinking: frequent well-tested changes.

1.9 Review questions and problems

1. What is the purpose of NSM's Grunnprinsipper for IKT-sikkerhet 2.1 framework?
2. Name the four main categories of principles in NSM's framework.
3. Why is it important to understand Windows infrastructure in enterprise environments?
4. What technological and methodological changes led to the rise of DevOps?
5. Why has Git become central in modern IT operations?
6. Define cloud computing in simple terms.
7. What are the three main service models in cloud computing? Provide an example for each.
8. What is meant by on-premise infrastructure?
9. What is OpenStack, and what is its primary purpose?
10. What type of cloud is SkyHiGh, and who can access it?
11. What do you consider as a major *privacy* concern/threat when you use a public cloud? Give an example including a technical measure (a technical solution) to prevent it.
12. What is *orchestration*? Which component in OpenStack provides this service?
13. (Do all the following exercises in the Windows Server you created in the lab tutorial. Start by installing PowerShell Core as shown earlier in this chapter.)
14. Write a PowerShell command that lists *only directories* (not files) inside your home directory and all its subdirectories. In other words, the command should search *recursively* through every folder under your home directory.
    (Hint: use `Get-Help -Online Get-ChildItem` to explore available parameters.)
15. Split the following one-line command into four lines using backticks (`` ` ``), so that each parameter appears on its own line.
    `Get-ChildItem -Recurse -Path C:\Windows\System32\wbem\MO* -File`
16. Use the cmdlet `Get-Date` to store the current date and time in a variable named `$now`. Then, using the variable `$now`, perform the following tasks:
    (a) Output only the Year.
    (b) Output only the DayOfWeek.
    (c) Output only the Hour.

(d) Output only the TimeOfDay.

(e) Convert the date to universal time (UTC).

(f) Output the date and time 45 minutes ago.

(g) Output the date and time 3 months from now.

17. You are preparing a list of server names for a maintenance script. Create an array named `$servers` containing the following three servers: `'web01','db01','filesrv01'`. Then:

    (a) Output the first server in the list.

    (b) Add a new server `'backup01'` to the array.

    (c) Display the total number of servers in the array.

18. You are documenting local groups on a Windows Server. First, populate a hashtable `$groups` (key = group name, value = group description) using the following two lines:

```
$groups = @{}
Get-LocalGroup | ForEach-Object { $groups[$_.Name] = $_.Description }
```

    Then do the following:

    (a) Output the entire hashtable.

    (b) Output the description for the group `'Administrators'`.

    (c) Add a new entry mapping `'HelpDesk'` to `'Tier-1 support team'`.

    (d) Change the description for `'Users'` to `'Built-in standard users (non-admins)'`.

    (e) Show how many entries are in the hashtable.

19. Without using an intermediate variable, perform the following tasks directly with the output of `Get-Date`:

    (a) Display only the current Year.

    (b) Display only the current DayOfWeek.

    (c) Display the date and time 30 minutes ago.

    (d) Display the date and time 2 months from now.

20. *Select-Object — four use-cases (local groups).* Use `Get-LocalGroup` and apply `Select-Object` in four ways:

    (a) Inspect all properties: For a single group (e.g., the first one), show all properties using `-Property *`.

(b) Create a reduced object: Show only `Name`, `Description`, and `SID` (Security Identifier) for all groups.

(c) Head/Tail behavior: Display only the last 5 groups returned.

(d) (Advanced exercise) Calculated property: Add a column `MemberCount` that counts how many members each group has, then show `Name`, `MemberCount`, and `Description`.

21. Write a PowerShell command that lists all processes whose `StartTime` is within the last 12 hours. (Hint: use `Get-Date` with `AddHours(-1)`.)

22. Write a PowerShell command that lists all files in the directory `C:\Windows` that are larger than 10 KB. Then extend the pipeline to:

    (a) Sort the output by file size (`Length`) in descending order.

    (b) Display only the three largest files.

    (c) Show only the following properties:
    `Name`, `Length`, `LastAccessTime`, and `LastWriteTime`.

    (Hint: Use `Where-Object`, `Sort-Object`, and `Select-Object`.)

23. Use `Get-Process` to list all running processes. Export only the following properties to a CSV file named `processes.csv` in your home directory: `Name`, `Id`, and `CPU`. Finally, verify the export by opening the file in Notepad.

    (Hint: use `Select-Object`, `Export-Csv`, and `notepad`.)

24. Write a PowerShell command that lists the *names of all directories* (starting from your current directory) that contain *at least 10 items* (files or subdirectories). Note: you might need to switch to a directory with many subdirectories/files to test this command, e.g.
    `cd 'C:\Program Files\WindowsPowerShell\'`

    (Hint: use `Measure-Object` to count items inside each directory.)

25. Write a PowerShell script named `trynet.ps1` that performs the following tasks:

    - Accepts two command-line parameters: `-MyHost` (hostname) and `-MyPort` (port number).
    - Uses `Test-NetConnection` to test connectivity to the specified host and port, and measures how long the test takes.
    - Outputs a different message depending on whether the connection attempt was successful.
    - Suppresses warnings by using the `-WarningAction` parameter (see about_commonparameters).

    The script should behave like this:

```
PS> .\trynet.ps1 -MyHost localhost -MyPort 3389
Test was successful and took 1069.6293ms
PS> .\trynet.ps1 -MyHost localhost -MyPort 3390
Test was not successful and took 5138.0973ms
```

Make sure you run *PSScriptAnalyzer* to check the quality of your script.

26. Read about_splatting. Then use splatting to write a command that lists *only files* (not directories) under `C:\Windows\System32\LogFiles` *recursively*, and ensures that errors are suppressed by setting `ErrorAction` to `SilentlyContinue`.

2
Windows Server

2.1 Windows Server

Computers can often be divided into two categories: *client* and *server*. These terms also apply to computer programs. For example, a web browser is a client program that communicates with a web server. For operating systems running on physical hardware (”bare metal”) or in a virtual machine, a client is characterized by having a user present who actively uses interactive programs (such as on a laptop, tablet, or phone). A server, on the other hand, is characterized by offering services to clients or other servers. Windows 11 is a version of the Windows operating system configured for use as a client, whereas Windows Server is configured to function as a server. While clients also run services, these are typically for internal use only.

2.1.1 Process and Service

Processes in figure 2.1. When you execute a program on an operating system, the running instance of that program is referred to as a *process*. For example, when you run Notepad, you are executing the code contained in the file `notepad.exe`, which results in a running process named Notepad:

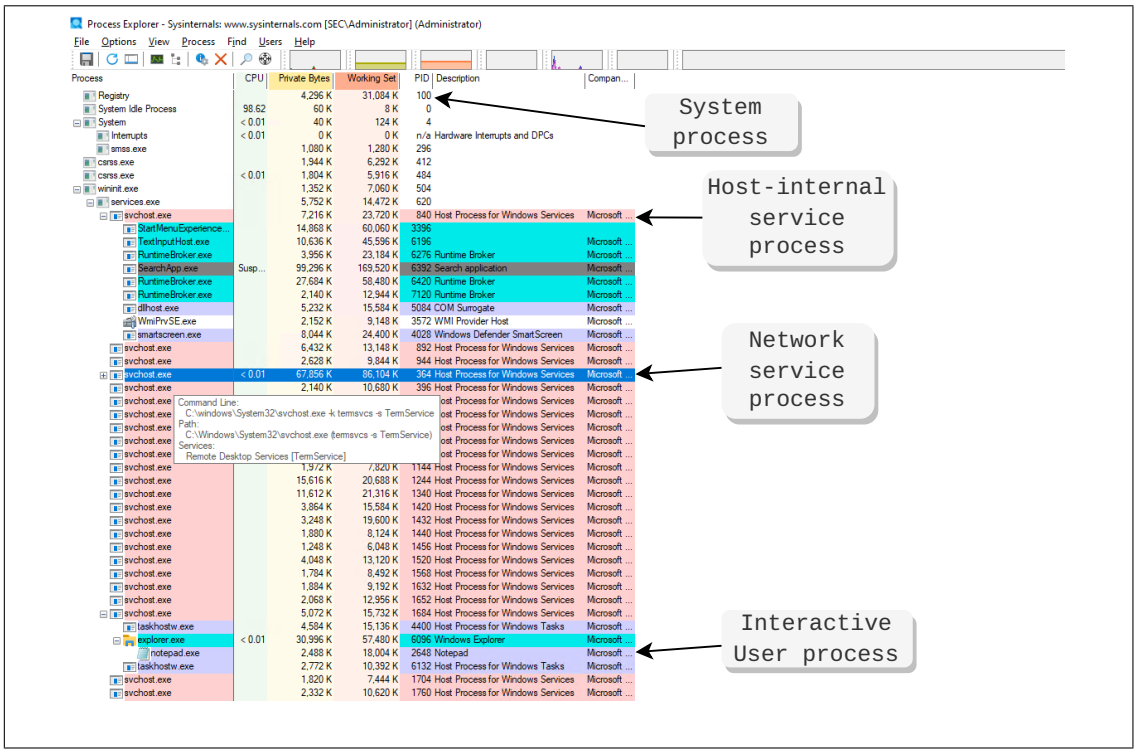


Figure 2.1: Processes.

```
PS> notepad.exe
PS> Get-Process notepad -IncludeUserName

Handles      WS(K)   CPU(s)     Id UserName               ProcessName
-------      -----   ------     -- --------               -----------
    216      20352     0.14   6288 DC1\Admin             notepad
```

Notepad is a typical *interactive process* designed to be used directly by a user. Notice that each process has an *owner*. In this example, the Notepad process is owned by the user `Admin` logged into the *host* `dc1` (a computer running an operating system is often called a *host* because it hosts many processes, files, users, and other resources).

A process can be single-threaded or multithreaded, meaning it contains one or more *threads*. Most programs you have written so far are single-threaded – when you don't explicitly create threads in your code, you get a single-threaded process. A thread is a lightweight sub-process that runs within a process. You will learn more about this in the Operating Systems course, but for now, understand that a process needing to use multiple CPU cores simultaneously must be multithreaded. The operating system allocates CPU cores to threads, not to processes. You can check how many threads each process has using this PowerShell command:

```
Get-Process |
  Select-Object -Property Name, ID, `
    @{Name='ThreadCount';Expression ={$_.Threads.Count}} |
  Sort-Object -Property ThreadCount -Descending
```

Opposed to an interactive process, a *service* (or *service process*) is a process running in the background, typically started during the computer's boot sequence. A service runs independently of any logged-in users and is usually owned by a dedicated system account. Services can be divided into two categories: *host-internal services* and *network services*:

host-internal service A service that supports the operating system or performs local system maintenance, such as Windows Update or Task Scheduler. Some critical host-internal services are called *system processes* [8], which are essential to Windows operating system functionality.

network service A service that listens on a port for incoming connections from clients, such as Remote Desktop Services, which listens on port 3389 by default.

Network Service Processes in figure 2.2. Hosts (computers) consist of hardware, an operating system, and applications. Both the operating system and applications run as processes on the hardware. Hosts on the Internet communicate with each other by specifying an *IP address* and a *port number*. A port is considered "open" if a network

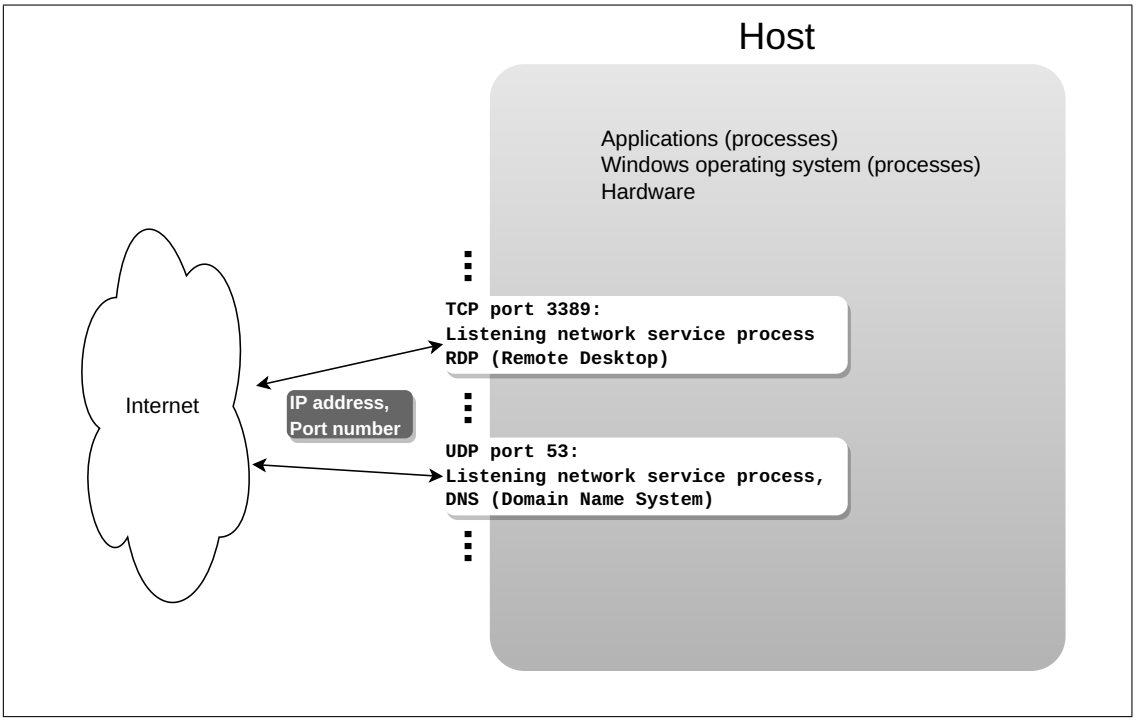


Figure 2.2: Network Service Processes.

service process is listening on it for incoming connections. There are two types of ports: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Both TCP and UDP use 16-bit port numbers, meaning port numbers range from 0 to 65535 (since $2^{16} = 65536$).

Identifying which service processes are listening for incoming network connections is often useful for troubleshooting, security analysis, and forensic investigations:

```
Get-NetTCPConnection |
  Select-Object -Property LocalAddress,LocalPort,State,OwningProcess,`
  @{Name='ProcessName';Expression={(Get-Process `
    -Id $_.OwningProcess).ProcessName}},`
  @{Name='ServiceName';Expression={(Get-CimInstance -ClassName `
    Win32_Service -Filter "ProcessID=$($_.OwningProcess)").Name}} |
  Format-Table -AutoSize

# A more readable version of the above command:

# Define calculated properties for Select-Object. Btw, this is
# what is called "Splatting" (putting parameters into a variable)
$selectParams = @(
  'LocalAddress',
```

```
  'LocalPort',
  'State',
  'OwningProcess',
  @{
    Name = 'ProcessName'
    Expression = {
      (Get-Process -Id $_.OwningProcess).ProcessName
    }
  }
  @{
    Name = 'ServiceName'
    Expression = {
      Get-CimInstance `
        -ClassName Win32_Service `
        -Filter "ProcessID=$($_.OwningProcess)" |
        Select-Object -ExpandProperty Name
    }
  }
)

# Run pipeline
Get-NetTCPConnection |
  Select-Object @selectParams |
  Format-Table -AutoSize
```

Service processes must comply with the protocol and rules established by the *Service Control Manager* (a special *system process* called `services`). This means that all service processes must be able to respond to messages from the Service Control Manager, such as Stop, Start, Suspend, and Resume. We can use PowerShell to instruct the Service Control Manager to send these control messages to services:

```
PS> Get-Command *service | Format-Table -Property Name

Name
----
Get-Service
New-Service
Remove-Service
Restart-Service
Resume-Service
Set-Service
Start-Service
Stop-Service
Suspend-Service
```

Service processes can be either stand-alone processes or wrapped individually or together with other services in a *service host process* named `svchost`. The easiest way to view all services and identify which services are contained within a service host process is to use the legacy `cmd` program with the command `tasklist /svc`. When a service is wrapped in a service host process, the service is loaded into that process from a DLL (Dynamic Link Library). This means the service does not run as a separate process but instead exists only within the service host process. Historically, service host processes were used to conserve system resources, since each process consumes a significant amount of memory on Windows. However, with modern 64-bit Windows running on 64-bit hardware, memory is no longer a critical constraint, and most service host processes today contain only a single service [36].

Understanding the concepts of processes and services is essential for troubleshooting issues on Windows systems. For example, if you encounter a "hanging service" (a service that fails to stop when you attempt to stop it), you can locate the service host process containing the service and attempt to stop that process instead, which may resolve the issue.

2.1.2 Accounts

As mentioned previously, every process is owned by an account on the system. To get a quick overview of which processes are owned by which users, you can run the following PowerShell command (note that this must be executed with Administrator privileges):

```
Get-Process -IncludeUserName |
  Select-Object -Property Name,UserName,ID |
  Sort-Object -Property UserName,ID
```

Files and directories are also owned by accounts on the system. To implement any kind of access control or security, we need to manage ownership of all objects on the system. To view which local accounts (*user accounts* and *system accounts*) exist on your Windows system, use:

```
# List user accounts
Get-LocalUser
# or
Get-CimInstance Win32_UserAccount
# List system accounts
Get-CimInstance Win32_SystemAccount
```

We can also use PowerShell to see a list of differences between user and system accounts. The key difference is that user accounts have several password-related fields because they are intended for interactive login sessions (and thus as owners of interactive user processes), whereas system accounts are used as owners of service processes.

Notice how the "SideIndicator" points to the right for entries in the "DifferenceObject" (the user account stored in the variable `$u`). This indicates that these properties only exist in `$u` and not in `$s`:

```
PS> $s = Get-CimInstance Win32_SystemAccount | Get-Member
PS> $u = Get-CimInstance Win32_UserAccount | Get-Member
PS> Compare-Object -ReferenceObject $s -DifferenceObject $u

  InputObject                                  SideIndicator
  -----------                                  -------------
  uint AccountType {get;}                      =>
  bool Disabled {get;set;}                     =>
  string FullName {get;set;}                   =>
  bool Lockout {get;set;}                      =>
  bool PasswordChangeable {get;set;}           =>
  bool PasswordExpires {get;set;}              =>
  bool PasswordRequired {get;set;}             =>
  PSStatus {Status, Caption, PasswordExpires} =>
  PSStatus {Status, SIDType, Name, Domain}     <=
```

2.1.3 Configuration Data

All applications consist of executable code combined with *configuration data*. Configuration data specifies how an application should behave – for example, which port should a network service listen on, or where should Microsoft Word store documents by default?

Registry

The Registry [76] is a database that stores most of the configuration on a Windows host. Its purpose is to centralize configuration in one location, rather than scattering it across multiple files throughout the file system. In earlier versions of Windows (before 2000), applications typically stored configuration in separate `.ini` files (some still do), but today most applications store their configuration in the registry instead.

The Registry in figure 2.3.

- HKLM contains the sub keys SAM (user account database), SECURITY, SOFTWARE and SYSTEM which have corresponding files on disk. It also contains the sub key HARDWARE which is generated at runtime (similar to the `proc` filesystem on Linux).
- HKCC just links into HKLM.

HKEY_LOCAL_MACHINE (HKLM) Config for local computer, SAM, SECURITY, SOFTWARE and SYSTEM, files in `C:\Windows\System32\config`

HKEY_CURRENT_CONFIG (HKCC) link to `HKLM\System\CurrentControlSet\Hardware Profiles\Current`

HKEY_USERS (HKU) each user profile actively loaded on the machine

HKEY_CURRENT_USER (HKCU) link to currently logged on user in `HKU`

HKEY_CLASSES_ROOT (HKCR) a compilation of `HKCU\Software\Classes` and `HKLM\Software\Classes`

Figure 2.3: The Registry.

- HKU is user specific config data for each active user profile (note: a user profile can be active without the user being logged in).
- HKCU just links into HKU.
- HKCR stores file associations and is just linked into HKLM and HKCU.

You can edit the registry using either the GUI tool `regedit.exe` or PowerShell commands such as `Get-ItemProperty` and `Set-ItemProperty`.

In production environments on a larger scale, you almost never modify the registry directly. Instead, you change it indirectly using configuration management tools such as Group Policy. Here are some examples of retrieving data from the registry using PowerShell:

```
# List installed software that can be uninstalled:
Get-ChildItem `
  HKLM:\software\microsoft\windows\currentversion\uninstall |
  ForEach-Object {Get-ItemProperty $_.PSPath} |
  Format-Table -Property DisplayName
# List all local firewall rules (messy)
# (combine into one line):
Get-ItemProperty HKLM:\System\CurrentControlSet\Services\
  SharedAccess\Parameters\FirewallPolicy\FirewallRules
```

WMI

Windows Management Instrumentation in figure 2.4. The CIM standard defines a large

- CIM (Common Information Model)
- WMI: objects with properties and methods grouped into namespaces
- Mostly *config data from registry* and *performance* data (from the operating system)

Figure 2.4: Windows Management Instrumentation.

set of management objects that are designed to represent "everything that can be managed on a host" in a standardized way. These CIM objects have properties and methods. WMI is Microsoft's implementation of CIM, which uses WMI providers (objects implemented as Dynamic Link Libraries – approximately 100 in total) to provide access to these CIM-based objects. Since the number of CIM objects is extremely large, they are organized into *namespaces*.

Note: Much of the information accessible and modifiable through WMI originates from the registry. However, WMI also provides access to information beyond the registry, such as performance counters and real-time data from the operating system's internal structures.

```
# Show the list of all namespaces:
Get-CimInstance -Namespace root -ClassName __Namespace

# List all classes in namespace:
Get-CimClass -Namespace root/CIMV2

# List all instances (objects) of class Win32_Processor:
Get-CimInstance Win32_Processor

# List all properties of a specific instance (object):
Get-CimInstance Win32_Processor -Filter "DeviceID='CPU0'" |
  Select-Object -Property *

# Show a specific property of a specific instance (object):
(Get-CimInstance Win32_Processor `
  -Filter "DeviceID='CPU0'").LoadPercentage
```

Make sure you become familiar with the terminology *namespace*, *class*, and *object*. You will encounter these terms frequently throughout various courses this semester. Understanding these concepts is essential, as they form the foundation for many programming principles:

Namespace A namespace is used to group a large number of classes, objects, or variables together. It serves as a high-level category that helps separate different groups

from one another. An analogy for this concept is postal codes (postnummer): for example, 2843 is the postal code for Eina, but it only holds that meaning within the *namespace Norway*. In other countries, the postal code 2843 may represent something entirely different if it is defined there.

Class A class is a definition of an object and serves as an abstract blueprint. When you learned programming in C, you encountered various data types (such as int, float, char, etc.). A data type is analogous to a class; it defines a specific kind of data. For example, an int represents a number that occupies four bytes of memory.

Object An object is a concrete instance of a class. It represents something that actually exists and can be used, similar to how you create a variable in C. For example, in the statement `int i;` the type `int` is analogous to a class, while `i` is analogous to an object.

Note that we use the term *instance* in many situations to describe the concrete realization of something. In the text above, we refer to objects as ”instances of a class.” When using cloud services like SkyHiGh, we call virtual machines ”instances of an image”—or more commonly, just ”instance.” The term ”image” refers to the disk image of a base operating system from which we create our virtual machine (for example, Ubuntu 26.04 or Windows Server 2025).

When studying computer science, you must recognize that some terms have different meanings depending on the context in which they are used. Often, we use different terms interchangeably to refer to the same thing. For example, in this course (and later in your study program), when referring to a virtual machine in SkyHiGh, we may use the terms *instance*, *VM*, *host*, *server*, or simply use the name we have assigned to the virtual machine, such as *DC1*.

2.1.4 Using Windows Server and Installing Software

When you log into a Windows Server, you are automatically greeted with the application *Server Manager*. We will not use Server Manager extensively (we will use the newer Windows Admin Center instead), but it is useful for getting a quick overview of server status and for quickly locating the tools we need (accessible via the *Tools* menu).

Server Manager in figure 2.5. Notice that one of the first things Server Manager proposes is to add *Roles* and *Features* [44]:

Role A server role is a set of software programs that provide a specific function for multiple users or other computers on a network.

Feature Features are software programs that, although they are not directly part of roles, can support or enhance the functionality of one or more roles, or improve overall server functionality, regardless of which roles are installed.

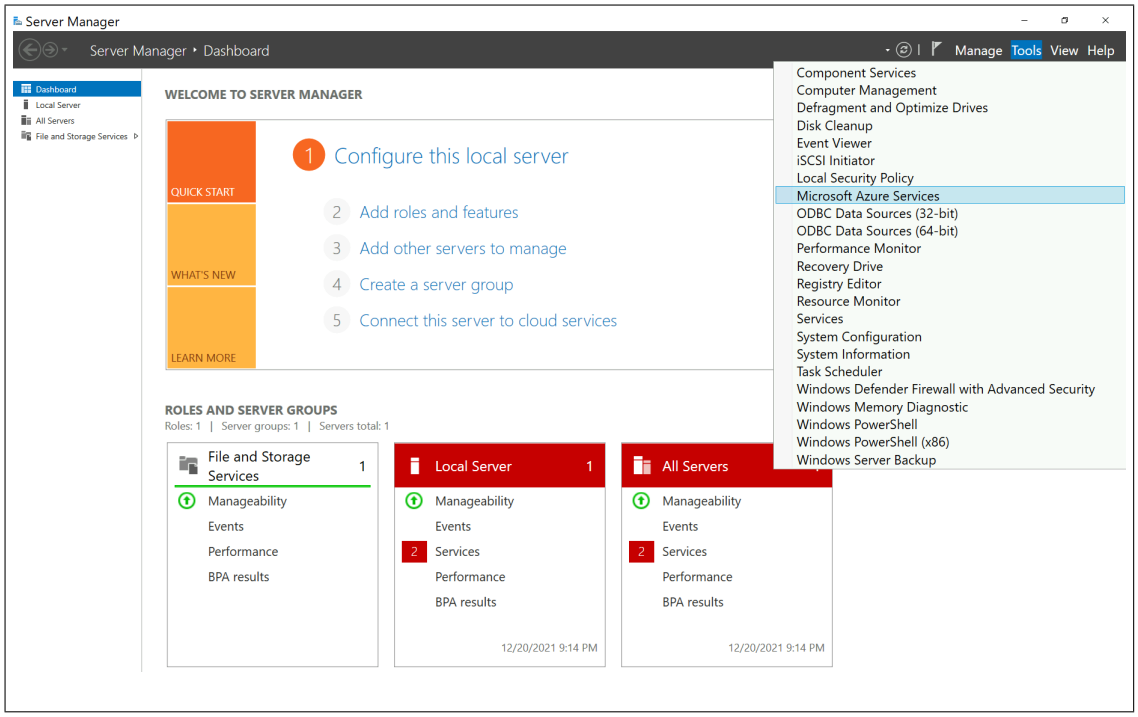


Figure 2.5: Server Manager.

Capability Similar to features but specific to Windows clients, capabilities are software packages that can be installed separately. For example, `OpenSSH.Client` is an installed capability.

Right Click Start-button in figure 2.6. It is also worth noting that many of the same GUI tools for managing a Windows Server can be accessed by right-clicking the Start button.

Sysinternals

Sysinternals in figure 2.7. Sysinternals [47] is a collection of powerful utilities developed and maintained over the past 25 years, primarily by Mark Russinovich. We will install the complete suite and use several of these tools in the exercises:

```
# always check the status of a package before installing, if ok:
choco install -y sysinternals
```

Before installing, always check the package web page status. If you see "Some Checks Have Failed or Are Not Yet Complete," read the "Details" section to understand what it means. Also consult the Chocolatey Community Package Repository documentation [12] for clarification.

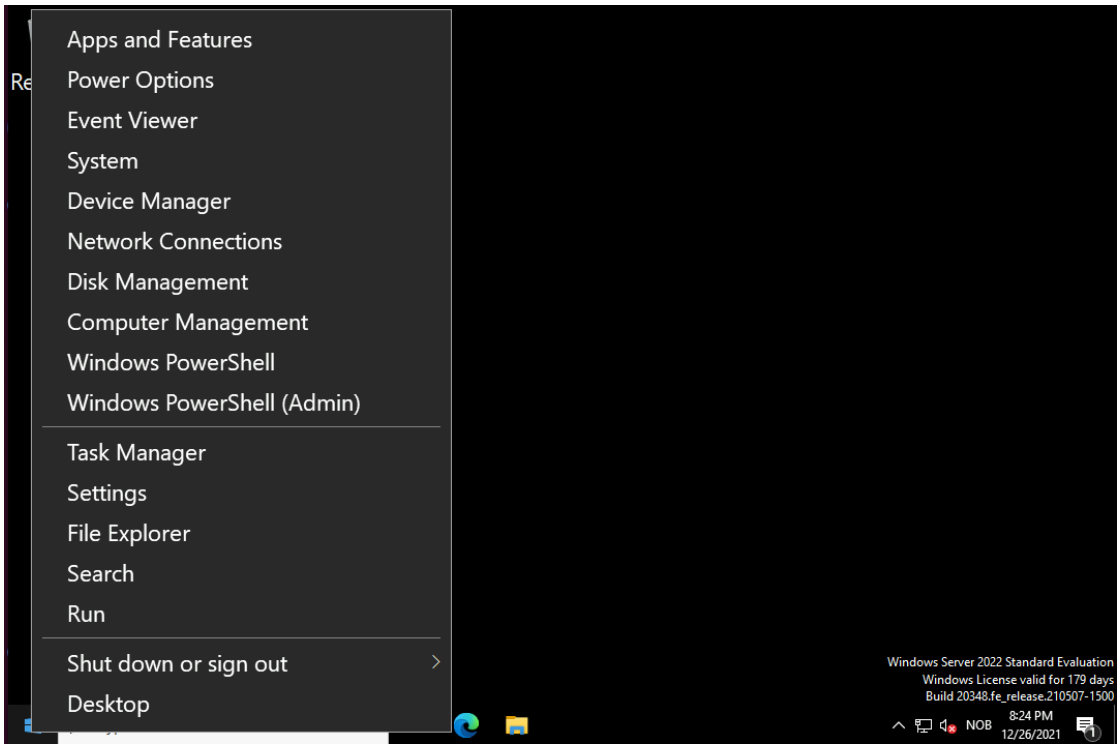


Figure 2.6: Right Click Start-button.

- File and Disk (`AccessChk`)
- Networking (`AD Explorer`, `TCPview`)
- Process (`Process Explorer`)
- Security (`SysMon`, `SDelete`, `AccessChk`)
- System info (`coreinfo`, `RAMMap`)
- Misc (`BgInfo`, `ZoomIt`)

Figure 2.7: Sysinternals.

If you want to deepen your understanding of how Windows works and learn to use Sysinternals utilities effectively, Mark Russinovich has co-authored the book "Windows Internals" [77], which is an excellent resource.

2.1.5 Windows Command Line Interface (CLI)

Before 2006, Windows only had the traditional command line interface (CLI) `cmd` (which is still present on all versions of Windows). PowerShell arrived in 2006 and has since grown continuously in popularity, becoming the standard CLI on Windows hosts. Most tasks on a Windows host can be accomplished with PowerShell, but occasionally PowerShell does not offer the same functionality as the traditional Windows commands [52] from the `cmd` CLI. You saw an example of this earlier when we mentioned the `tasklist` command. Fortunately, these legacy commands work just as well in PowerShell as they do in `cmd`.

In this course, we will primarily use the command line, but we should use a GUI when we need visualization or when a one-time task is significantly simpler with a GUI. The main reasons we prefer the command line interface (CLI) are:

- Commands can be treated as code and stored in a version control system like Git, making your work reproducible and trackable.
- *Humans make errors, computers do not*: with CLI, you can automate your tasks programmatically, ensuring *consistency* across multiple executions.
- It is much easier to document what you are doing by storing commands in scripts rather than capturing screenshots of a GUI interface.

2.2 Lab tutorials

1. No lab tutorial in this chapter, do this chapter's review questions and problems in the Windows Server you created in chapter one.

2.3 Review questions and problems

1. What is a *service* in Windows? Describe briefly.
2. Explain the difference between an *interactive user process*, a *host-internal service*, and a *network service process*. Why is this distinction important in system administration and troubleshooting?
3. Describe the relationship between *processes* and *threads* in Windows. Why is it important for system administrators and developers to understand how threads work?
4. What is *svchost.exe* and why is it used? Explain one advantage and one disadvantage of hosting multiple services inside a single process.
5. Why have modern versions of Windows reduced the number of services hosted inside a single *svchost.exe* process?
6. Compare *user accounts* and *system accounts* in Windows.
7. What is a *WMI* (Windows Management Instrumentation)? Describe briefly.
8. Use PowerShell to identify *open (listening) UDP ports* on your Windows Server, and correlate them with owning processes.
   (a) List all UDP endpoints (local address and local port).
   (b) Add the owning *ProcessId* and resolve the *process name*.

   Hint: use `Get-NetUDPEndpoint`, see chapter example with `Get-NetTCPConnection`.
9. Use PowerShell to analyze how *running services* are mapped to *processes*.
   (a) Which process hosts the largest number of running services?
   (b) Is this process typically a `svchost.exe` process?
   (c) What does this tell you about how Windows services are implemented?

   Hint: use `Get-CimInstance Win32_Service` and `Group-Object`.
10. (Complete this exercise using the Windows Server you created in the lab tutorial from chapter one.) Let's explore roles and features:
    (a) Use Server Manager to install the *role* "Web Server (IIS)".
    (b) Which *feature* is required during installation of this role?
11. (Complete this exercise using the Windows Server you created in the lab tutorial.) Ensure that you have completed the previous exercise where you installed the "Web Server (IIS)" role before proceeding with this one. Let's explore the topic of processes and services:

(a) Start PowerShell as an Administrator (keep PowerShell open for the duration of this exercise). Install Sysinternals (refer to the instructions in the chapter text). Also, start Process Explorer [1] as an Administrator.

(b) Sort the processes by CPU usage. Identify which process consumes the most CPU and explain why you think that is the case. (Note: you may need to scroll to the top to see this.)

(c) Check how much memory/RAM (in the "Working Set" column) PowerShell uses. If you are using Windows PowerShell, the process name is `powershell`; if you have installed PowerShell Core, it is `pwsh`. How much of that memory is private to PowerShell?

(d) Processes are organized in a hierarchical tree structure, where some processes are parents of others. Click three times on the Process tab to switch between views. Identify the parent process of PowerShell and describe what that process is.

(e) Right-click on the Explorer process and choose "Properties." How many threads does this process have?

(f) What command line was used to start the Explorer process?

(g) Start the Sysinternals tool TCPview. Which process is listening on port 80?

(h) Return to Process Explorer, right-click on the process that is listening on port 80, and select "Properties." Go to the TCP/IP tab, and check and uncheck the "Resolve addresses" box to see two different views. Do you feel confident about the information you have gathered regarding the process that is listening on port 80?

(i) Add the "Command line" column to Process Explorer (View, Select Columns, Process Image). Expand the width of the column to view the entire text. Can you find a command line that contains `iissvcs`? If it is difficult to locate, try searching for `iis` in the "Find" menu under "Find Handle or DLL..."; this will lead you to the relevant process IDs (PIDs). Which services does this svchost process provide?

12. Explain how the Registry, WMI, services and the Sysinternals tools relate to each other.

[1] Note: Process Explorer is widely used, for example, in the discovery of a Chromium bug [4] (which was rewarded $3000).

3

Storage, Backup, Restore

3.1 NSM Grunnprinsipper

NSM Grunnprinsipper for IKT-sikkerhet in figure 3.1.

A company's core value is often closely tied to its data: the files and directories it maintains. Losing access to data prevents work from continuing, and if unauthorized parties gain access to confidential company information, the damage can be severe. In this section, we will primarily focus on NSM chapter 2.9 (Establish capability to restore data), though chapters 2.7 (Protect data at rest and in transit) and 2.6 (Control identities and access rights) are also relevant to our discussion.

3.2 Storage and Windows

File-level vs Block-level in figure 3.2. We are accustomed to dealing with the concept of a *file*.

A File has two components: metadata and data. The metadata component varies by file system, but most file systems support owner, permissions, timestamps, and size. (Size is sometimes called length because the file contents can be treated as an array of bytes, and this array has a length.) In addition, the metadata contains information about how to access the data (the actual file contents). In the simplest form, this can be all the block addresses of the data blocks stored in the metadata.

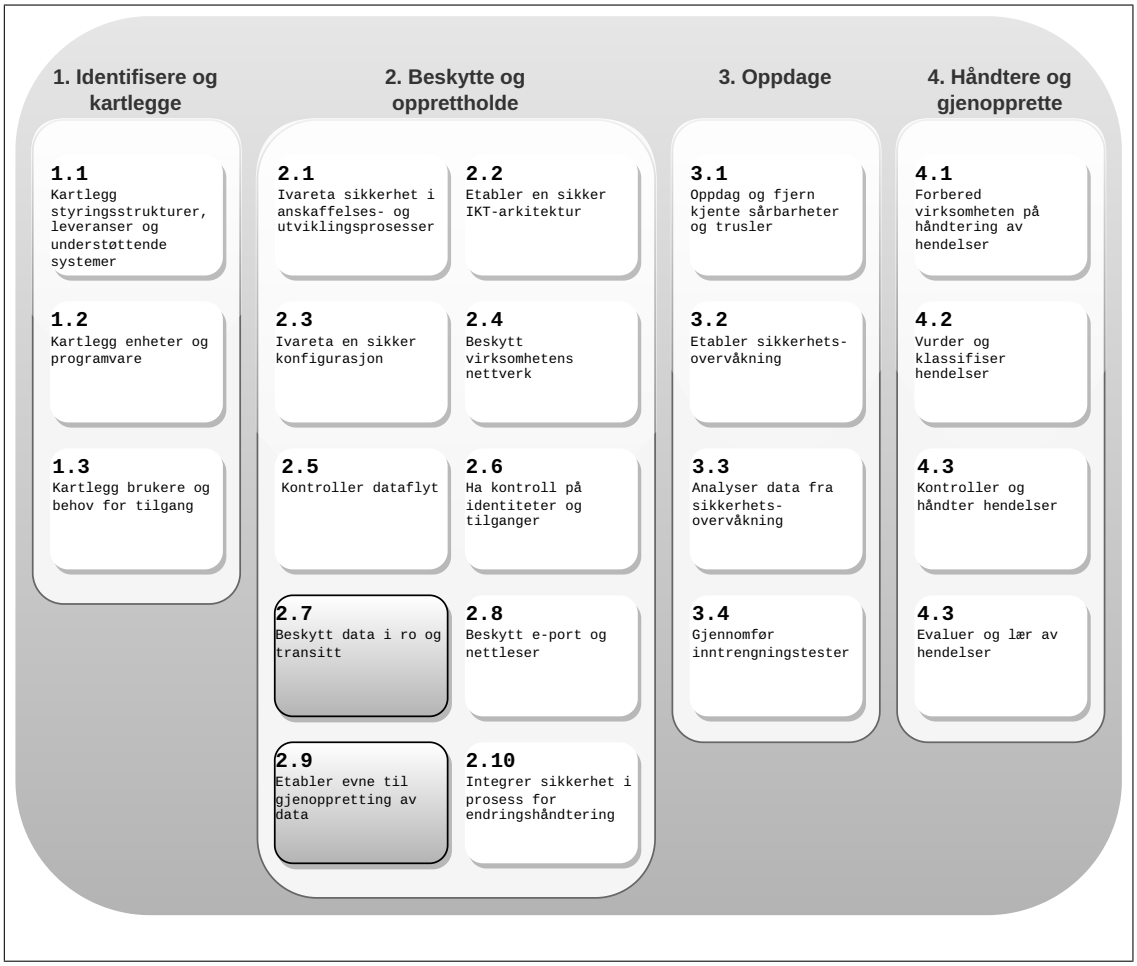


Figure 3.1: NSM Grunnprinsipper for IKT-sikkerhet.

Many of us have used a storage device like a USB hard drive, memory stick, or memory card. A storage device consists of blocks. Before we can use it for files, it needs to have a file system written to it.

When we work with data storage, backup, and restore, we sometimes work with files at the *file-level*, and sometimes we work at the *block-level*. It is important to understand this difference, especially in the cloud and other virtualized infrastructures.

When you use a virtual machine, the block-level storage device is commonly a file. This means you are using a *storage stack* where the file you save in the virtual machine is stored in a file system on a block device, and the block device is actually just a file on a physical server. That file is stored on a file system on another block device. The block device might be the actual hard drive on the server or a block device accessed over the network from a storage solution like Ceph [3], which is the one used in SkyHiGh.

The key point: when you save a file in a virtual machine running in a cloud, many layers of

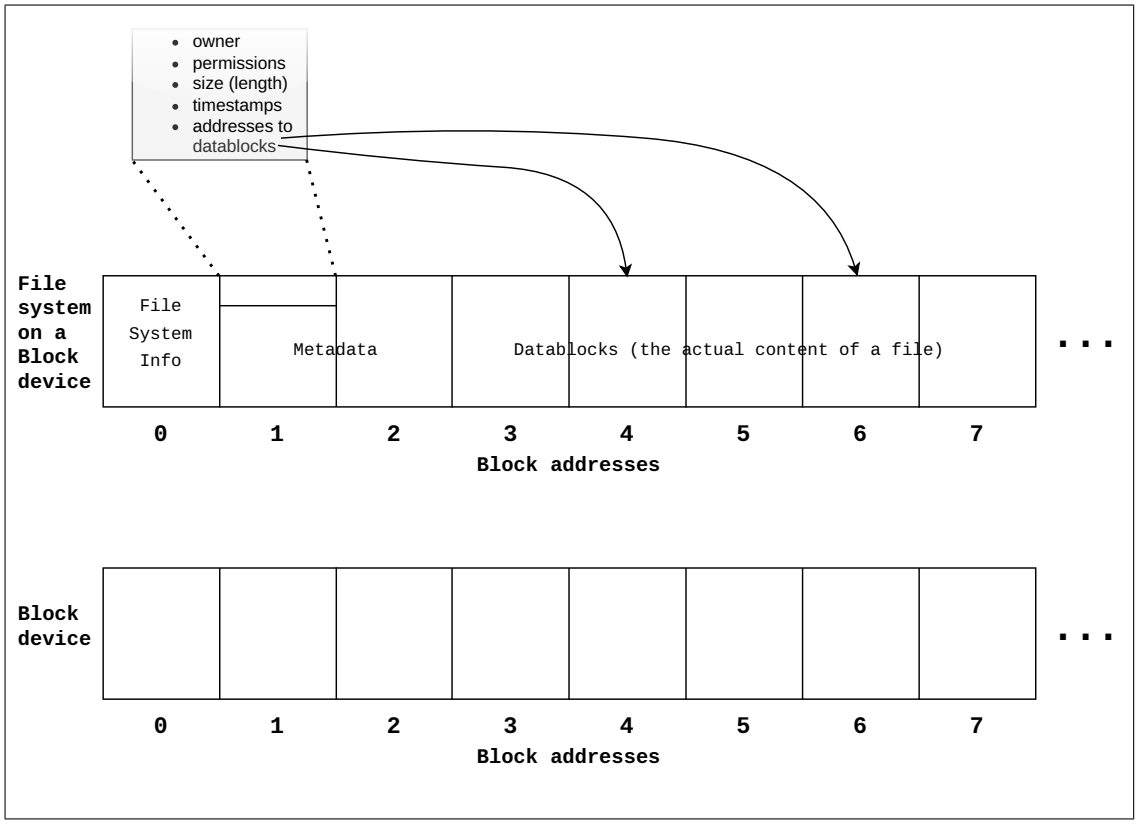


Figure 3.2: File-level vs Block-level.

technology are involved before your data is actually stored as bits and bytes on a physical device. For IT system performance, CPU and memory are rarely a problem, but data storage is often a bottleneck.

PowerShell: File-level vs Block-level in figure 3.3. The following is a summary of key concepts (including corresponding PowerShell cmdlets and command lines) we need to know about:

Disk (`Get-Disk`) What Windows perceives as a physical disk (SSD or HDD). In our case, this is what we create in OpenStack (in the OpenStack component called "Cinder") as a "Volume" and attach to a server. *Important:* Inside our server, this is seen as a physical disk, and we treat it as a physical disk. However, outside our server (in our cloud), this is called a volume.

Partition table (`Initialize-Disk`) On a new physical disk we create a partition table. This is written in the first part of the disk, and is either the old-style "Master Boot Record (MBR)" or the newer (and what we typically use today) "GUID Partition Table (GPT)" [39].

Partition (`Get-Partition`) A partition is a part of a disk (it is defined in the partition

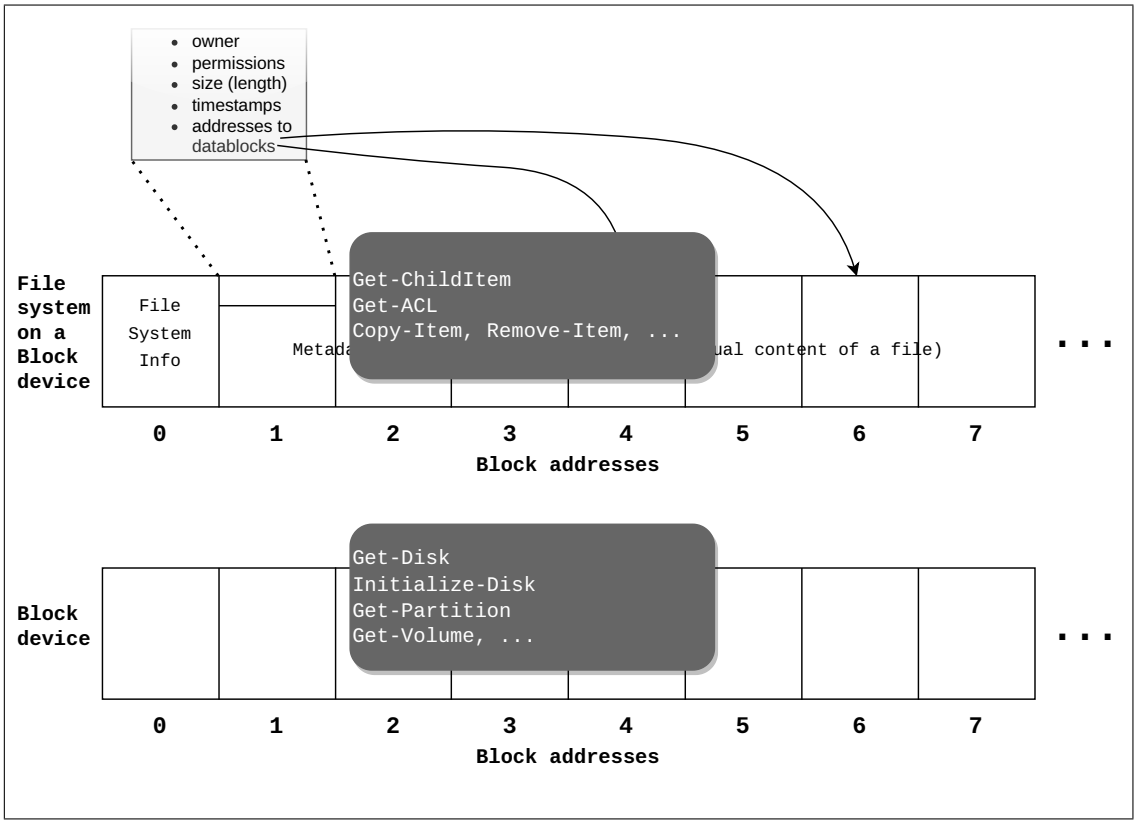


Figure 3.3: PowerShell: File-level vs Block-level.

table).

Volume (`Get-Volume`) A volume is a term that varies in meaning depending on context. Sometimes it is the same as a partition, sometimes it is multiple partitions joined together, and sometimes we distinguish between physical and logical volumes. For now, think of it as either a partition or a group of partitions treated as one unit. *A volume is something we can create a file system on.* (We can also create a file system directly on a partition, but we typically have a volume layer between the file system and the partition.)

File system (`Get-Volume | Format-Table -Property DriveLetter,FileSystemType`) A file system is the data structure we write to a volume (or a partition in some cases) that allows us to store files and directories/folders. On Windows the most common file system is NTFS.

File (`Get-ChildItem`) A file consists of metadata (owner, permissions, timestamps, etc.) and data (the actual contents of the file). A directory is just a special kind of file.

File Security: ACL with ACE (`(Get-Acl file).Access`) All files and folders have and Access Control List (ACL). An ACL is a list of Access Control Entries (ACE). Each

ACE defines a user or group and what permission (read, write, execute, append, etc.) they have. Each entry can be of type allow or deny (deny can be used to exclude a permission that a user otherwise would have). ACLs are scanned by the operating system (Windows) in order and the first ACE that match the access attempted is used.

3.3 Backup: Why?

Chapter 2.9 ”Etabler evne til gjenoppretting av data” of NSM’s ”Grunnprinsipper for IKT-sikkerhet 2.0” [1] is worth reciting in full. We will address most of these measures and submeasures in the rest of this chapter.

Measure 2.9.1:

> Legg en plan for regelmessig sikkerhetskopiering av alle virksomhetsdata. En slik plan bør som minimum beskrive
>
> a) Hvilke data som skal sikkerhetskopieres.
> b) Regelmessighet på sikkerhetskopiering av ulike data, basert på verdi.
> c) Ansvar for sikkerhetskopiering av ulike data.
> d) Prosedyrer ved feilet sikkerhetskopiering.
> e) Oppbevaringsperiode for sikkerhetskopier.
> f) Logiske og fysiske krav til sikring av sikkerhetskopier.
> g) Krav til gjenopprettingstid for virksomhetens ulike systemer og data (se prinsipp 4.1 - Forbered virksomheten på håndtering av hendelser).
> h) Godkjenningsansvarlig(e) for planen.

Measure 2.9.2:

> Inkluder sikkerhetskopier av programvare for å sikre gjenoppretting. Dette inkluderer (som minimum)
>
> a) sikkerhetskonfigurasjon ref. prinsipp 2.3 - Ivareta en sikker konfigurasjon og
> b) maler for virtuelle maskiner og
> c) ”master-images” av operativsystemer og c) installasjonsprogramvare.

Measure 2.9.3:

> Test sikkerhetskopier regelmessig ved å utføre gjenopprettingstest for å verifisere at sikkerhetskopien fungerer.

- Why?
  - Archival purposes
  - Disk failure
  - Theft
  - Accidental deletion
  - Ex-employees (insider threats)
  - Malware: *ransomware*
  - Natural disasters, bomb, el.magn.pulse., etc

Figure 3.4: Backup System = Data Restoration System.

Measure 2.9.4:

> Beskytt sikkerhetskopier mot tilsiktet og utilsiktet sletting, manipulering og avlesning.
>
> a) Sikkerhetskopier bør være separert fra virksomhetens produksjonsmiljø. Se bl.a. prinsipp 2.1 - Ivareta sikkerhet i anskaffelses- og utviklingsprosesser.
>
> b) Tilgangsrettigheter til sikkerhetskopier bør begrenses til kun ansatte og systemprosesser som skal gjenopprette data.
>
> c) Det bør jevnlig tas offline sikkerhetskopier som ikke kan nås via virksomhetens nettverk. Dette for å hindre tilsiktet/utilsiktet sletting eller manipulering.
>
> d) Sikkerhetskopier bør beskyttes med kryptering når de lagres eller flyttes over nettverket. Dette inkluderer ekstern sikkerhetskopiering og skytjenester.

Backup System = Data Restoration System in figure 3.4. A backup system is fundamentally a ”data restoration system” since backups are useless if we cannot restore backed up data. The backup system is an integral part of the overall Disaster Recovery Plan (in the NSM document [1], this is in section four and called ”planverk for hendelseshåndtering som ivaretar behovet for virksomhetskontinuitet ved beredskap og krise”). It is important to think about why we should do backups because it directly influences which strategy we should choose. If the most common reason for restore requests is accidental deletion, then we should put some effort into making sure users have easy access to basic file restore. This can for instance be to have a local backup on each user’s laptop and educate users on how they can restore files from it.

from Æleen Frisch [21]

- *Which files* need to be backed up? (which files or file types should be *ignored*)?
- Where, when and under what conditions should backups be performed?
- How often do these files change? What is *a change*?

NSM measure 2.9.2: Remember *configs, software, OS images, installation software*

Figure 3.5: Data Analysis.

3.3.1 Ransomware

Ransomware is one of the biggest cybersecurity threats. Ransomware is highly relevant for us since it is a threat to business continuity in at least two ways: 1. It disrupts production by encrypting data and 2. It steals confidential data (company secrets) and threatens to disclose it. This means that reliable backups with fast restore procedures are necessary to protect against threat number one, and protection of backups is one of the measures to counter threat number two.

3.4 Backup: What?

Data Analysis in figure 3.5. It is important to carefully estimate the value of the information you have stored in your files. Some files are more important than others, e.g. source code is much more important than a compiled executable file. When you know which files to back up, you should analyze how much space these files occupy and how frequently they change. This analysis builds the ground for designing the backup system. We will not go into the exact computations, value estimations and risk assessments needed to build a complete backup system in this course. We focus on the basic understanding of how backups work and what we need to do to have the necessary basic protection.

3.4.1 Exclude/Ignore files

Not every file should be backed up, so it is important to set up a list of what we want to exclude/ignore from the backup. This list can be files, directories or file/directory names that match certain patterns. A simple algorithm for doing this is:

1. Create a basic list of what is the most obvious to exclude.
2. Do a backup. Look for log messages of errors or warnings. Examine the backup and look for unwanted files.

- Actual content changed? (scanning for this takes time...)
- Only scan content if a "B-MAC" change?
  - *B*irth (`CreationTime`)
  - *M*odified (`LastWriteTime`)
  - *A*ccess (`LastAccessTime`)
  - Change (only change in metadata)

Read it without updating AccessTime?

Figure 3.6: File Change Detection.

3. Modify your list of what to exclude.
4. Repeat until you are happy.

A typical list to start with if you are setting up a full backup of a Windows-host, would look something like this:

```
C:$$Recycle.Bin
C:\pagefile.sys
C:\hiberfil.sys
C:\swapfile.sys
C:\ProgramData\Microsoft\Windows Defender\Definition Updates\Backup\*
C:\ProgramData\Microsoft\Windows Defender\Support\*.log
C:\Windows\Temp\MpCmdRun.log
C:\Windows\SoftwareDistribution\DataStore\Logs\*
C:\Users\*\AppData\Local\Temp\Diagnostics\*
C:\Windows\SoftwareDistribution\Download\*
C:\ProgramData\Microsoft\Network\Downloader*
C:\Windows\system32\LogFiles\WMI\RtBackup\*.*
C:\Windows\memory.dmp
C:\System Volume Information\*
```

3.4.2 File Change Detection

File Change Detection in figure 3.6. Always scanning files for content changes is very expensive for large file collections. Sometimes it is necessary to do it, and to be absolutely certain of detecting any changes it must be done. Most of the time it is sufficient to only scan for changes in Modified time. The backup software *Restic* on platforms other than Windows checks if Modified time, Change time, File Size, and "File-ID" all match the previous version of the file for Restic to consider the file unchanged [7]. For

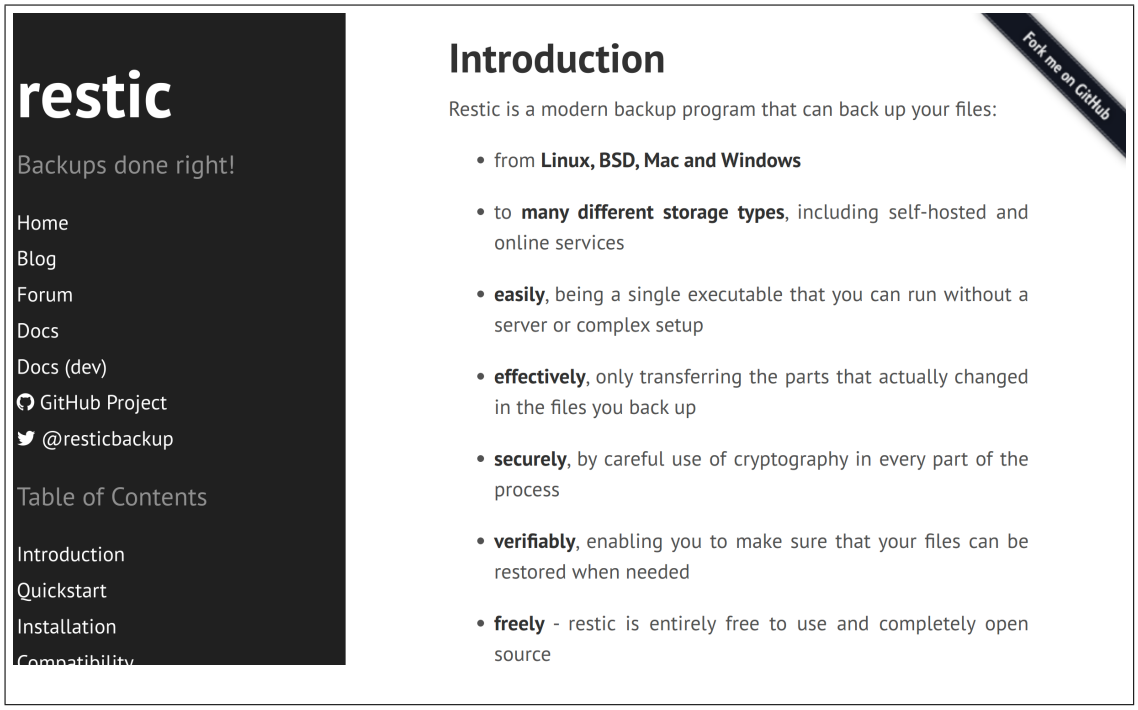


Figure 3.7: Restic.net.

Windows environments, the Restic documentation states: "' On Windows, a file is considered unchanged when its path, size and modification time match . . . "'

3.5 Backup: How?

3.5.1 Tools

Restic.net in figure 3.7. There are many backup tools available. We have chosen the software Restic in the exercises since it is widely used, cross-platform and supports using OpenStack Object Storage as a storage repository.

3.5.2 Architecture

From NRK beta [23]:

> - Jeg tenkte "det her kan sikkert løses med backup", sier Jacobsen i dag. Han ler nervøst. Slik gikk det dessverre ikke. Selskapet hadde en sikkerhetskopi av selskapets viktigste filer, men også den hadde blitt kryptert.

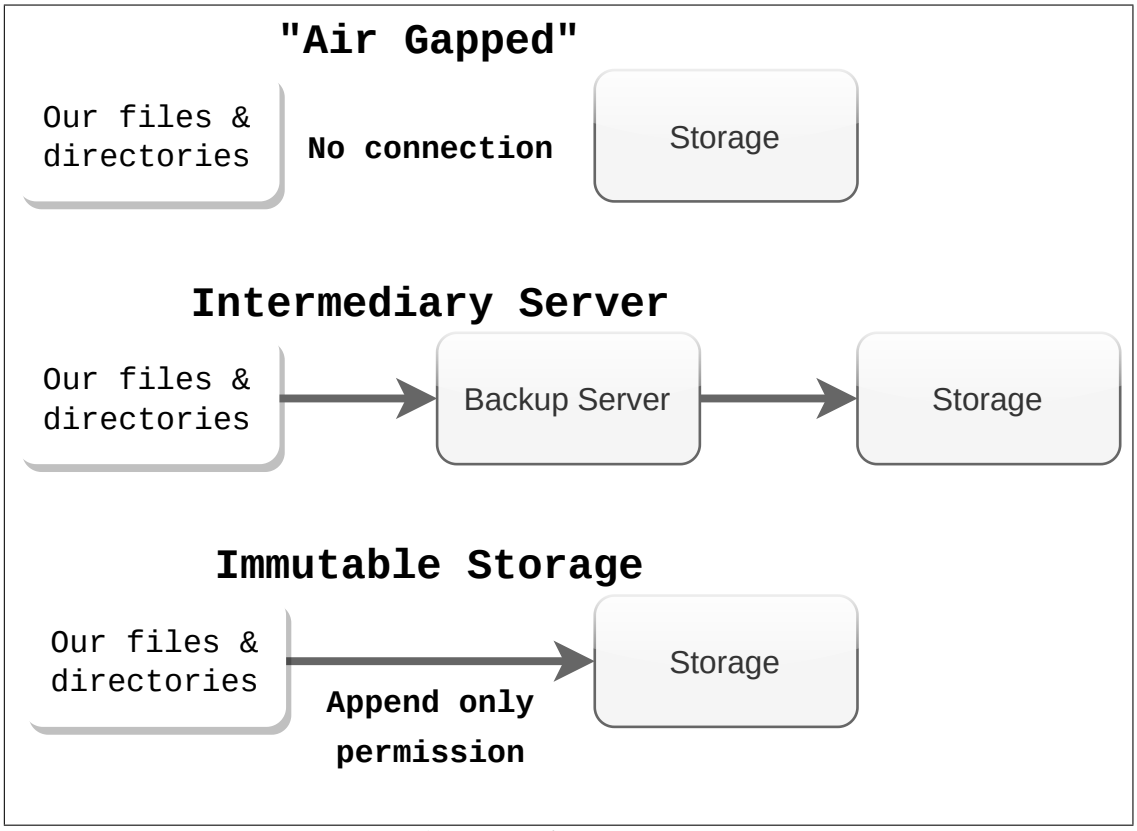


Figure 3.8: Architecture for Ransomware Protection.

Architecture for Ransomware Protection in figure 3.8. A key problem we need to solve when setting up a backup system architecture is that backups should be *immutable*, meaning they cannot be tampered with. If malware (e.g., ransomware) attacks us and runs with the permissions of the backup account, it could overwrite or encrypt our backups. To protect against this threat, we have at least three architectural options:

Air gap means to have an ”air gap” (and no wireless connection either of course) between our data and the backup storage, e.g. when you have your backup on a hard drive that is not connected in any way to your computer.

Intermediary server means to have a dedicated backup server which a local backup process on your computer can write to, but a separate process on the backup server will copy/move, verify and secure your backup as soon as you have written to it. This way the backup server can move a copy out of reach for any malware on your computer.

Immutable storage means for the backup process on your computer to have ”append only” permission to the backup storage. In this way, malware can add encrypted data (or malware) to the backups but not influence previously backed up data.

Full (level 0) Copy all files

Partial Two concepts sometimes referred to as the same:

Differential (level 1) All new or modified files since last full backup

Incremental (level n) All new or modified files since last full *or partial* backup

- Difference between differential and incremental sometimes referred to as *(dump) level* n where $n \in \{0, 1, 2, \ldots\}$

Figure 3.9: Full or Partial.

In the exercises we will set up backup similar to "Immutable storage" when we store backups in the object store of SkyHiGh.

3.5.3 Full or Partial

Full or Partial in figure 3.9. Dump levels refer to how many backups are needed to do a full restore. If the most recent backup performed was a level three ($n = 3$), then you would need to complete the following steps for a full restore:

1. Restore last full backup (level 0)
2. Restore last level 1 incremental backup
3. Restore last level 2 incremental backup
4. Restore last level 3 incremental backup

Doing a full backup takes much more time than doing a partial backup, so "dump level strategy" is based on a trade-off between backup-time and restore-time. These concepts (differential and incremental) are mostly relevant when talking about tape backups [74], since reading from multiple tapes can be quite time-consuming. When backing up to hard drive based storage solutions (including cloud storage) we talk about *snapshots* instead of increments. Snapshots are in principle increments, but they are implemented in a way that is transparent to the user when doing restore. A snapshot is a copy of your data at a certain point in time (although the mechanism behind the scene is similar to an increment).

3.5.4 3-2-1 plan

3-2-1 Backup Plan in figure 3.10. The 3-2-1 backup plan was coined by photographer

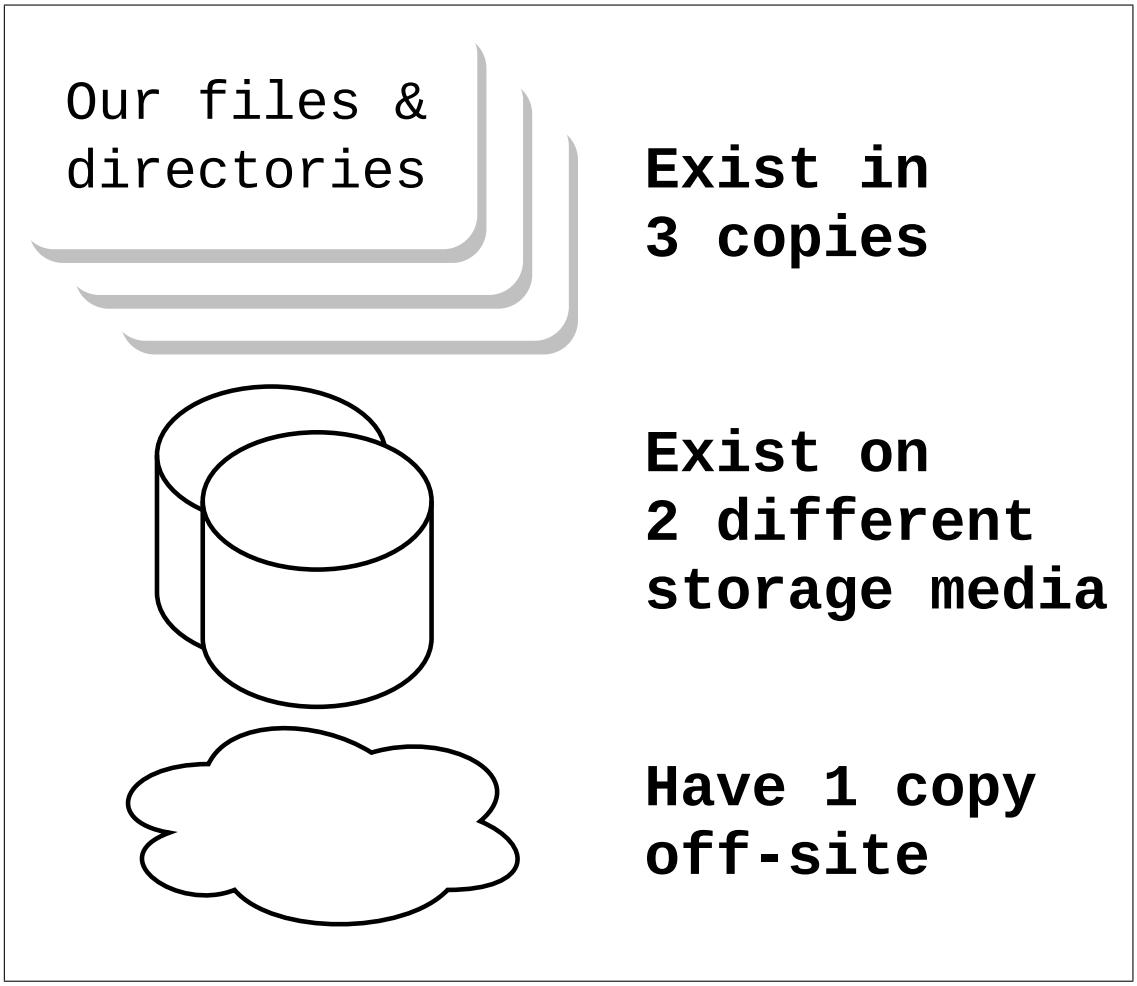


Figure 3.10: 3-2-1 Backup Plan.

Peter Krogh in the book ”The DAM Book: Digital Asset Management for Photographers” [28] and is a general concept for having what should be a minimum of protection for your data. It states that there should be three copies of your data (typically one in production and two backups), two different storage media should be in use (e.g. hard drive and magnetic tape) and one copy should be off-site (in case of fire or similar incidents). We should think of this rule as a minimum, e.g. the author practices a 4-3-2 version with production data in home directory, local backup on same SSD, remote backup on NTNU's file server [57] and NTNU has a backup system for the file server (probably a ”tape robot” of some kind).

3.5.5 Schedule

NTNU's web page on backups state "A backup is taken of every home directory, every night, so that files can be restored if corrupted or deleted" [57]. This is considered the off-site secure backup. In addition, you probably want to protect against accidental deletion and have a local backup that takes a snapshot/increment every half hour (or even more often). In addition, you also do not want your backup storage to grow indefinitely, so periodically (perhaps once per day), you should delete old backups according to a schedule you have set as policy. This process is sometimes called *pruning backups*. A typical policy might be to keep only:

1. all from last 24 hours
2. last from each of last 7 days
3. last from each of last 4 weeks
4. last from each of 12 months
5. last from each of 100 years

This can be enforced in restic with:

```
restic  -r restic-repo forget \
  --password-file /root/.backupcreds \
  --keep-within-hourly 24h \
  --keep-daily 7 \
  --keep-weekly 4 \
  --keep-monthly 24 \
  --keep-yearly 100 \
  --prune
```

3.5.6 Backup user

Measure 2.6.5 "Minimer rettigheter på drifts-kontoer" of NSM's "Grunnprinsipper for IKT-sikkerhet 2.0" [1] is worth reciting in full:

a) Etabler ulike kontoer til de ulike drifts-operasjonene (selv om det kanskje er samme person som reelt sett utfører oppgavene), slik at kompromittering av en konto ikke gir fulle rettigheter til hele systemet. Dvs. forskjellige drifts-kontoer for backup, brukeradministrasjon, klientdrift, serverdrift, mm.

b) Begrens bruken av kontoer med domeneadmin rettigheter til kun et minimum av virksomhetens drifts-operasjoner. Spesielt bør kontoer med domene-admin rettigheter aldri benyttes interaktivt på klienter og servere (reduserer konsekvensene av "pass the hash" angrep).

- Dedicated account
  - *Minimize risk* if compromised
  - *Traceability*
- Minimum privileges
  - Must be able to *read all files*
  - Must be able to *read "files in use"*

Figure 3.11: Backup Account/User.

c) Unngå bruk av upersonlige kontoer ("backup_arne" er bedre enn bare "backup") slik at man har god sporbarhet og lettere kan deaktivere kontoer når noen slutter. Hvis det er vanskelig å unngå å ha en upersonlig konto bør man først logge seg inn med en personlig bruker for å ivareta sporbarhet.

This means we should use a dedicated account to run the backup service and this account should have a minimum of privileges.

Backup Account/User in figure 3.11. On Linux this is most easily done by creating an account with the password field set to an "invalid character" like `!` or `*`) and the account's login shell set to `/usr/sbin/nologin`. In addition, the backup binary executable that will run in the security context of this account can get a special privilege that allows it to read all files in the file system. This can be done for e.g. the Restic-binary with something like
`setcap cap_dac_read_search=+ep /usr/bin/restic`
(a problem still exists with this approach in that this `cap_dac_read` capability must be set any time the Restic-binary is updated).

On Windows this is unfortunately a bit harder. The parallel to Linux would be to create a user that is disabled and without a password:
`New-LocalUser -Name BackupMysil -Disabled -NoPassword`

seBackupPrivilege in figure 3.12. The user `BackupMysil` should then be given a special *privilege* called *seBackupPrivilege* to be able to read all files on the file system. On Windows there exists by default a group called "Backup Operators":

```
PS> Get-LocalGroup -Name 'Backup Operators' | Select-Object -Property *

Description     : Backup Operators can override security restrictions for
                  the sole purpose of backing up or restoring files
Name            : Backup Operators
SID             : S-1-5-32-551
```

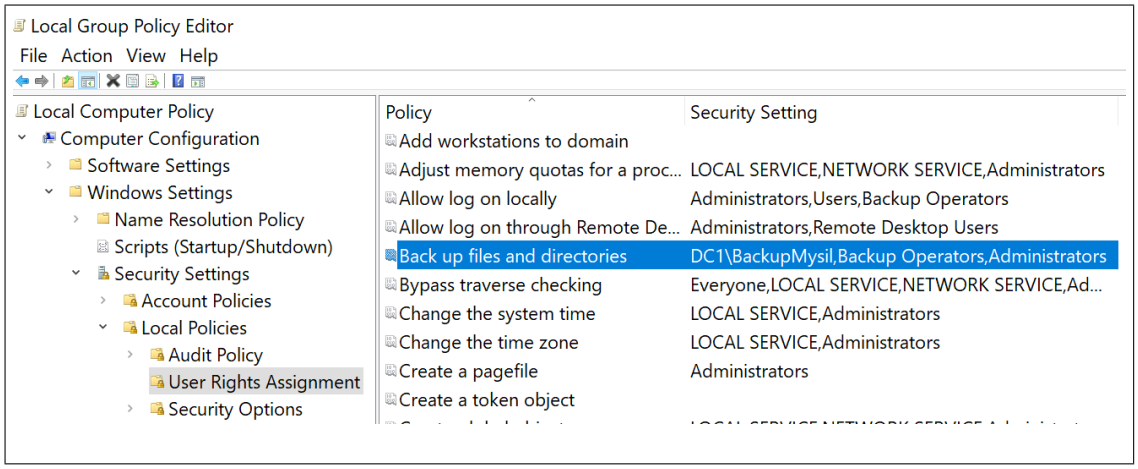


Figure 3.12: seBackupPrivilege.

```
PrincipalSource : Local
ObjectClass     : Group
```

This group also has the privilege *seRestorePrivilege* which basically is the right to write anywhere in the file system, and we don't want that. If an attacker is able to compromise a process running under an account with the *seRestorePrivilege*, they will be able to own the system (take control of everything, become Administrator).

Adding the seBackupPrivilege to an account is a bit complicated to do using the command line, but easy to do with the Local Group Policy Editor by choosing in sequence: "Computer Configuration", "Windows Settings", "Security Settings", "Local Policies", "User Rights Assignment", right click "Back up files and directories" and add the BackupMysil user. We can verify which groups and users have which privileges with the Sysinternals tool `AccessChk` (notice how BackupMysil only has seBackupPrivilege while the Backup Operators group also have the seRestorePrivilege):

```
PS> accesschk.exe -a seBackupPrivilege

          BUILTIN\Backup Operators
          BUILTIN\Administrators
          DC1\BackupMysil

PS> accesschk.exe -a seRestorePrivilege

          BUILTIN\Backup Operators
          BUILTIN\Administrators

# can also verify with
secedit /export /areas USER_RIGHTS /cfg OUTFILE.CFG
```

```
# and a logged in user can check its own privileges with
whoami /PRIV
```

On Windows there also exists a special service that can be used for reading "files in use". This service is called Volume Shadow Service (VSS) [50] and from the documentation [45] it should be possible to let an account use it by adding the account name to the registry with a value of `1`:

```
PS> Get-Item HKLM:\SYSTEM\CurrentControlSet\Services\VSS\VssAccessControl\

Name                           Property
----                           --------
VssAccessControl               NT Authority\NetworkService : 1
                               DC1\BackupMysil             : 1
```

Turning theory into practice is sometimes challenging. Unfortunately, getting this setup to work has proven difficult[1]. As you will see in the exercises, we run the backup process under the `SYSTEM` account (probably displayed as `NT Authority\SYSTEM`) which has a full set of permissions to all files [41] in the file system. It is unfortunately quite common that trying to achieve least privilege is too hard, and we end up using a high privilege account because it's the only "practical" solution. The Windows Backup Agent in the widely used Veeam Backup software also runs under the `SYSTEM` account [20].

> *You will encounter this problem many times throughout your career: implementing security principles (such as a least privilege backup account) is challenging, and sometimes it may not be feasible. In those cases, you will have to accept a less secure solution. However, you should always strive to achieve the most secure approach. When you must compromise on security, compensate with additional measures: for example, segment your network, restrict access at the network level, and implement extended monitoring (both on individual hosts and across network traffic).*

3.5.7 Important features

Encryption

Some of the data we need to back up is confidential, and we must protect it both during transit and at rest (in storage). The best approach is to encrypt the data as early as

[1] Perhaps you can get it to work? See the exercises. You might try the `Network Service` account with added seBackupPrivilege or another `-LogonType`. In a Windows Domain, Daniel Hinderaker has demonstrated that we can use a gMSA account, which suggests that on a single host we might use a virtual account. But how do we assign privilege to a virtual account?

- Lossless compression (not lossy like jpeg-images)
- *Compress these 32 bits (4 Bytes) into 1 Byte?*
  `10111011 10111011 10111011 10111011`
- We see the pattern `1011` repeated 8 times. The decimal number 8 is in binary `1000`, so this information content can with Run-length encoding be represented as 8 times `1011`, or `1000` times `1011` or simply:
  `10001011`

Figure 3.13: Compression.

possible—ideally on the client side before transmitting it over the network (although network connections are typically encrypted as well).

If you are unsure whether the backup system encrypts your data on the client side before transmission over the network, you can always encrypt it manually (as shown in Chapter 1.3.2). However, professional backup software like Restic handles this automatically, and Restic does encrypt data client-side by default.

A problem we frequently encounter when automating tasks on a computer is *authentication*, which typically involves passwords. If we want 100% automation, there is no way to avoid storing the password in a file somewhere. The best we can do is *use a strong, unique password and ensure the file is only accessible by the account that needs it.* In Restic, this file can be supplied during the backup process by adding the following parameter: `--password-file /root/.backupcreds`

Compression

Compression in figure 3.13. This is a simple example of compression using a basic algorithm called run-length encoding. It demonstrates how data can be compressed more efficiently than standard file storage. All files (unless already compressed) contain repeating patterns that can be encoded more efficiently. Backup systems always use compression to save storage space, and this compression works because your data contains redundancy – repeating patterns that can be reduced.

Deduplication

Deduplication in figure 3.14. Deduplication is similar to compression but operates at a much higher level. It identifies high-level data structures – typically entire data blocks (or in simple implementations, entire files) – and stores only one copy of each unique structure. While deduplication can occur at the file level using tools like rsync [16], professional backup software like Restic performs deduplication at the block level, enabling more granular deduplication.

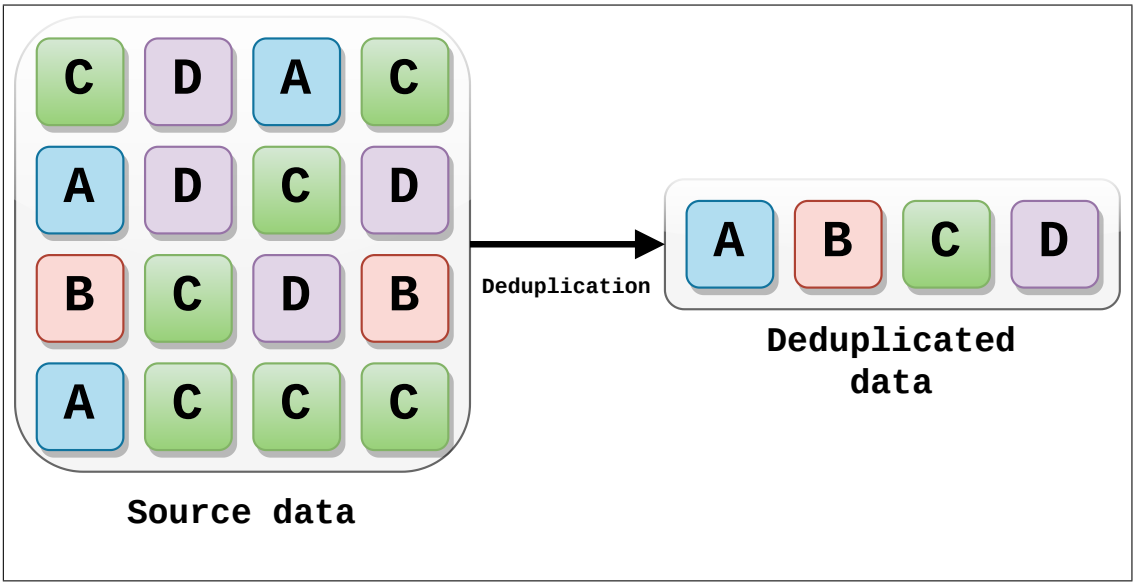


Figure 3.14: Deduplication.

- Structural consistency and integrity
  - `restic -r restic-repo check`
- Integrity of the actual data
  - `restic -r restic-repo check --read-data`

Figure 3.15: Verifying.

Consider the difference: in file-level deduplication, A, B, C, and D represent files, whereas in block-level deduplication, they represent blocks, parts of blocks, or groups of blocks. Block-level deduplication is always more efficient because the system can deduplicate across the entire storage device. File-level deduplication (such as with rsync) is less efficient: a small change to a large file requires storing the entire file again. For example, if two files differ by only 1%, file-level deduplication would store both files in full, even though 99% of their data blocks are identical.

3.6 Secure Storage and Restore

3.6.1 Verifying

Verifying in figure 3.15. A backup copy is not very useful if we cannot trust it. If the actual storage device where the backup is located has been damaged (e.g., a faulty RAID volume), we can detect this by checking *structural consistency*, which means verifying

- Restore everything?
  - `restic -r restic-repo restore latest \`
    `--target /tmp/restore-work`
- Browse and select files to restore?
  - `restic -r restic-repo find a.txt`
  - `restic -r restic-repo restore latest`
    `--target /tmp/restore-work`
    `--include a.txt`

Figure 3.16: Restore.

all the files used by the backup system itself (index files and other data structures). In the worst case, an attacker could gain access to the backup storage and write malware into the actual backup files, hoping that we will restore them and inadvertently spread the malware. To protect against this threat, we must actually verify the contents of the files by checking whether they have been altered and whether they still match hash sums stored in the metadata (though keep in mind that this metadata could also be compromised in an attack). This verification can also include scanning the backed-up files for malware. For example, when recovering from a ransomware attack, how can you be certain the ransomware is not present in your backups? Which snapshots can you trust? How far back in time must you go? These are challenging questions. The recommended approach is to build a timeline using all available logs from your computers and any historical data from your monitoring systems. This allows you to identify the point in time when the initial breach most likely occurred.

3.6.2 Restore

Restore in figure 3.16. A full restore can be very time-consuming and resource-intensive. Make sure you choose a backup system that provides flexible restore options. Keep in mind that the most common restore scenario is when a user accidentally deletes a file, so design your backup system with this use case in mind.

To improve user experience and reduce administrative overhead, consider these approaches:

- Allow users to access a local backup where they can search for and restore deleted files themselves
- Make it easy for users to retrieve individual files without administrator involvement
- Enable administrators to perform complete system restores when needed

3.6.3 Logs and Monitoring

As mentioned, a backup system is not valuable if we cannot perform a restore, so we must ensure that backups are made successfully. Several issues can arise during the backup process. For instance, we might lose the network connection, the destination storage could be full or faulty, or the backup system might refuse to proceed due to a previous backup attempt resulting in a *stale lock file*. Lock files are commonly used to prevent multiple processes from writing to the same files simultaneously. If a backup is interrupted – perhaps due to a lost network connection or a power outage – a lock file may remain, preventing the next backup attempt. This type of lock file is referred to as a stale lock file, as it indicates a failed backup attempt.

Your logs and the corresponding monitoring system should check for exit codes from the backup system to ensure that backups are performed correctly. For example, when we run the command `restic backup --help`, it provides information about restic's exit codes:

- An exit status of 0 indicates that the command was successful.
- An exit status of 1 signifies a fatal error, meaning no snapshot was created.
- An exit status of 3 indicates that some source data could not be read, resulting in an incomplete snapshot.

3.7 About the Exercises

In this week's exercises, you will set up a real backup system. The exercise is designed to provide you with several additional learning benefits:

- Learn how to use environment variables for automation.
- Understand how to execute tasks periodically.
- Discover how to automate tasks that require authentication (passwords).
- Explore how to use cloud object storage (similar to Amazon S3 and Azure Blob Storage).
- Learn about application credentials (we should always strive for "least privilege" and use a separate account for each task whenever possible).

3.8 Lab tutorials

1. **Block-level (OpenStack Cinder): Attach a block device on D:**. This exercise is not mandatory, but if you are unsure what a block device is and how disks and file systems show up in Windows, then do it.

   (a) Login to SkyHiGh, click ”Volumes”, ”Volumes”, ”Create Volume”, give it any name you want and leave it to 1GB in size.

   (b) To the right of your newly created volume click ”Edit Volume”, ”Manage Attachments”, in ”Attach to Instance” choose the server you would like to attach the volume to.

   (c) On the server do the following to make this volume accessible on `D:`:

   ```
   # Find disk number (in PowerShell):
   Get-Disk

   # The following commands needs "Run as Administrator"
   # Initialize
   Initialize-Disk DISK_NUMBER

   # Partition and format
   New-Partition -DiskNumber DISK_NUMBER -UseMaximumSize `
     -AssignDriveLetter | Format-Volume

   # View
   Get-PSDrive
   ```

   (d) If you want to ”safely remove” the attached volume, do

   ```
   # Unmount (if Drive has letter 'D')
   Get-Volume -Drive D | Get-Partition |
     Remove-PartitionAccessPath -AccessPath D:\
   ```

2. **File-level (OpenStack Swift): Backup to object storage**. Let’s try to set up the backup software Restic on a Windows Server and use OpenStack’s object storage (the Swift component of OpenStack) as backend (storage repository for the backups). Ideally we should be able to set this up in such a way that the backups are *immutable* (unchangeable), but it does not appear that OpenStack Swift supports this yet, so we will only use it as a regular writable backup storage which would be vulnerable for malware but useful for us if we delete files by accident.

   (a) If you don’t already have a Windows Server to use, create a new stack and log in to the server (see lab tutorial in week one).

(b) Create the object store container where the backups will be stored: Login to SkyHiGh, click "Object Store", "Containers", "+Container", name it `mysil` (you can name it whatever you want, but if you name it mysil you can copy and paste more of the text later).

(c) Many times when we need to automate tasks, we need to have authentication, which typically means putting a password into a script. Password in scripts or configuration files is not something we want, but we cannot always avoid this, so the best we can do is use a separate account with minimum privileges. For the current task, let's create a separate account which is valid only for a short time period (e.g. just for the current month). In SkyHiGh, click "Identity", "Application Credentials", "Create Application Credential":

- Choose a name, e.g. "backupcreds"
- Set expiration date to last day of the current month.
- Click on role "_member_".
- Click "Create Application Credentials"
- Click "Download openrc file". In this file you will find the ID and the Secret (the "password").

(d) Inspect the Restic package[2] (does it look safe to install?), Open PowerShell as Administrator, install Restic with
`choco install -y restic`.

(e) Open PowerShell (not as Administrator), set the necessary environment variables (replace `YOUR_ID` and `YOUR_SECRET`):

```
$env:OS_AUTH_TYPE='v3applicationcredential'
$env:OS_AUTH_URL='https://api.skyhigh.iik.ntnu.no:5000/v3'
$env:OS_IDENTITY_API_VERSION='3'
$env:OS_REGION_NAME="SkyHiGh"
$env:OS_INTERFACE='public'
$env:OS_APPLICATION_CREDENTIAL_ID='YOUR_ID'
$env:OS_APPLICATION_CREDENTIAL_SECRET='YOUR_SECRET'
```

(f) Initialize your backup repository (you will now need to provide a password of your choice, which is used for encrypting your backups to protect them) with
`restic -r swift:mysil:/ init`

(g) Confirm that your backup repository is empty (it does not have any snapshots) with
`restic -r swift:mysil:/ snapshots`

[2] community.chocolatey.org/packages/restic

(h) Open PowerShell as Administrator (so we can use the Volume Shadow Service (VSS) to also backup files that are in use), create a backup of your home directory with
`restic -r swift:mysil:/ backup $HOME --use-fs-snapshot`
(remember that you have to set our environment variables whenever you open a new PowerShell session)

(i) Confirm that your backup repository now have one snapshot with
`restic -r swift:mysil:/ snapshots`

3.9 Review questions and problems

1. What is the difference between file-level storage and block-level storage, and why is this distinction important in cloud and virtualized environments?
2. What are the two main components of a file, and what type of information is stored in file metadata?
3. How do the Windows storage concepts disk, partition table, partition, volume, and file system relate to each other?
4. According to NSM measure 2.9.1, what elements should be included in a backup plan?
5. What are common reasons for performing backups, and how do these reasons influence backup strategy?
6. Why is it important to exclude certain files from backups, and what is the recommended process for creating an exclude list?
7. What challenges are associated with file change detection, and which attributes are commonly used to detect changes?
8. What does it mean for backups to be *immutable*, and what architectural approaches can be used to achieve this?
9. What is the difference between full, differential, and incremental backups, and how does this affect restore operations?
10. What is the 3-2-1 backup rule, and what minimum level of protection does it provide?
11. Why are logging and monitoring essential components of a backup system?
12. Consider the setup we have done with Restic in the lab tutorial. This does not follow the ideal setup with respect to user accounts and permissions/privileges. In what way is our setup bad? In what way is our setup good?
13. Write a command line that will show all volumes that have more than 5GB size remaining.
14. Write a command line that will output the following (note: sorted by size)

```
DiskNumber PartitionNumber        Size
---------- ---------------        ----
         1                3       66048
         1                1    16693760
         2                1    16759808
```

```
2                2  4294967296
2                3  6424625152
1                2 10719592448
0                1 32210157568
```

15. Before initiating a backup scheme, we need to know a bit about the data we would like to back up. Create PowerShell command lines to answer the following questions:

    (a) How much space does your home directory use including all files and directories?

    (b) Which file is the largest? Which is the smallest?

    (c) How many files are there? What is the average file size?

    (d) How many files changed and how much space did those files use in total, during the last hour? Last day? Last week? (hint: see examples in the Sort-Object section of the PowerShell tutorial[3])

    (e) (Advanced) Which one of the last seven days had the largest change? ("largest change" meaning total space used by all files that was changed during the 24 hours of that day)

    (tip: remember that PowerShell is cross-platform, so maybe try this out on your own laptop to have a more realistic setting than what we have in our virtual machines.)

16. **Automate backup**. Do lab tutorial 2 "File-level (OpenStack Swift): Backup to object storage". The goal of this exercise is to wrap Restic in a PowerShell script and let it run every half hour by using Task Scheduler.

    (a) Create a new folder `scripts` in your home directory and in this folder create

    - a file `excludes.txt` with all the file patterns you want to exclude from your backup (see example earlier in this chapter)
    - a file `mybackup.ps1` where you set all environment variables and run the backup command from the lab tutorial
      (hint: use `$env:RESTIC_PASSWORD` to set the repository password)

    (b) At the end of your script, add commands for deleting the environment variables containing secrets (think "just in time", these secrets are only needed when restic is executed).

    (c) Schedule the script to run every half hour with the following commands (must be in a PowerShell as Administrator):

[3] gitlab.com/erikhje/dcsg1005/-/blob/master/powershell.md#sort-object

```
$taskName = "MyBackup"
$taskDescription = "Restic backup of Users data"

$taskAction = New-ScheduledTaskAction `
  -Execute 'C:\Program Files\PowerShell\7\pwsh.exe' `
  -Argument '-File C:\Users\Admin\scripts\mybackup.ps1'

$taskTrigger = New-ScheduledTaskTrigger `
  -Once -At (Get-Date) `
  -RepetitionInterval (New-TimeSpan -Minutes 30) `
  -RepetitionDuration (New-TimeSpan -Days (7))

# Note: Really should use another account, but
# too hard to get it to work...
$taskPrincipal = New-ScheduledTaskPrincipal `
  -UserId "NT AUTHORITY\SYSTEM" `
  -LogonType ServiceAccount

Register-ScheduledTask `
  -TaskName $taskName `
  -Description $taskDescription `
  -Action $taskAction `
  -Trigger $taskTrigger `
  -Principal $taskPrincipal
```

(d) Run your registered task manually once using Start-ScheduledTask[4].

(e) Study the Restic manual[5] and run a command to show the difference between two of your snapshots.

(f) Use Get-ScheduledTaskInfo[6] to print only TaskName, LastTaskResult and NextRunTime for your registered task.

(g) Use Get-ScheduledTask[7] to print only the path to pwsh.exe for your registered task (hint: this is the property Execute ”under” the Actions-property, use Get-Member to see that the Actions-property is not just a string).

(h) Open PowerShell as Administrator, create the directory `C:\restoretest` and do a restore with restic[8] of your latest snapshot to this directory.

[4] docs.microsoft.com/en-us/powershell/module/scheduledtasks/start-scheduledtask
[5] restic.readthedocs.io/en/latest/040_backup.html#comparing-snapshots
[6] docs.microsoft.com/en-us/powershell/module/scheduledtasks/get-scheduledtaskinfo
[7] docs.microsoft.com/en-us/powershell/module/scheduledtasks/get-scheduledtask
[8] restic.readthedocs.io/en/latest/050_restore.html

4

Git, Markdown and CI/CD

4.1 TL;DR

TL;DR in figure 4.1. Note: from the cheat sheet (and the video), you need to learn properly:

- 01 Git configuration

- Watch the 20-minute Git-Lecture from IfI, UiO at
  youtu.be/vWdmXunGQC8
  (from the course IN3110[1])
- Study the Git Cheat Sheet at
  about.gitlab.com/images/press/git-cheat-sheet.pdf
- Study the Markdown Cheat Sheet at
  Markdown Cheat Sheet
  and browse the GitLab Flavored Markdown at
  docs.gitlab.com/ee/user/markdown.html
- Figure out what `.gitlab-ci.yml` at
  gitlab.com/erikhje/heat-mono/-/blob/master/.gitlab-ci.yml
  is used for

Figure 4.1: TL;DR.

- 02 Starting A Project
- 03 Day-To-Day Work (not `git stash`)
- 05 Review your work (only `git log`)
- 08 Synchronizing repositories (not `git fetch`)
- D The zoo of working areas (don't worry about stash or branches)
- theory of file stages
- and later how to deal with conflicts

4.2 Version Control with Git

It's not a joke in figure 4.2. Everyone in IT uses git, but very few truly understand its internal details. In practice, most users memorize a small set of commonly used commands and look up the rest when needed. This is usually sufficient for day-to-day work, but a deeper understanding becomes important when collaborating in larger teams or when something goes wrong.

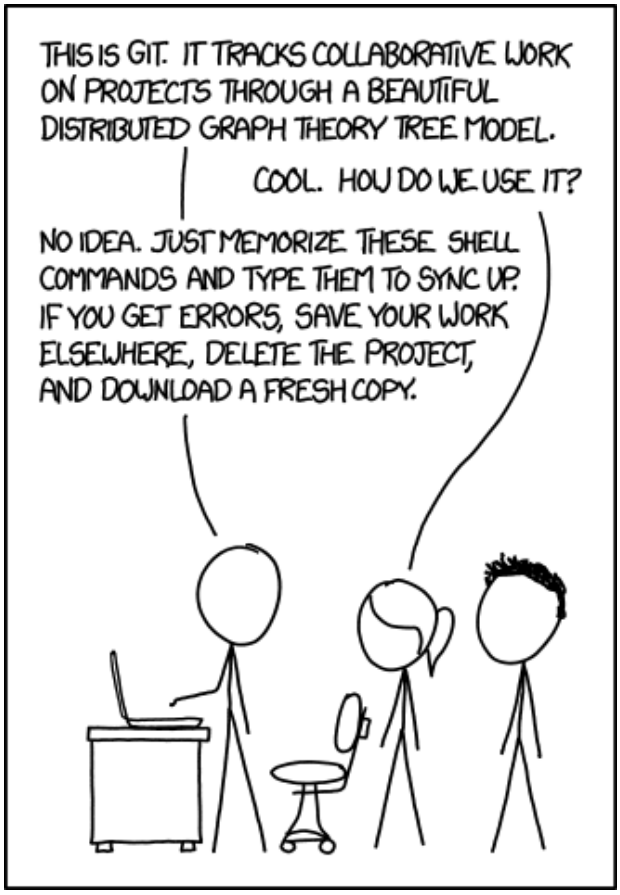
Git was created in 2005 by Linus Torvalds, together with other Linux kernel developers, with the goal of providing a better version control system for the source code of the Linux operating system kernel [10]. At the time, existing tools did not scale well enough for a large, distributed group of developers working concurrently on the same code base. Git was therefore designed to be fast, reliable, and well suited for distributed development.

Git is today the leading system for *version control*. When we use git, we work with files stored in a directory, typically referred to as a *repository*. These files can be text documents, configuration files, or program source code – although git is primarily optimized for *text-based files* rather than binary files such as `pdf`, `exe`, or `jpg`. While binary files can be stored in git, changes to such files are harder to track and compare.

During development, we modify files and save them locally. When we reach a meaningful state – such as completing a small feature, fixing a bug, or making a logical improvement – we *commit* the changes. A commit represents a recorded snapshot of the repository at a specific point in time. Each commit is permanently stored in the repository history and can be revisited later, allowing us to inspect past versions or revert changes if necessary.

Using version control systems such as git provides several important advantages, especially when working in teams:

- Each commit records who changed which files and when, providing clear *accountability* and traceability.



https://xkcd.com/1597
https://creativecommons.org/licenses/by-nc/2.5

Figure 4.2: It's not a joke.

```
erikhje@spock:/tmp/test$ tree -a
.
└── .git
    ├── branches
    ├── config
    ├── description
    ├── HEAD
    ├── hooks
    │   ├── applypatch-msg.sample
    │   ├── commit-msg.sample
    │   ├── fsmonitor-watchman.sample
    │   ├── post-update.sample
    │   ├── pre-applypatch.sample
    │   ├── pre-commit.sample
    │   ├── pre-merge-commit.sample
    │   ├── prepare-commit-msg.sample
    │   ├── pre-push.sample
    │   ├── pre-rebase.sample
    │   ├── pre-receive.sample
    │   └── update.sample
    ├── info
    │   └── exclude
    ├── objects
    │   ├── info
    │   └── pack
    └── refs
        ├── heads
        └── tags
```

Figure 4.3: An Empty but Initialized Repo.

- Developers can *push* their local commits to a shared repository and *pull* changes made by others, keeping multiple copies of the repository synchronized.
- Git can automatically merge changes when multiple people edit different parts of the same files, reducing manual coordination.
- Git can easily visualize what has changed between commits, making it straightforward to review modifications and understand their impact.
- In other words, git serves as a central component that enables efficient collaboration, coordination, and quality control within a development team.

4.2.1 Repository

An Empty but Initialized Repo in figure 4.3. Git can be understood as consisting of two

closely related parts:

1. The Git repository (often abbreviated as *repo*)
2. The Git program

The Git repository is, at its core, just a directory on the file system. What makes it a Git repository is the presence of one special hidden directory called `.git`. This directory contains all of Git's internal data structures, including the complete history of the project, metadata about commits, branches, and tags, as well as information needed to manage collaboration with other repositories.

The contents of the `.git` directory are managed entirely by the Git program. As users, we normally never edit these files manually. Instead, we interact with the repository exclusively through Git commands, such as `git commit`, `git status`, and `git pull`. These commands read from and write to the `.git` directory in a controlled and consistent way, ensuring the integrity of the repository history.

It is important to understand that the files we actively work on – the source code and other project files – are located *outside* the `.git` directory. Git continuously compares the current state of these working files with the information stored inside `.git` in order to track changes, create commits, and manage different versions of the project.

4.2.2 File States

Four States of a File in figure 4.4. It is important to note that a file can exist inside a Git repository without Git being aware of it. Such a file is called an *Untracked* file. Untracked files are present in the directory, but Git does not include them in version control until explicitly told to do so.

Files that have been added to Git are referred to as *tracked* files. Tracked files can be in several different states. They can be *Unmodified*, meaning they are identical to the version stored in the most recent commit, such as `README.md` and `test.ps1` in the figure. They can also be *Modified but not staged*, such as `b.txt`, meaning the file has been changed in the working directory but is not yet prepared for inclusion in the next commit. Finally, files can be newly added or modified and then *Staged*, indicating that they are ready to be included in the next commit.

A file that is *Modified but not staged* will *not* be part of the next commit if you run `git commit -m "msg"`. However, such files *will* be automatically staged and committed if you use `git commit -am "msg"`. According to the manual page for `git-commit`, the option `-a` (or `--all`) behaves as follows:

> Tell the command to automatically stage files that have been modified and deleted, but new files you have not told Git about are not affected.

```
PS C:\Users\Admin\mysil> ls

    Directory: C:\Users\Admin\mysil

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---           1/3/2022  8:38 PM              6 a.txt
-a---           1/3/2022  8:38 PM              6 b.txt
-a---           1/3/2022  8:35 PM              3 c.txt
-a---           1/3/2022  8:33 PM              3 d.txt
-a---           1/3/2022  7:54 PM             25 README.md
-a---           1/3/2022  8:00 PM             14 test.ps1

PS C:\Users\Admin\mysil> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   a.txt
        new file:   c.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   b.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        d.txt
```

Unmodified

Staged

Modified

Untracked

Figure 4.4: Four States of a File.

1. `git pull origin main`
2. edit one or more files
3. `git commit -am "docs: spellcheck Foo"` *conventionalcommits.org!*
4. `git push origin main`

Figure 4.5: Typical Workflow.

This distinction is important: the `-a` option only applies to files that are already tracked by Git. Newly created files must first be explicitly added to the repository using `git add` before they can be staged and committed.

All files that are in the *Staged* state will be included in the next commit, which represents the next version of the repository.

4.2.3 Workflow

Typical Workflow in figure 4.5. When working on a shared project, a typical workflow consists of pulling the latest version of the repository, making small and focused

- *Avoid them when possible by working on separate files or clearly separated parts of a file.*
- Browse "Merge conflicts" at
  docs.gitlab.com/ee/user/project/merge_requests/conflicts.html
  if Git does not resolve them automatically

Figure 4.6: Conflicts.

changes, committing those changes, and then pushing them back to the shared repository. Keeping changes small and commits focused makes it easier for others to review your work and reduces the risk of merge conflicts.

It is important to understand that there is nothing technically special about the repository hosted on platforms such as GitHub or GitLab. These services do not introduce a new type of repository; they simply host ordinary Git repositories, identical in structure to the one you have locally. Git is a fully decentralized version control system, meaning that every clone contains the complete history of the project. The reason we treat one repository as *the* shared repository is purely a matter of convention and convenience: it provides a common place where team members can push their work and pull changes made by others.

You should always work with the assumption that other people will, at some point in time, read, maintain, or extend your code. For that reason, strive to write readable, high-quality code and clear documentation. Equally important are your commit messages: they form part of the project's long-term documentation and should clearly explain *what* was changed and *why*. A useful guideline for writing consistent and informative commit messages is the *Conventional Commits* specification. You are encouraged to browse https://www.conventionalcommits.org and follow this standard when writing commit messages, especially in collaborative projects.

4.2.4 Conflicts

Conflicts in figure 4.6. A *merge conflict* occurs when two or more people make changes to the same file and Git is unable to automatically determine how those changes should be combined. This situation typically arises when multiple developers work in parallel and then attempt to merge or push their changes to a shared repository. If the changes affect different parts of the file, Git is usually able to merge them automatically without user intervention. However, if the changes overlap – for example, if two people edit the same lines in a file – Git cannot safely decide which version is correct. In such cases, the merge operation stops and the conflict must be resolved manually.

When a conflict occurs, Git marks the conflicting sections directly in the file using special markers. The developer must then inspect these sections, decide which changes to keep (or how to combine them), remove the conflict markers, and create a new commit

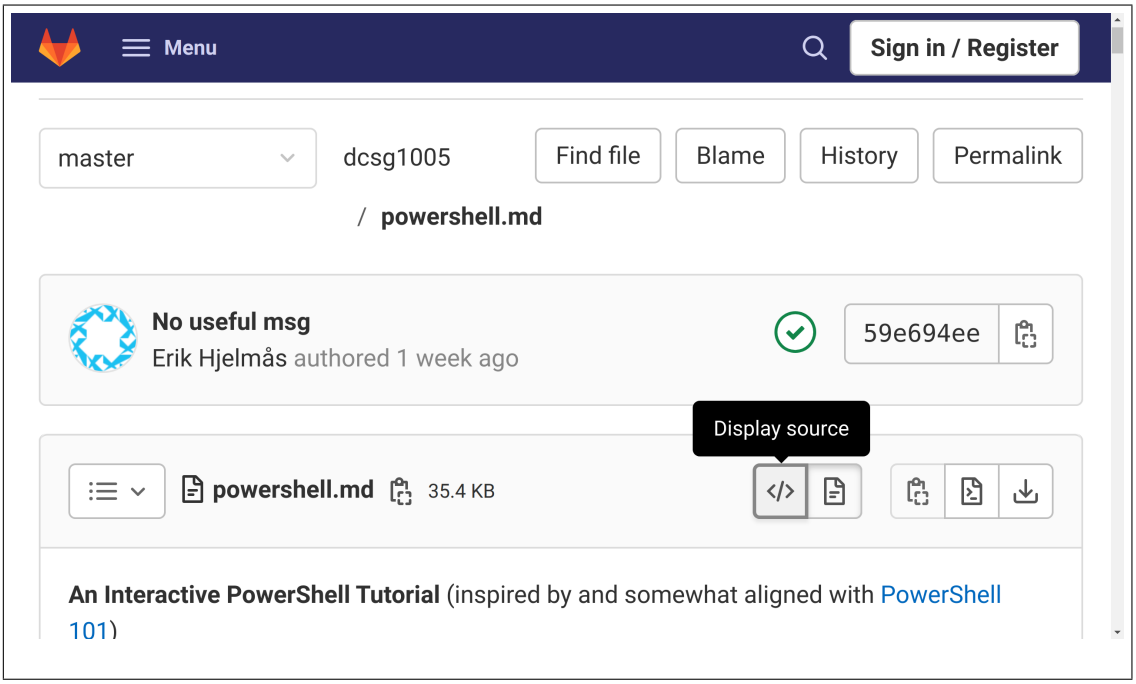


Figure 4.7: The Magic of "View Source".

that resolves the conflict. Although this process may seem intimidating at first, resolving conflicts is a normal and unavoidable part of collaborative software development.

The best strategy for handling conflicts is prevention. Working on separate files when possible, making small and focused commits, and frequently pulling changes from the shared repository all reduce the likelihood of conflicts. Nevertheless, understanding how to resolve them is an essential skill when working with Git in real-world projects.

4.3 Markdown

The Magic of "View Source" in figure 4.7. Markdown is something you can learn very quickly – often in as little as ten minutes – by reading the Markdown Cheat Sheet[2]. In practice, the "Basic Syntax" section will take you a long way. When you need more advanced features, it is usually sufficient to look up GitLab Flavored Markdown[3] or the corresponding documentation for the platform you are using. Beyond that, a very effective way to learn is simply to study the Markdown source of documentation written by others.

Markdown is deliberately much simpler than traditional markup languages such as HTML or LaTeX. This simplicity is one of its main strengths: it allows you to focus on

[2] www.markdownguide.org/cheat-sheet
[3] docs.gitlab.com/ee/user/markdown.html

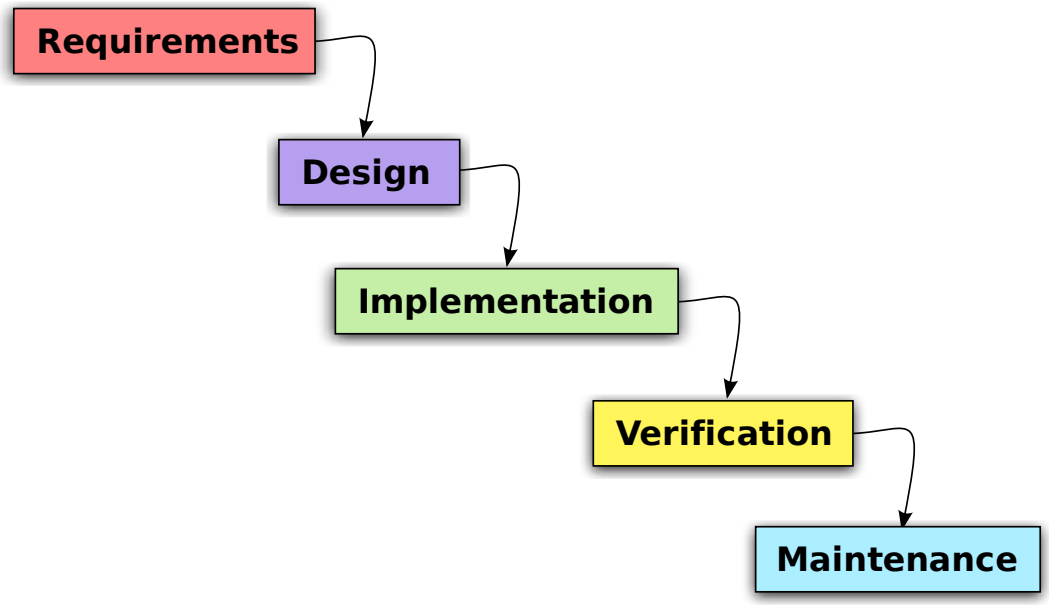


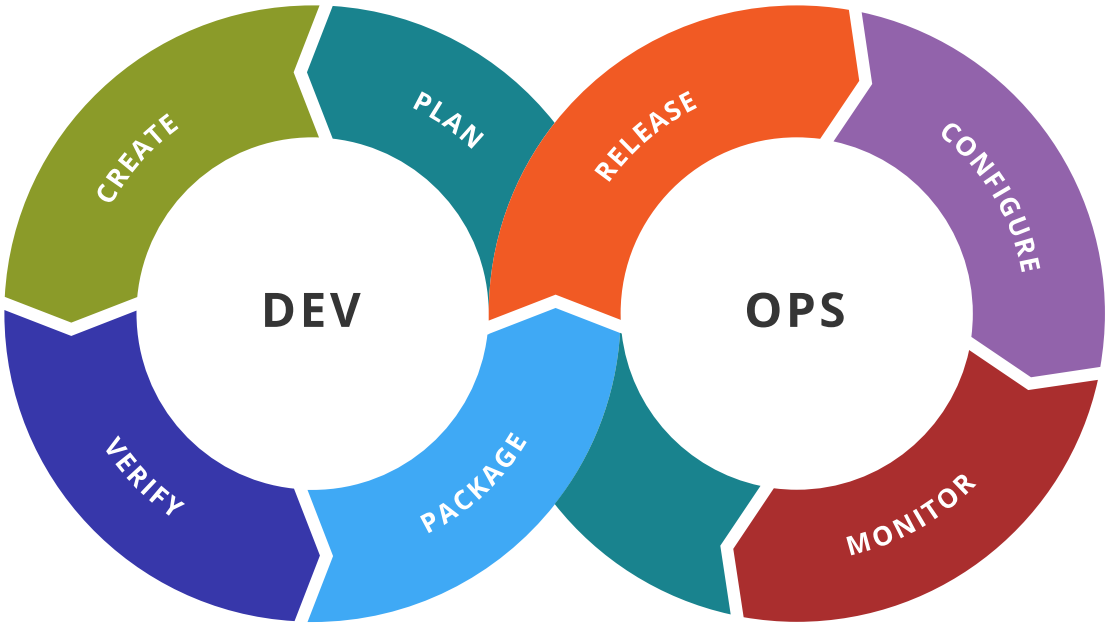
Figure 4.8: Waterfall (OLD!).

content rather than formatting, while still producing readable and well-structured documents. For this reason, Markdown has become the *go-to* language for quick notes, README files, and lightweight documentation – especially in the context of a Git repository, where Markdown files integrate naturally with version control systems and platforms such as GitLab and GitHub.

4.4 CI/CD

Waterfall (OLD!) in figure 4.8. When the author was taught software engineering in 1993, the *Waterfall* model was the dominant development methodology. Waterfall is a sequential process model consisting of clearly separated phases, typically *Requirements*, *Design*, *Implementation*, *Verification*, and *Maintenance* (or some variation thereof). Each phase is intended to be completed before the next one begins, with limited feedback between phases.

In practice, this approach led to careful, plan-driven, and relatively slow development processes. As a result, new versions of a system were often deployed to production only once or twice per year. While this model worked reasonably well for stable requirements and long-lived systems, it struggled to adapt to rapidly changing user needs and



Kharnagy, CC BY-SA 4.0, via Wikimedia Commons

Figure 4.9: DevOps (TODAY!).

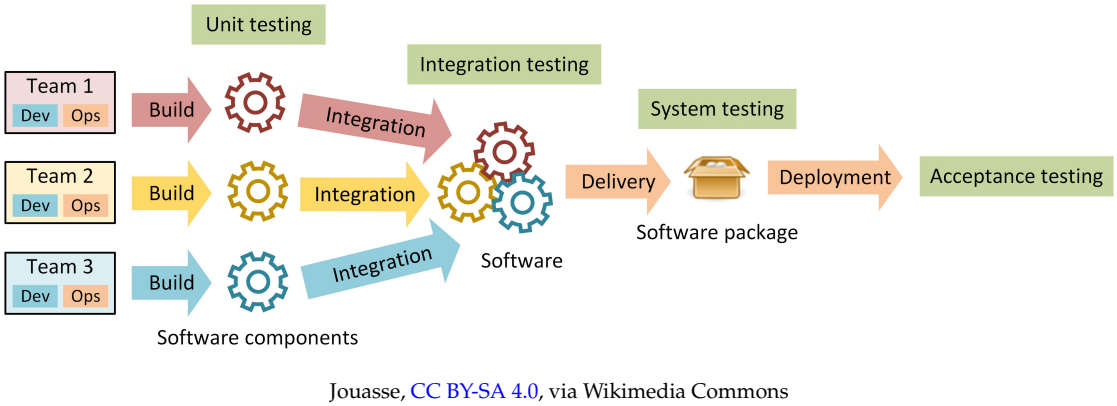
technological environments.

The Waterfall model also reinforced a strict separation between *Developers* (software engineers and programmers) and *Operations* (system and network administrators). Developers were primarily responsible for building the software, while operations staff were responsible for deploying and maintaining it in production. This lack of close cooperation often resulted in misunderstandings, slow feedback cycles, and friction when software moved from development into operational use.

DevOps (TODAY!) in figure 4.9. The *DevOps* model, which is widely used today, facilitates close cooperation between *Developers* and *Operations*. Rather than treating software development and system operation as separate phases handled by different groups, DevOps emphasizes shared responsibility for the entire lifecycle of a system, from development to deployment and operation.

This model has been made possible to a large extent by advances in virtualization and cloud computing technologies. These technologies enable systems to be provisioned, modified, and rolled back quickly and reliably, which in turn allows for fast and frequent changes in production environments. As a result, it is common today to deploy changes to production software on a weekly basis, daily, or – in some organizations – even multiple times per day.

Because security was undervalued or treated as an afterthought for many years, some practitioners refer to *DevSecOps* as a separate model. In reality, DevSecOps is not fun-



Jouasse, CC BY-SA 4.0, via Wikimedia Commons

See our example at
gitlab.com/erikhje/heat-mono/-/blob/master/.gitlab-ci.yml

Figure 4.10: CI/CD Pipeline.

damentally different from DevOps; it simply emphasizes that security considerations must be incorporated into every step of the DevOps process. This includes secure coding practices, automated security testing, and continuous monitoring in production. Incorporating security throughout the development and deployment pipeline is not optional – it is something that should be an integral part of DevOps from the beginning.

4.4.1 Pipeline

CI/CD Pipeline in figure 4.10. To make DevOps work in practice – especially with frequent changes to production software – we must automate everything that can reasonably be automated. This applies in particular to testing, in all its forms and at multiple levels. Manual testing does not scale when changes are deployed frequently, and it quickly becomes a bottleneck in the development process.

Automated testing can include many aspects, such as checking code quality (for example code comments, code style, and best-practice rules), detecting bugs and problematic code patterns, performing security checks, and in some cases even running automated performance tests. The goal is to detect problems as early as possible, ideally before changes ever reach production.

These automated tests are typically executed in a defined sequence of steps, known as a *CI/CD pipeline*. The term *CI* stands for *Continuous Integration* and refers to the practice of frequently integrating changes into a shared code base and automatically validating those changes. The term *CD* can stand for either *Continuous Delivery* (primarily used by developers) or *Continuous Deployment* (primarily used by operations). Continuous delivery means that the software is always in a deployable state, while continuous de-

ployment goes one step further and automatically deploys changes to production once all tests have passed.

CI/CD pipelines can become quite complex, and you will study them in more depth in later courses. In this course, we focus only on the basic idea. Specifically, we configure a very simple pipeline consisting of a single test step that runs automatically every time we push changes to the central Git repository.

In our case, this pipeline performs one task only: it checks whether our PowerShell scripts pass the PSScriptAnalyzer[4] without producing any warnings. PSScriptAnalyzer is a *static code checker* for PowerShell, meaning that it analyzes the code without executing it. Even this simple form of automated testing provides immediate feedback and helps enforce consistent code quality, illustrating the core idea behind CI/CD in a practical and manageable way.

[4] github.com/PowerShell/PSScriptAnalyzer

4.5 Lab tutorials

1. Install git with `choco install -y git` (as always, check if package is OK first[5]). Do "01 Git configuration" from the Git Cheat Sheet at about.gitlab.com/images/press/git-cheat-sheet.pdf.

2. **Shared and Local Repo**.

   (a) Log in on https://git.ntnu.no and create a new git repository with the "plus-button" in the top right corner, choose "New Repository" and fill out the form as follows:

   - Owner: choose yourself
   - Repository name: `mysil` (or whatever you want, but if you name it mysil you can copy and paste more of the text later)
   - Leave it as private
   - Check "Initialize repository with a README"
   - Click "Create repository"

   You now have a remote Git repository that you want to clone locally, make changes to, and push those changes back to the remote repository *without having to enter a password every time*. To achieve this, Git commonly uses *SSH key-based authentication*.

   (b) SSH authentication works by associating a public–private key pair with your user account. The private key remains on your local machine and must be kept secret, while the corresponding public key is uploaded to the Git hosting service (`git.ntnu.no`). When you interact with the remote repository, Git uses this key pair to authenticate you securely and automatically. The following steps add an SSH public key to your account, allowing you to clone, pull, and push changes to the remote Git repository without repeated password prompts.

   (Do this on your laptop or on a Windows host in SkyHiGh)

   Generate a new SSH key pair for GitLab access:

   ```
   ssh-keygen -t ed25519 -C "git.ntnu.no projects"
   ```

   Name the key as proposed (if you use another name, you have to set up a config file) and leave a blank password (this means that if anyone gets hold of your private key, they will have write access to your repo). Copy the public key to your clipboard with

   ```
   Get-Content $HOME\.ssh\id_ed25519.pub | Set-Clipboard
   ```

   i. Log in to your git.ntnu.no account.
   ii. Click your profile photo in the upper-right corner, then click *Settings*.
   iii. In the left sidebar, click *SSH and GPG keys* under *Access*.

[5] community.chocolatey.org/packages/git

iv. Click *New SSH key* or *Add SSH key*.
v. In the *Title* field, add a descriptive label for the new key (for example, "My Laptop" or "SkyHiGh Windows Host").
vi. In the *Key* field, paste the public key you copied in the previous step.
vii. Click *Add SSH key*.

(c) (Do the following on your laptop or on a Windows host in SkyHiGh, dependent on what you used to generate the SSH key previously)

```
git clone git@git.ntnu.no:NTNU_USERNAME/mysil.git
cd mysil
git status
Write-Output "A small edit" >> README.md
Write-Output "# Script TBA" > test.ps1
git status
git add test.ps1
git status
git commit -am "Added a file and a small edit"
git status
git push origin main
```

(d) Visit your project on git.ntnu.no to see the changes and click the "2 commits"-button to see the version history.

3. **Conflicts**. Add some of your fellow students to your newly created git project, have everyone clone the repo, make changes to the same file and push those changes. Study "Merge conflicts" at
docs.gitlab.com/ee/user/project/merge_requests/conflicts.html
and try to solve the conflicts that probably appears.

4.6 Review questions and problems

1. What is the difference between a Git repository and the Git program?
2. Explain the difference between *untracked*, *modified*, and *staged* files in Git.
3. Why does the command `git commit -am "msg"` not include newly created files?
4. What is meant by saying that Git is a *distributed* version control system?
5. What is a merge conflict, and when does it occur?
6. Why is Markdown particularly well suited for documentation in Git repositories?
7. What problem does the DevOps model address compared to the traditional Waterfall model?
8. What is the purpose of a CI/CD pipeline, and what does the simple pipeline used in this course do?
9. The file `file.md` is in state *Unmodified* in a git-repo. What happens when you run the command `git rm file.md`?
10. Create a new directory, change into it and create an empty git repo with `git init`
    Create the following status in the git repo (in other words: create the four files and run git commands, so the status ends up being just like this):

```
PS> Get-ChildItem *.txt | Format-Table -Property Name

Name
----
a.txt
b.txt
c.txt
d.txt

PS> git status

On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
      modified:   a.txt
```

```
      new file:   c.txt

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
      modified:   b.txt

Untracked files:
(use "git add <file>..." to include in what will be committed)
      d.txt
```

5

Active Directory: DNS, LDAP and Kerberos

5.1 Active Directory

Active Directory in figure 5.1. Active Directory is a special database and a core component of most companies' IT infrastructure. It is where accounts (usernames and passwords) and all resources (laptops, workstations, servers, shared file systems, printers,

- LDAP and Kerberos (and (Dynamic) DNS)
- Forests, (Trees), Domains and OUs: Users and Hosts
  Scales to really large organization!

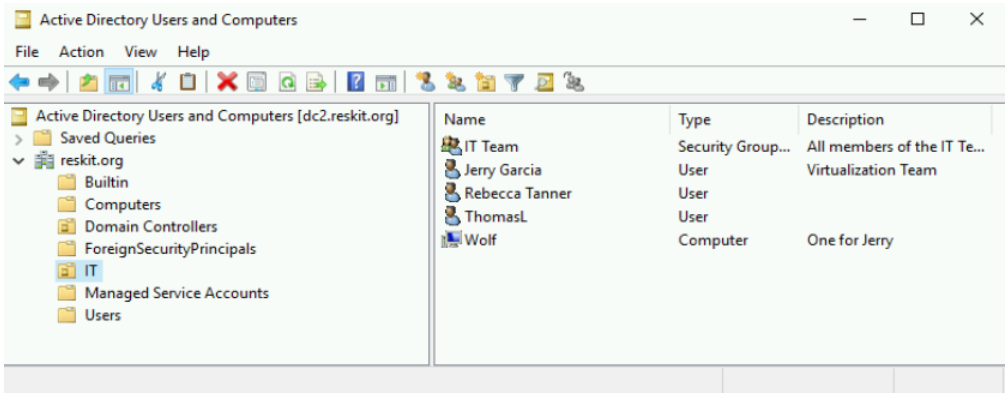


Figure 5.1: Active Directory.

etc.) are stored. Anything in your network that you want to manage with Windows-based management tools should be stored in Active Directory. Active Directory is an implementation of the LDAP and Kerberos protocols, and it also relies heavily on DNS:

LDAP defines how Active Directory is structured and how we can query it for information or add and update entries in the directory.

Kerberos provides authentication in an Active Directory–based infrastructure.

DNS is not a part of Active Directory itself, but it is used to locate computers and the services running on them.

On a Windows Server where Active Directory is installed, the LDAP-structured database (where all resources are stored) is the file `C:\Windows\NTDS\ntds.dit`. Active Directory runs as a set of services, where the most important ones are NTDS, KDC, and DNS:

```
PS> Get-ChildItem C:\Windows\NTDS\ntds.dit

Directory: C:\Windows\NTDS

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---           1/2/2023 12:43 AM       16777216 ntds.dit

PS> Get-Service NTDS,KDC,DNS

Status   Name               DisplayName
------   ----               -----------
Running  DNS                DNS Server
Running  KDC                Kerberos Key Distribution Center
Running  NTDS               Active Directory Domain Services
```

In this chapter, we will become familiar with the three technologies—LDAP, DNS, and Kerberos – before we move on to actually working with Active Directory.

5.2 DNS

DNS - What is it? in figure 5.2. The Domain Name System (DNS) is often described as the *glue* of the Internet. DNS plays a role in many major outages and incidents every year. In 2021, Facebook (now Meta) temporarily disappeared from the Internet, and in their own words [24]:

- DNS is a distributed database with (key, value) pairs
- DNS provides the following primary services
  - name-to-IPaddr mapping (A or AAAA)
  - (IPaddr-to-name mapping (PTR))
  - aliases (CNAME)
  - mail routing (MX)
- Also be used for
  - Other lookups (SRV records, certificates, etc.)
    Service Discovery in Windows domains, e.g. where is login server?
  - Load distribution
  - RBL/SPF (spam prevention)
  - Learn about your network...
  - Passive DNS
  - DNSSEC with SSHFP, DANE, TLSA, ...

Figure 5.2: DNS - What is it?.

One of the jobs performed by our smaller facilities is to respond to DNS queries. DNS is the address book of the internet, enabling the simple web names we type into browsers to be translated into specific server IP addresses. Those translation queries are answered by our authoritative name servers that occupy well known IP addresses themselves, which in turn are advertised to the rest of the internet via another protocol called the border gateway protocol (BGP).

To ensure reliable operation, our DNS servers disable those BGP advertisements if they themselves can not speak to our data centers, since this is an indication of an unhealthy network connection. In the recent outage the entire backbone was removed from operation, making these locations declare themselves unhealthy and withdraw those BGP advertisements. The end result was that our DNS servers became unreachable even though they were still operational. This made it impossible for the rest of the internet to find our servers.

A haiku about DNS in figure 5.3. (Haiku originated in Japan and is a form of poetry written in three short phrases.) DNS is well known for being the source of many problems. Every experienced system administrator has, at some point, spent a long time troubleshooting an issue while being convinced that it could not possibly be related to DNS – only to discover in the end, often after much frustration, that it *was* a DNS issue

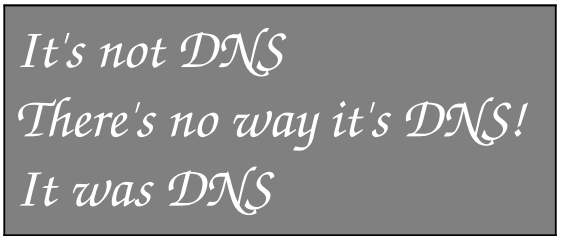


Figure 5.3: A haiku about DNS.

1971	RFC226, HOSTS.TXT (Peggy Karp)
1981	RFC799, DNS concepts (David Mills)
1982	RFC819, DNS structure (Zaw-Sing Su & Jon Postel)
1983	RFC882/883, Hostname lookup, authority and delegation (Paul Mockapetris)
1984	RFC920, Outline of work to be done and TLDs/TopLevelDomains (Jon Postel)
1985	Start of DNS, first name registered (symbolics.com or think.com)

Figure 5.4: DNS History.

after all.

DNS History in figure 5.4. In the early days of the Internet in the 1970s, only a few computers were connected to the network, and *name resolution* (translating between host names and IP addresses) was handled by simply reading from a *hosts-file*. If a new computer was added or an existing name was changed, someone would update the hosts-file and distribute a new version to all machines on the network. This approach worked well as long as there were only a few computers, but it was not a *scalable* solution. As the number of computers and the frequency of updates grew, maintaining a single shared hosts-file became impractical. This eventually led to the development of the Domain Name System (DNS) that we use today.

Note that the hosts-file from the 1970s still exists and remains part of modern systems. Machines on the Internet still check the hosts-file before querying DNS. On Windows, you can find the hosts-file at:
`$env:SystemRoot\system32\drivers\etc\hosts`
Sometimes, when a quick workaround is needed for a service that relies on name resolution, we may temporarily edit the hosts-file as a short-term fix.

- A DNS server is usually divided into *a resolver/cache* and *an authoritative server*. The following servers are common
  - Bind 9
  - Microsoft DNS
  - Many others, see en.wikipedia.org/wiki/Comparison_of_DNS_server_software
- To query DNS servers we usually use the `dig` program on Linux and the `Resolve-DnsName` cmdlet in PowerShell (or `nslookup` on either platform)

Figure 5.5: DNS software.

5.2.1 Software

DNS software in figure 5.5. A DNS server can function as *a DNS resolver/a caching DNS server*, *an authoritative DNS server*, or both. In our case, we will use the Microsoft DNS server, which provides both roles. A DNS resolver (also called a caching DNS server) answers DNS queries by forwarding them to other DNS servers and storing the responses in its cache. An authoritative DNS server, on the other hand, contains unique DNS data: it defines the names in a specific zone/domain and can answer queries for that zone directly. It is authoritative for only a small part of the Internet.

It is important to understand the distinction between these two functions, because separating them supports good system design. Following the "least-privilege" principle, we often install a dedicated caching DNS server for clients to use, while the authoritative server is not contacted directly by hosts. Separating caching and authoritative DNS servers is a solid architectural practice—both for security (isolating services) and for performance (reducing the number of queries that reach the authoritative server).

```
PS> Resolve-DnsName example.com

Name           Type   TTL   Section    IPAddress
----           ----   ---   -------    ---------
example.com    AAAA   444   Answer     2606:2800:220:1:248:1893:25c8:1946
example.com    A      865   Answer     93.184.216.34
```

Domains and Zones in figure 5.6. The concepts *domain* and *zone* can be a bit confusing. A domain is everything under `.no`, but the `.no` zone excludes all its subdomains that have been delegated. For example, all hosts (e.g. `www.ntnu.no`) under `ntnu.no` is part of the `.no` domain but not part of the `.no` zone. An *authoritative DNS server* is responsible for a zone, not a domain. When we talk about the `ntnu.no` domain, we include all names that end with `ntnu.no`, but `ntnu.no` and `iik.ntnu.no` are two separate zones.

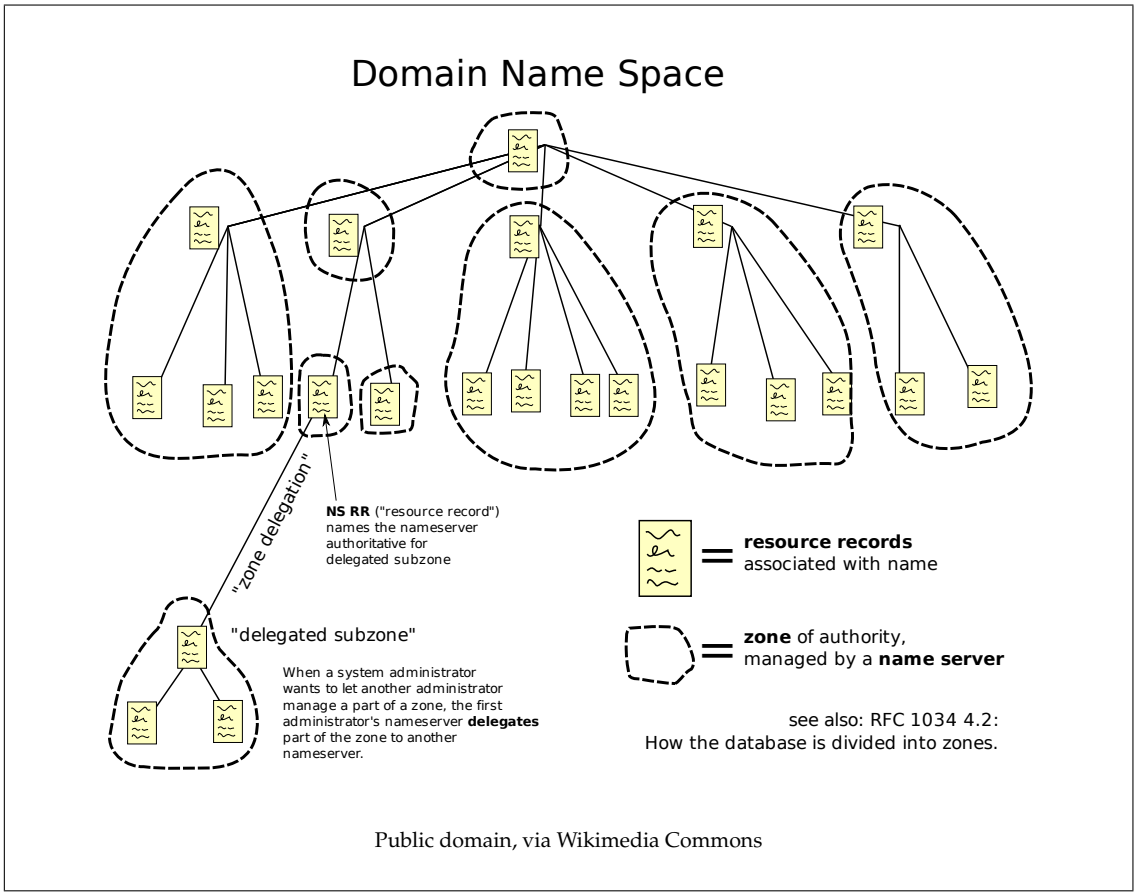


Figure 5.6: Domains and Zones.

However, note that if IIK did not operate its own authoritative name server, we could simply let the `ntnu.no` authoritative name servers handle all names ending in `iik.ntnu.no`. In that case, `ntnu.no` and `iik.ntnu.no` would belong to the same zone.

A few useful things to know about DNS:

- For example: hostname is `www`, domain name is `ntnu.no`, and the Fully Qualified Domain Name (FQDN) is `www.ntnu.no`.
- Names are case-insensitive and must follow the *LDH-rule* (Letters, Digits, and Hyphen). Since 2009, Unicode characters have also been permitted in domain names, allowing the use of Norwegian letters such as æ, ø, and å (although whether this is a good idea is another question). The mechanism that enables this is called Punycode [13], which converts Unicode names into LDH-compliant form *in the application before the DNS query is made*. The DNS system itself still requires the LDH-rule. Try entering `http://blåbærsyltetøy.no` in your browser (this will probably work) and then try running `Resolve-DnsName blåbærsyltetøy.no` (this will likely not work).

- DNS works by passing around *Resource Records (RRs)* through port 53 (TCP if larger than 512 bytes; else UDP)
- The most important resource records are

  SOA Start Of Authority (accepts a delegation)

  NS Name Server (delegates a zone)

  MX Mail eXchanger (defines the mail server)

  A Address, define the canonical name of an IP address

  CNAME Canonical Name, define an alias

  PTR PoinTer Record, define the reverse mapping (the IP address of a fqdn)

  SRV Service record, which hosts and ports have the service

Figure 5.7: How a Query Works.

- Sometimes, in DNS contexts, we need to include the top-level root node as well – the trailing dot: `www.ntnu.no.`
- The top-level root node has a series of root servers holding information about all Top Level Domains (TLDs).
- The current list of TLDs (Top Level Domains) can be found at https://www.icann.org.
- Information about the root servers is available at https://root-servers.org.
- The thirteen root servers are normally listed in a configuration file on DNS servers (e.g. `C:\Windows\System32\dns\cache.dns` in Microsoft DNS and `named.root` in BIND). The file can also be downloaded from https://www.internic.net/domain/named.root.

5.2.2 DNS Query

How a Query Works in figure 5.7. A resource record (the DNS data packet, which is in most cases transported in a UDP packet) always contains the following fields:

Name e.g. the name we want to translate into an IP address.

Type e.g. `A`.

Value "the answer", e.g. the corresponding IP address.

TTL Time-To-Live, the time interval during which the resource record may be cached before it must be retrieved again from the authoritative server.

For example, if we want to find out which server(s) accept email for the domain `ntnu.no`:

```
PS> Resolve-DnsName -Type MX ntnu.no

Name     Type   TTL   Section    NameExchange    Preference
----     ----   ---   -------    ------------    ----------
ntnu.no  MX     39    Answer     mx.ntnu.no      10

Name       : mx.ntnu.no
QueryType  : A
TTL        : 101
Section    : Additional
IP4Address : 129.241.56.67

Name       : mx.ntnu.no
QueryType  : AAAA
TTL        : 255
Section    : Additional
IP6Address : 2001:700:300:3::67
```

or which port number we use for LDAP TCP-queries in our `sec.core` domain[1]:

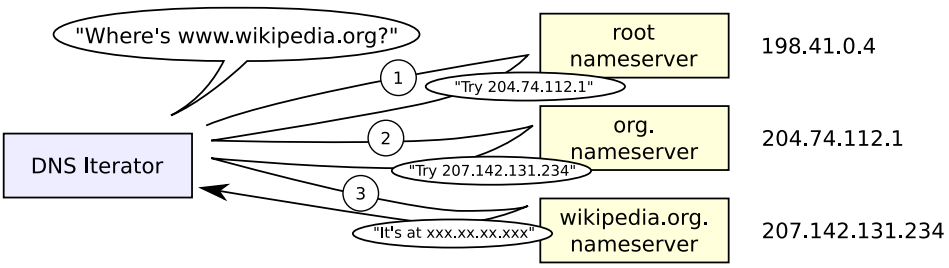
```
PS> Resolve-DnsName -Type SRV _ldap._tcp.sec.core |
      Select-Object -Property Name,IP4Address,Port

Name                IP4Address      Port
----                ----------      ----
_ldap._tcp.sec.core                  389
dc1.sec.core        192.168.111.103
```

A DNS query ("asking a server for a resource record") can be either *recursive* ("do whatever you can to resolve this" — this is what a client sends to a resolver) or *iterative* ("please answer this without asking anyone else" – this is what a resolver sends to an authoritative server). In the DNS data packet, there is a bit called RD, which is set to 1 for a recursive query and 0 for an iterative query.

A host can have multiple names. For example, in a small organization you might have a server running both SMTP and IMAP services, with the names `smtp.example.com`, `imap.example.com`, `mail.example.com`, and `mikke.example.com`. `mikke` is likely the original name of the server (sometimes called the *canonical name*), while `smtp`, `imap`, and `mail` are service names that identify the specific services the host provides.

These names should be implemented either as additional `A` records or as `CNAME` records. Using `A` records is more efficient, since `CNAME`s require two lookups (a `CNAME` points to an

[1] We will install the `sec.core` domain in this chapter's lab tutorial.



Lion Kimbro, Public domain, via Wikimedia Commons

Figure 5.8: Interaction of DNS servers.

`A` record, so you must resolve the `A` record to obtain the IP address). However, `CNAME`s make it easier to distinguish between service names and the canonical name.

Separating names into a canonical hostname ("the real name of the server") and service names is a good practice, since services may later need to be moved to a different host without changing client-facing names.

Interaction of DNS servers in figure 5.8. The DNS dialogue happens as follows (assuming all involved servers have just started and their *caches* are empty):

1. Check the hosts-file (`$env:SystemRoot\system32\drivers\etc\hosts` on Windows).
2. Check the local configuration to determine which resolver to use (via `Get-DnsClientServerAddress`).
3. Send the resolver a recursive query for `www.wikipedia.org`.
4. (1) The resolver checks whether it already has the `A` record for `www.wikipedia.org`, `wikipedia.org`, or `.org`. Since it does not, it must contact (send an iterative query to) one of the root servers.
5. The root server responds with the NS records for the `.org` zone along with their corresponding `A` records (these `A` records are called "glue records").
6. (2) The resolver then sends an iterative query to one of the `.org` servers.
7. The `.org` authoritative server replies with the `NS` records for `wikipedia.org`, along with their `A` records (again, glue records).
8. (3) Finally, the resolver sends an iterative query to one of the `wikipedia.org` authoritative servers, which responds with the `A` record for `www.wikipedia.org`.

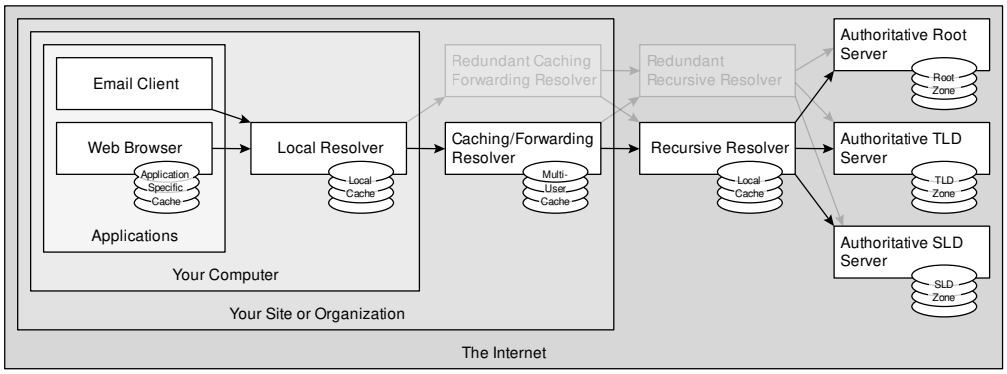


Figure 5.9: Several Caches Involved.

Several Caches Involved in figure 5.9. You can easily imagine that if every DNS query on the Internet had to begin by contacting the root servers, those servers would quickly become overloaded, and the Internet would be flooded with DNS traffic. This is why all resolvers cache the replies they receive. *Caching* means storing a copy of data obtained from a server so that the next time the same information is needed, the resolver can return the answer from its cache instead of querying the server again.

A key question then becomes: how long should cached data be kept? An authoritative server may update its information, and cached data might become outdated. The duration for which data may remain in cache is defined in a field of the DNS Resource Record called the TTL (Time To Live). The TTL is set by the authoritative name server—for example, it might be one hour (3600 seconds). This means that a DNS response may be outdated if the authoritative DNS data was changed within the last hour.

You can inspect which resource records are cached on your computer using `Get-DnsClientCache`. Run the cmdlet twice and compare the TTL values—you should see the TTL decreasing over time.

5.2.3 Dynamic DNS

Dynamic DNS means that a client can notify an authoritative DNS server that it has a new IP address, and the server will update the client's DNS entry so that the same name now maps to the new address. *This mechanism can also be used for other kinds of DNS records (e.g., service records (SRV) in Active Directory).* Active Directory relies on dynamic DNS to update its SRV records, allowing clients to discover which IP addresses and ports various services in the Active Directory infrastructure are running on.

- Security, see table of contents and browse chapter two of DNS Security Introduction and Requirements at
  tools.ietf.org/html/rfc4033
- Privacy, see DNS Privacy - The Problem at
  dnsprivacy.org/wiki/display/DP/DNS+Privacy+-+The+Problem

Figure 5.10: Security and Privacy.

5.2.4 DNS security

Security and Privacy in figure 5.10. A key problem with DNS is that it is vulnerable to cache poisoning, with one of the most well-known vulnerabilities being "Multiple DNS implementations vulnerable to cache poisoning"[2], where DNS servers that did not randomize the UDP source port were susceptible to attack.

DNS operates by sending messages called Resource Records inside UDP packets. It uses UDP because the protocol is lightweight and simple; if a packet is lost, it can simply be retransmitted. However, DNS also has significant security and privacy concerns [18], and implementations of DNS over HTTPS (DoH) are increasingly being deployed. DoH appears to work well without a major performance penalty [9].

However, DoH introduces a new privacy concern: if most users rely on a small number of public DoH providers[3], those providers gain access to everyone's DNS queries, instead of this metadata being distributed across many local ISPs.

Security and privacy in DNS is a major topic that you should be aware of, but we will not study it in depth in this course. Our focus will remain on how DNS works in general, and specifically how it functions within a Windows domain.

5.2.5 Our Setup

Our Setup Before AD Install in figure 5.11. The infrastructure we will work with during the next weeks consists of two Windows clients and two Windows servers. Read the text in the figure carefully so that you understand the role of each host. When we install a DNS server on DC1 (which happens as part of the Active Directory installation), we will receive the following warning:

> WARNING: The following recommended condition is not met for DNS: No static IP addresses were found on this computer. If the IP address changes, clients might not be able to contact this server. Please configure a static IP address on this computer before installing DNS Server.

[2] www.kb.cert.org/vuls/id/800113
[3] github.com/curl/curl/wiki/DNS-over-HTTPS#publicly-available-servers

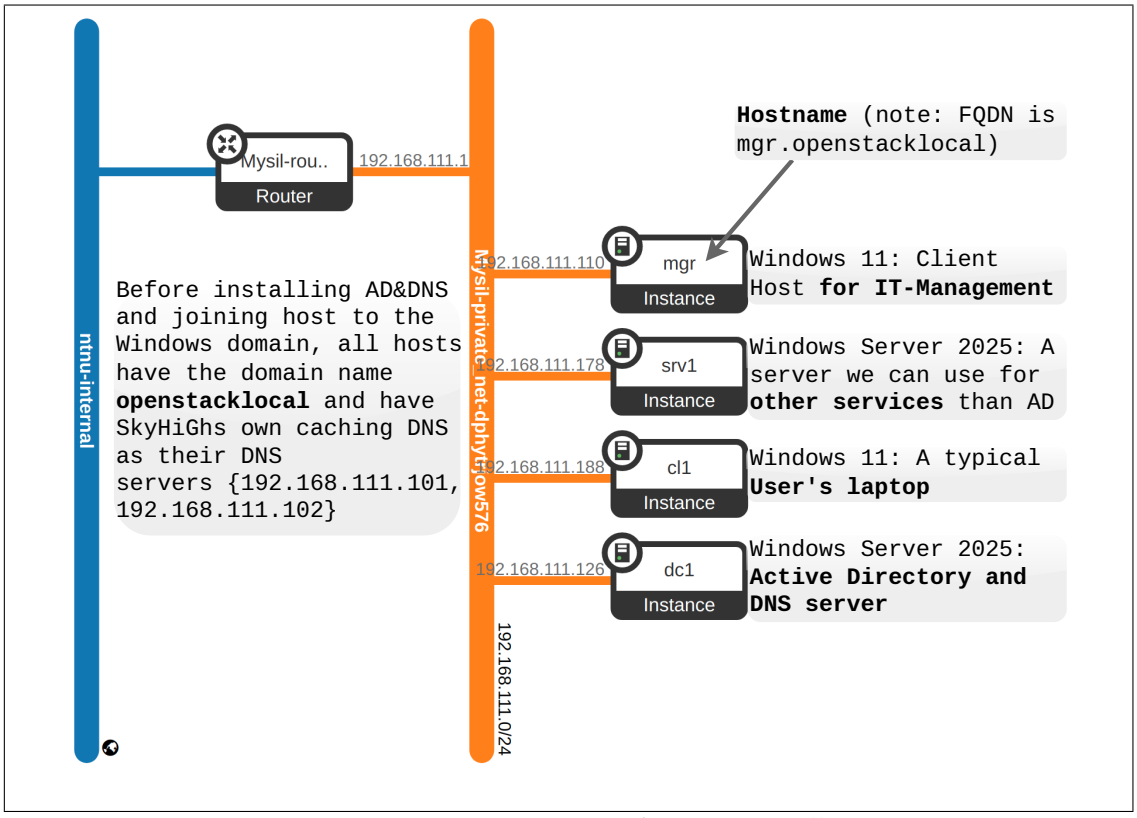


Figure 5.11: Our Setup Before AD Install.

We can ignore this warning because of the way IP address assignment works in cloud infrastructures: IP addresses are always assigned to hosts through DHCP (Dynamic Host Configuration Protocol), which normally means that they may change during a host's lifetime. However, in both public and private cloud environments, the cloud platform (such as OpenStack) guarantees that each host will consistently receive the same IP address from the DHCP server for its entire lifetime.

This means that even though the warning message "the host does not have a static IP address" is technically correct, it is not a problem in practice, because the dynamically assigned IP address effectively behaves like a static one – it does not change. For example, in Azure [37]:

> A virtual machine (VM) is automatically assigned a private IP address from a range that you specify. This range is based on the subnet in which the VM is deployed. The VM keeps the address until the VM is deleted.

Our Setup After AD Install in figure 5.12. Read the text in the figures carefully so that you understand how each computer's DNS configuration changes when we install Active Directory and join the computers to Active Directory (creating a *Windows Domain*).

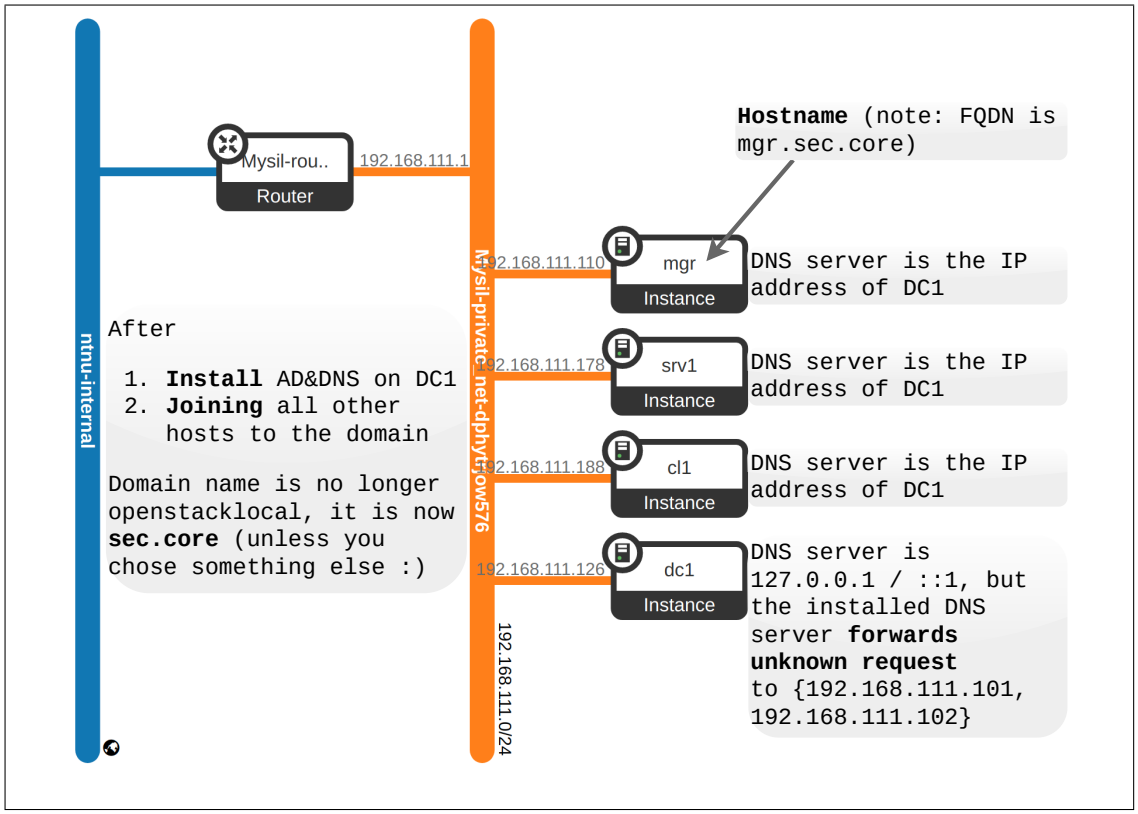


Figure 5.12: Our Setup After AD Install.

DNS and *LDAP (Lightweight Directory Access Protocol)* are examples of directory services:

- A simple database *optimized for reads and searches (lookups)*
- Allows for centralized storage of the resources you have in your network: users, groups, computers, services, printers, mailing lists, ...

Figure 5.13: What is a Directory Service?.

5.3 LDAP

What is a Directory Service? in figure 5.13. A directory is a special database that you use for finding values stored in attributes (properties). The ratio between reads and writes in a directory service is typically 1000:1, 10 000:1, or even 100 000:1. There are very frequent searches and reads, while new writes (new registrations or updates) occur much less often.

LDAP is a directory service similar to DNS, but it serves a different purpose. While DNS is the glue of the Internet, LDAP (as implemented in Active Directory) is the glue

- LDAP is a client-server protocol for communication with a directory service
- LDAP structure:
  - a tree (a hierarchy) of directory entries (records/objects)
  - an entry (record/object) consists of a set of attributes
  - an attribute has a name and one or more values
- *LDAP uses TCP on port 389* (or SSL-connection to 636 in addition in some setups)

Figure 5.14: Lightweight Directory Access Protocol.

for resource management inside a company's IT infrastructure. Both are directory services primarily used for reads ("lookups"), but their internal structures differ. DNS is a flat database optimized for name resolution, while LDAP organizes data in a hierarchical tree structure, allowing for more complex relationships between entries. LDAP is designed to store a wide variety of resources, such as users, groups, computers, services, printers, and mailing lists, making it a versatile directory service for managing an organization's IT resources.

5.3.1 Structure

Lightweight Directory Access Protocol in figure 5.14. The actual data in LDAP can be stored in any type of backend, such as flat files or a relational database. The requirement, however, is that the data must be accessed (read, searched, updated, added, modified, etc.) according to the LDAP protocol. In other words, LDAP is a protocol (a set of rules) that defines how communication between an LDAP client and an LDAP server should take place. The LDAP protocol specifies that an LDAP client must be able to request data from the server in a specific structure: a tree-like hierarchy of entries, where each entry contains a set of attributes.

LDAP Structure in figure 5.15. The directory tree structure (hierarchical structure) is similar to a file system. In the same way that we address file paths in a file system using absolute or relative paths (remember: absolute paths start with a "/"), we address entries in a directory tree using the *distinguished name (DN)* or the *relative distinguished name (RDN)*:

- *DN = RDN + Parent's DN*
- DN is the unique identifier (the "primary key")
- RDN is unique identifier only at its own level
- A DN is typically composed of

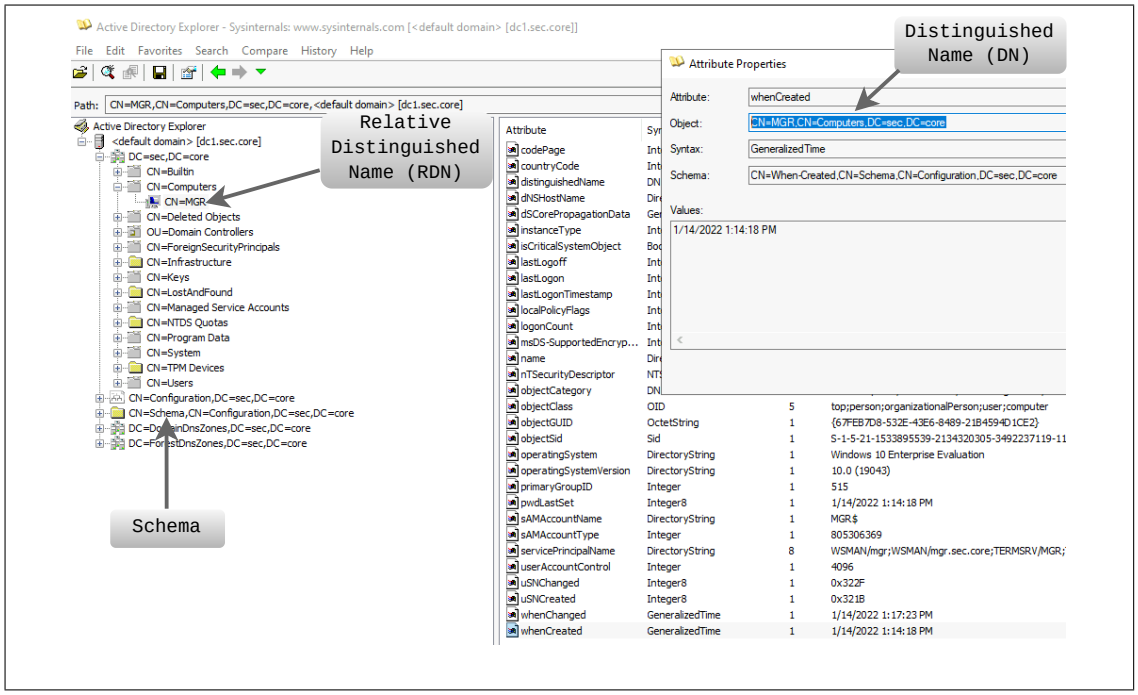


Figure 5.15: LDAP Structure.

DC Domain Component

OU Organizational Unit

CN Common Name

E.g. in NTNU's LDAP directory we can have an entry with RDN: `uid=erikhje` and DN: `uid=erikhje,ou=iik,ou=ie,dc=ntnu,dc=no`). Maybe NTNU's LDAP has a structure like this:

```
dc=no
 |
 +--dc=ntnu (these could be one entry dc=no,dc=ntnu)
     |
     +--ou=ie
         |
         +-ou=iik
            |
            +-uid=erikhje
            |     roomNumber=T518
            |     mobile=93034446
            |     .
            |     .
            +-uid=eigilo
```

```
+-uid=erjonz
  .
  .
```

The *common name (cn)* is an alternative name and an attribute that identifies an entry at its own level, so the CN is often the same as the RDN. In NTNU's LDAP and in Active Directory, the CN is set to be the Name attribute.

The Windows tools `Get-ADObject` and `dsquery` will only query LDAP on port 389. If we want to access NTNU's LDAP without authenticating first, we must use LDAP over SSL on port 636. To demonstrate access to NTNU's LDAP, we can use `ldapsearch` on Linux (from the `ldap-utils` package; try this on `login.stud.ntnu.no` if you want to):

```
# note: if newer version of ldapsearch do
# -H LDAP://at.ntnu.no instead of -h at.ntnu.no
# find Frode Haugs phone number:
ldapsearch -x -b "ou=people,dc=ntnu,dc=no" -h at.ntnu.no \
"(&(mail=*frode*)(mobile=*95*))" sn givenName mobile
# Note that non-ASCII characters are returned base64-encoded,
# e.g. the authors last name will look strange, you can decode like this:
ldapsearch -x -b "ou=people,dc=ntnu,dc=no" -h at.ntnu.no \
"(mail=erik.hjelmas@ntnu.no)" sn givenName mobile
echo SGplbG3DpXM= | base64 -d && echo
# asking LDAP using the common name (CN):
ldapsearch -x -b "ou=people,dc=ntnu,dc=no" -h at.ntnu.no \
"(cn=Frode Haug)" sn givenName mobile
```

LDAP lookups can be used to connect to NTNU LDAP from your email client[4] to have automatic address lookups.

5.3.2 Schema

Exactly how the LDAP is structured (which entries are allowed, which attributes must be present, etc.) is decided by the *schema*:

- Entries (objects) are instance of an objectClass
- *ObjectClass* is the link to Schema which defines the object
- From RFC4512 4. Directory Schema: The schema enables the Directory system to, for example:

[4] innsida.ntnu.no/wiki/-/wiki/English/Configuring+LDAP

Boolean expressions (true/false) in *Prefix* notation (aka Polish notation), not infix or postfix.

- () group with parenthesis
- & AND operator
- | OR operator
- ! NOT operator
- * Wildcard
- =, >=, <= comparison operators

e.g. `(&(!Name=A*)(logonCount>=10))`

Figure 5.16: Search Syntax.

  - prevent the creation of subordinate entries of the wrong object-class (e.g., a country as a subordinate of a person)
  - prevent the addition of attribute-types to an entry inappropriate to the object-class (e.g., a serial number to a person's entry)
  - prevent the addition of an attribute value of a syntax not matching that defined for the attribute-type (e.g., a printable string to a bit string).

The Active Directory LDAP schema includes the attributes required for Unix/Linux user accounts (POSIX attributes), allowing Active Directory to be used in heterogeneous environments where users may have a mix of Windows, macOS, and Linux systems.

5.3.3 Search Syntax

Search Syntax in figure 5.16. Prefix notation (also known as Polish notation) means that the operator comes first. In a mathematical expression this would mean writing `+ 2 2` instead of `2 + 2`. To perform LDAP queries in a Windows environment, we need the Active Directory tools installed. These tools are installed automatically together with Active Directory on DC1, but since we want to work on a Windows 11 machine (e.g., the host MGR), we can install them with the following command:

```
Add-WindowsCapability -Online `
  -Name 'Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0'
# test with e.g.
Get-ADObject -LDAPFilter '(sAMAccountName=*)'
```

- StartTLS
- Bind
- Search
- Compare
- Add (*atomic*)
- Delete (*atomic*)
- Modify (*atomic*)
- Modify DN (move entry) (*atomic*)
- Abandon
- Extended operation
- Unbind

Figure 5.17: LDAP Operations (”Commands”).

Knowing how to perform LDAP searches is essential for our ability to understand, troubleshoot, investigate, and automate tasks related to Active Directory.

5.3.4 Operations

LDAP Operations (”Commands”) in figure 5.17. A connection to an LDAP server is initiated with StartTLS and Bind on port 389 (TLS is Transport Layer Security[5], which ensures encrypted communication). A non-encrypted connection is first opened on port 389, and then the StartTLS operation is issued; from that point onward, the communication is encrypted. This is how LDAP connections work in Active Directory.

Some LDAP servers use an alternative approach: they establish an encrypted SSL connection *before* initiating the LDAP dialogue. In that case, port 636 is used instead of port 389.

All write operations to an LDAP server are *atomic*, meaning they are either fully completed or not executed at all—they cannot be interrupted or partially applied (think of “atomic” like an atom: something that cannot be split into smaller parts). Atomic operations are extremely important in network applications (such as LDAP client–server communication), because network outages or simultaneous updates from multiple clients could otherwise leave the server with inconsistent data. We will exam-

[5]TLS and SSL are basically the same thing.

ine atomic operations further in both the Operating Systems course and the Database course.

Note that both Windows and Linux have a `services` file where most standard service ports are listed. For example, to check which ports LDAP uses by default, we can ask PowerShell to return all lines in the `services` file that contain the word "LDAP":
`Select-String LDAP C:\Windows\System32\drivers\etc\services`

In this course we study LDAP because Active Directory is an implementation of LDAP, but you also use LDAP every day without necessarily being aware of it. Your FEIDE-ID (your NTNU account) can be used across universities, airports, and many other locations around the world because information exchange between FEIDE-enabled services is based on LDAP. The basic architecture of FEIDE[6] is built around LDAP. At NTNU, the FEIDE component "LDAP Brukerkatalog" is the server `at.ntnu.no`, which runs OpenLDAP rather than Active Directory.

As of January 2024, FEIDE is the authentication backend for 453 services[7], and you probably authenticate with FEIDE every day without thinking about it. The takeaway is that LDAP is widely used today, has been widely used for the past 20 years, and will remain important for the next 20 years as well. Learning LDAP is lasting knowledge.

LDAP also plays a role in the famous log4j vulnerability [15]:

> An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled.

5.4 Kerberos

To control access to resources (printers, files, email, databases, etc.) in an infrastructure, we need *identification*, *authentication*, and *authorization*. Most of the time we discuss these concepts in relation to users, but they also apply to computers, which often need to authenticate themselves to other computers.

Identification is the act of presenting a username (or computer hostname) to claim who you are.

Authentication is the process of verifying that the claimed identity is correct, typically by demonstrating knowledge of a password (and possibly additional factors such as a code from an app, a USB security key, biometrics, etc.).

Authorization is about granting access to a specific resource, e.g., allowing you to use a particular printer or write to a shared directory in a file system.

[6] www.feide.no/teknisk
[7] www.feide.no/tjenester-med-feide-innlogging

Traditional (and still commonly used) authentication works by entering a username and password on a client and sending these over an encrypted connection (TLS/SSL) to a server. The server then checks that the username exists in its user database, encrypts the provided password [61], and compares it with the encrypted password stored in the database. If they match, authentication is successful.

Kerberos [56] uses a different approach in which the password is never transferred over the network. The user asserts their identity by sending their username, and the server encrypts a random value using the user's stored encrypted password, then sends this value back. If the user can decrypt it successfully, they have proven their identity. This is a simplified explanation of how Kerberos performs authentication without transmitting passwords across the network.

Kerberos does much more than this initial authentication step. Microsoft chose Kerberos as the authentication system for Active Directory in the 1990s because it also provides *single sign-on*. Single sign-on means that you authenticate once, and after successful authentication you are automatically authenticated to the various services you need for a certain period of time (e.g., you may only need to enter your password once every 24 hours).

Kerberos in figure 5.18. Kerberos is a protocol with six steps. There are of course many details involved that we will skip here, since our goal is to understand the basic idea:

1. The client sends its identity to the domain controller (in this context also called the KDC, the Key Distribution Center), where the Authentication Server checks that the client exists.
   "Hello, I am erikhje."

2. The server sends the client a *Ticket-Granting Ticket (TGT)* (along with a challenge that the client must decrypt to prove its identity before the TGT can be used).
   "Confirm that you are erikhje by decrypting this with your password, and then you can use this TGT for the next 8–10 hours to request access to services."

3. The client looks up the *Service Principal Name (SPN)* of the service it wants to use, and sends both the TGT and the SPN to the server.
   "I would like to use the file server (identified by this SPN), and here is my valid TGT."

4. The server sends the client a *Ticket-Granting Service (TGS)* ticket.
   "Yes, the SPN you provided belongs to the file server, and your TGT is valid. Here is a TGS you can use to access the file server."

5. The client sends the TGS to the application server.
   "I would like to access a directory on this file server; here is my TGS."

6. The application server provides the requested service (e.g., access to a shared directory on a file server).
   "Your TGS is valid, so you are allowed to access the file server. However, I will check my

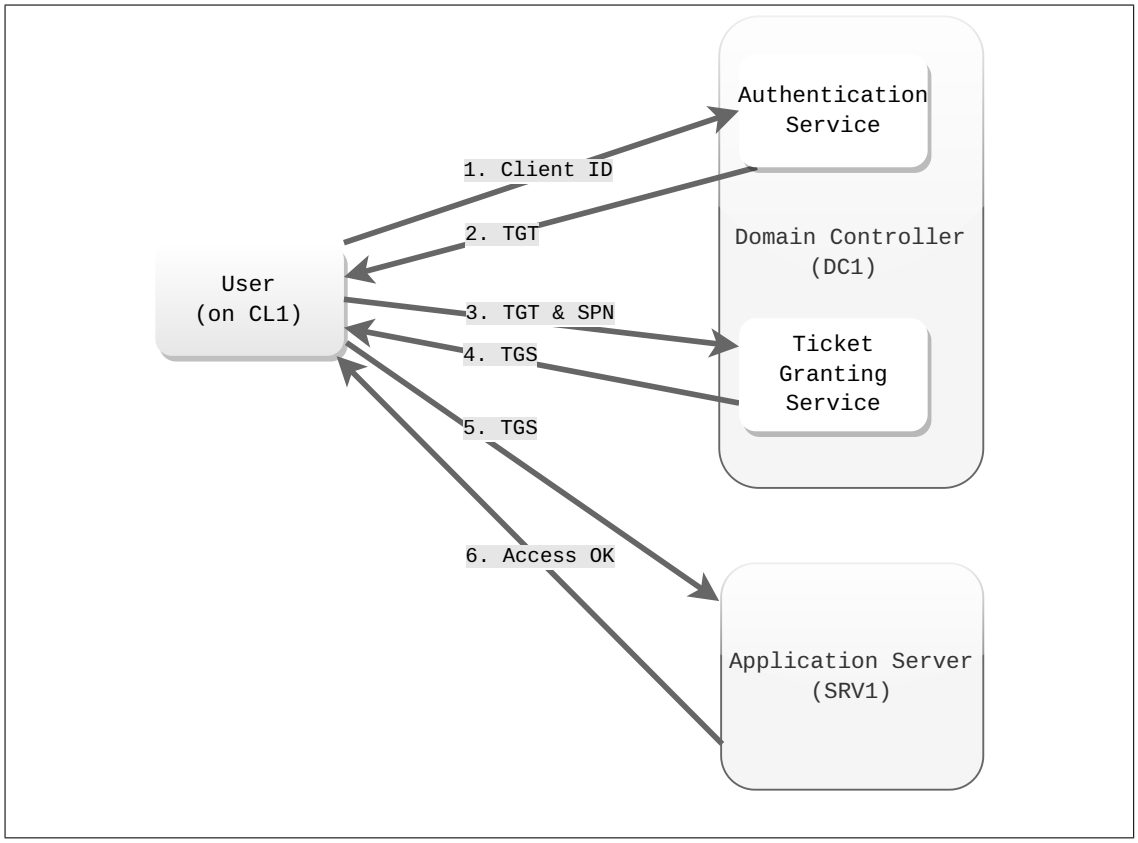


Figure 5.18: Kerberos.

internal access control lists to see whether you are permitted to access the specific directory you requested, since that is not determined by Kerberos."

According to Microsoft a Service Principal Name (SPN) is [46]

> "A service principal name (SPN) is a unique identifier of a service instance. SPNs are used by Kerberos authentication to associate a service instance with a service logon account. This allows a client application to request that the service authenticate an account even if the client does not have the account name."

In other words, Kerberos allows for mutual authentication, meaning that a client can also ask a server to authenticate itself to ensure that the information it receives can be trusted. Microsoft's description of a client authenticating a server is [31] (note: KDC is the Key Distribution Center, which is the Kerberos service):

> "The KDC searches the forest for a user or computer account on which that SPN is registered. If the SPN is registered on more than one account,

> the authentication fails. Otherwise, the KDC encrypts a message using the password of the account on which the SPN was registered. The KDC passes this encrypted message to the client, which in turn passes it to the service instance. The service uses the SSPI negotiate package to decrypt the message, which it passes back to the client and on to the client's KDC. The KDC authenticates the service if the decrypted message matches its original message."

We can list SPNs on a domain controller with the following command (note how an SPN is composed of a service name, e.g., `TERMSRV` (terminal services, also known as remote desktop), and a hostname indicating where the service is running, e.g., `dc1.sec.core` or simply `DC1`):

```
PS> Get-ADObject -Filter {Name -eq 'DC1'} -Property ServicePrincipalName |
                  Select-Object -ExpandProperty ServicePrincipalName
Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/dc1.sec.core
ldap/dc1.sec.core/ForestDnsZones.sec.core
ldap/dc1.sec.core/DomainDnsZones.sec.core
TERMSRV/DC1
TERMSRV/dc1.sec.core
DNS/dc1.sec.core
GC/dc1.sec.core/sec.core
RestrictedKrbHost/dc1.sec.core
RestrictedKrbHost/DC1
RPC/a6512bd5-ccd9-4095-b4db-de21e28f2f3e._msdcs.sec.core
HOST/DC1/SEC
HOST/dc1.sec.core/SEC
HOST/DC1
HOST/dc1.sec.core
HOST/dc1.sec.core/sec.core
ldap/DC1/SEC
ldap/a6512bd5-ccd9-4095-b4db-de21e28f2f3e._msdcs.sec.core
ldap/dc1.sec.core/SEC
ldap/DC1
ldap/dc1.sec.core
ldap/dc1.sec.core/sec.core
```

We see from this list that on a newly installed Active Directory server (a domain controller), most SPNs are related to DNS, LDAP, Kerberos (commonly abbreviated KRB), or the terminal service (which provides our remote desktop connection to the server).

Important characteristics to know about Kerberos are:

- Single sign-on.

- Kerberos provides centralized authentication (each service does not need its own username–password database).
- Timestamps are important because tickets have a limited lifetime. If the clocks on the hosts are not synchronized, authentication may fail.

There are several attack techniques [53] against Kerberos. The following attacks are among the most widely known and are important to be aware of:

Golden Ticket is a "fake" TGT.

Silver Ticket is a "fake" TGS ticket.

Kerberoasting is the process of obtaining additional TGS tickets by cracking weak SPN account passwords.

All these attacks depend on first gaining access as a regular Active Directory user, and most of them exploit either weak passwords or the use of outdated ("legacy") password encryption algorithms.

5.5 Lab tutorials

1. Install the `sec.core` domain according to "PowerShell tutorial 2: Domain-Joined Hosts and Remoting", chapter 1.7. Verify that all four computers are now registered in the domain with
   `Get-ADComputer -Filter * | Select-Object -Property Name`

2. After installing Active Directory and the Microsoft DNS server (which is installed as part of the process), list all *Resource Records (RRs)* in the DNS server (as Domain Administrator):
   `Get-DnsServerResourceRecord -ZoneName sec.core`
   Study carefully the SRV-records so you understand how DNS is used to locate services in a Windows domain (an Active Directory infrastructure).

3. **DNS (Domain Name System)**.

   i Log in to CL1 and DC1. Which DNS server(s) do they use? Run this cmdlet on both hosts

   ```
   Get-DnsClientServerAddress
   ```

   ii If you try to lookup the hostnames `srv1` and `cl1`, which domain suffixes are attempted added? Run this cmdlets on both hosts

   ```
   Get-DnsClient | Format-Table `
    -Property InterfaceAlias,InterfaceIndex,ConnectionSpecificSuffix
   Get-DnsClientGlobalSetting
   @('srv1','cl1') | Resolve-DnsName
   ```

   iii What is in their local cache? Run these cmdlets on both hosts

   ```
   Get-DnsClientCache
   Clear-DnsClientCache
   Get-DnsClientCache
   ```

   iv Do a DNS lookup to find the IP address of `rtfm.mit.edu`

   ```
   Resolve-DnsName rtfm.mit.edu
   ```

   The CNAME resource record represents an alias. It means it is a host name that maps to another host name. The A resource record is what we are really looking for. It contains the host name to IP address mapping.

   v Do a reverse DNS lookup to find the host name of ip address we found for `rtfm.mit.edu`

   ```
   $ip = (Resolve-DnsName rtfm.mit.edu).IP4Address
   $ip
   Resolve-DnsName $ip -Type PTR
   ```

The PTR resource record contains the reverse mapping. Note that some hosts might have A records but not PTR records, meaning a host name to IP address lookup will work, but not the other way around.

vi Skip the internal DNS server and ask two other servers for the lookup instead: first one of NTNUs servers, then one of Google's public servers.

```
Resolve-DnsName -Server 129.241.0.200 rtfm.mit.edu
Resolve-DnsName -Server 8.8.4.4 rtfm.mit.edu
```

In the output from the Resolve-DnsName command shown below, make sure you understand the TTL (Time To Live) number. Run one of the commands above repeatedly and see how the number changes.

```
Name             Type   TTL   Section    NameHost
----             ----   ---   -------    --------
rtfm.mit.edu     CNAME  1799  Answer     xvm-75.mit.edu
```

vii Log in to login.stud.ntnu.no and observe the entire hierarchy of servers involved in a fresh DNS lookup (here you will also see other resource records: the NS (Name Server) record used to delegate responsibility for a domain to another DNS server, and the RRSIG which belongs to DNSSEC)

```
dig @129.241.0.200 +trace rtfm.mit.edu
```

Notice that some servers (root name servers) are responsible for the top level called just `.`, some servers are responsible for `edu.` and some servers are responsible for `mit.edu.` and one of them (the ones responsible for `mit.edu.`) will return the IP address of `rtfm.mit.edu`. *Notice that it is not very often that we ask the root name servers since there is caching used at all levels (temporary storage for a time indicated by the TTL (Time To Live) value).* Do you know what caching is? Ask teacher if unsure.

4. **Testing our domain**.

i Let's see which lookups works. Do this on MGR.

```
$hostnames = @(
  'dc1',
  'srv1',
  'cl1',
  'mgr'
)
$hostnames | Resolve-DnsName
$hostnames | ForEach-Object {Resolve-DnsName "$_.sec.core"}
```

ii Which DNS server is authoritative for the domain `sec.core`?

```
Resolve-DnsName sec.core
Resolve-DnsName -Type SOA sec.core
```

SOA is the ”Start Of Authority” resource record.

iii Go back to CL1 and let's see how a Windows Domain uses DNS for service discovery (SRV records in DNS)

```
# Where is the "primary domain controller"?
# (the one holding FSMO roles)
Resolve-DNSName _ldap._tcp.pdc._msdcs.sec.core -Type SRV

# Where are all the domain controllers?
Resolve-DNSName _ldap._tcp.dc._msdcs.sec.core -Type SRV

# Where is the global catalog?
Resolve-DNSName _ldap._tcp.gc._msdcs.sec.core -Type SRV

# where is the Kerberos server(s)?
Resolve-DNSName _kerberos._tcp.dc._msdcs.sec.core -Type SRV
```

5.6 Review questions and problems

1. What are the three core technologies that Active Directory relies on, and what role does each of them play?
2. What is the difference between a DNS *domain* and a DNS *zone*?
3. Explain the difference between recursive and iterative DNS queries.
4. What is TTL in DNS, and why is it important?
5. What is Dynamic DNS and how does Active Directory use it?
6. Why does Windows warn about not having a static IP address when installing DNS, and why can this warning be ignored in cloud environments?
7. What is LDAP, and how does its directory structure resemble a filesystem?
8. Define DN (Distinguished Name) and RDN (Relative Distinguished Name) in LDAP.
9. What is the purpose of the LDAP schema?
10. What are atomic operations in LDAP, and why are they important?
11. Describe the three concepts of identification, authentication, and authorization.
12. What is the Ticket-Granting Ticket (TGT) in Kerberos?
13. What is a Service Principal Name (SPN) in Kerberos?
14. Why is time synchronization important in Kerberos?
15. What is Kerberoasting?
16. What is the purpose of SRV records in a Windows domain?
17. Why might a host have A records but no PTR records?
18. Why does Active Directory require DNS to function correctly?
19. Use `Resolve-DnsName` to figure out who (which servers from which company) accepts email you send to the domain `vg.no`
20. Consider the following session in PowerShell which shows two identical command lines executed with just a few seconds in between them (`1.1.1.1` is Cloudflare's public DNS server):

```
PS> Resolve-DnsName -Server 1.1.1.1 -Type A facebook.com

Name                   Type   TTL   Section    IPAddress
----                   ----   ---   -------    ---------
facebook.com           A      244   Answer     31.13.72.36

PS> Resolve-DnsName -Server 1.1.1.1 -Type A facebook.com

Name                   Type   TTL   Section    IPAddress
----                   ----   ---   -------    ---------
facebook.com           A      0     Answer     157.240.194.35
```

Why have the values for the properties TTL and IPAddress changed? Explain and comment as detailed as you can.

21. Log in to `login.stud.ntnu.no` with ssh from your laptop. In case you have not used ssh in a while, the syntax is:
`ssh YOUR_NTNU_USERNAME@login.stud.ntnu.no`
Use `ldapsearch` to find yourself (the query should list only you) with the following LDAP filters (hint: see examples in the chapter text with `frode` and `erik`):

    (a) provide your username (the attribute `uid`)

    (b) provide parts of your email address (the attribute `mail`)

    (c) provide parts of your first name (the attribute `givenName`) and parts of your mobile phone number (the attribute `mobile`)

    You should list the attributes `dn` (DistinguishedName), `givenName` (first name) and `sn` (last name). Remember that if an attribute contains norwegian characters (like my surname "Hjelmås"), you can decode the result you get from ldap with (example with encoded "Hjelmås"):
    `echo SGplbG3DpXM= | base64 -d && echo`

22. Expand the command line from the first chapter:

```
Get-NetTCPConnection |
  Select-Object -Property LocalAddress,LocalPort,State,OwningProcess,`
  @{Name='ProcessName';Expression={(Get-Process `
    -Id $_.OwningProcess).ProcessName}},`
  @{Name='ServiceName';Expression={(Get-CimInstance -ClassName `
    Win32_Service -Filter "ProcessID=$($_.OwningProcess)").Name}}
```

    to only show the services NTDS, KDC and DNS. What are the process names of those services?

6

Active Directory: Design and Implementation

In 2017, the company Maersk (shipping) was hit with a cyberattack that caused them a financial loss estimated at close to NOK 2,500,000,000. From Alsinawi [5]:

> Fortunately, Maersk had one small bit of luck. An unplanned power outage kept a single server from getting infected, helping preserve a lone domain controller in an office in Ghana. When the office in Ghana didn't have sufficient bandwidth to synch up the data center over the internet, a relay race was set in motion in which personnel from Ghana frantically met personnel from London in Nigeria! A perfect Ethan Hunt mission.

6.1 NSM Grunnprinsipper

NSM Grunnprinsipper for IKT-sikkerhet 2.0 in figure 6.1. NSM's "Grunnprinsipper for IKT-sikkerhet 2.0" [1] addresses ICT architecture in Chapter 2.2, and Active Directory constitutes a central component of nearly all corporate ICT architectures. Active Directory provides, among other capabilities, services for user identification and authentication, which are discussed in NSM Chapter 2.6.

6.2 Windows Domain

We are familiar with the concept of domains from the Domain Name System (DNS). A *Windows domain*, however, is something different: it is an LDAP-based domain. It is

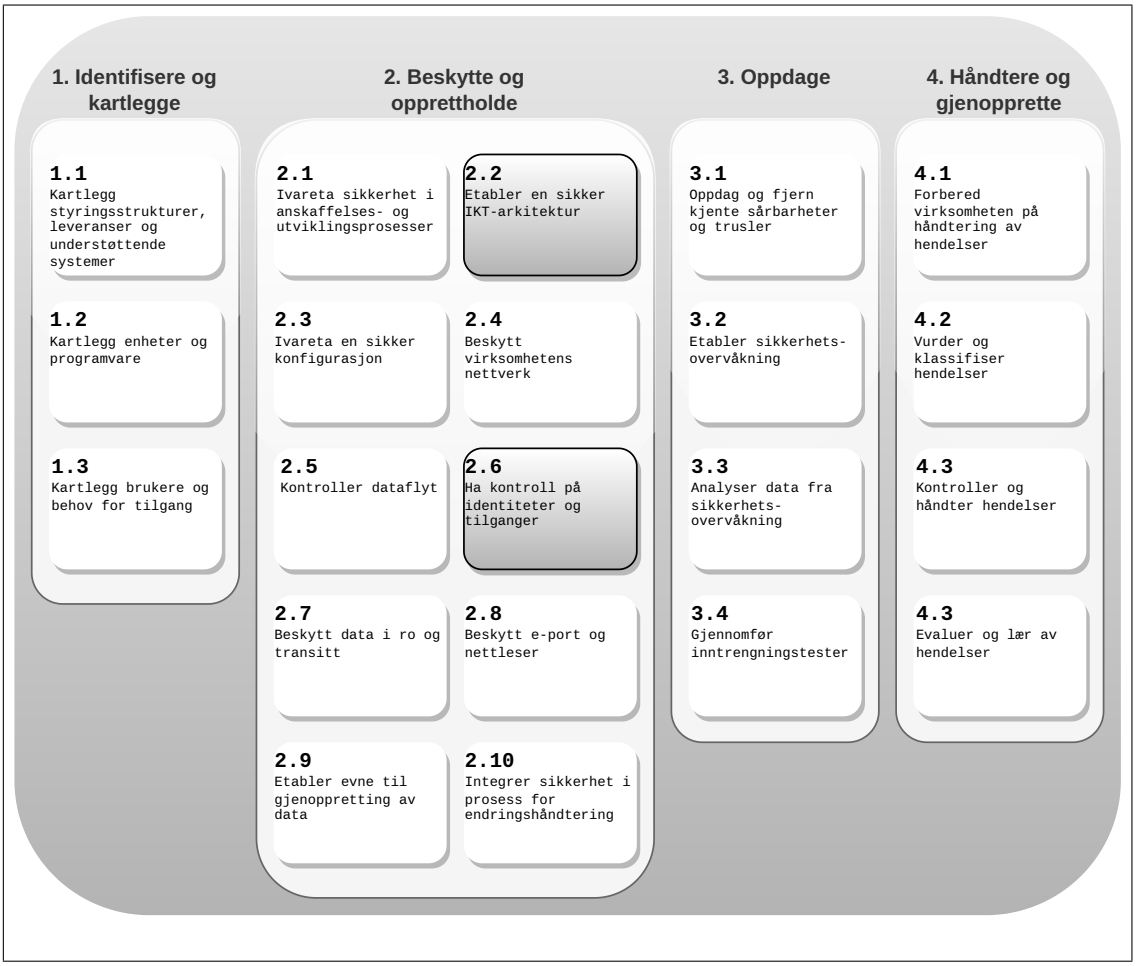


Figure 6.1: NSM Grunnprinsipper for IKT-sikkerhet 2.0.

common practice to use the same name for a Windows domain as for the corresponding DNS domain.

As students and employees at NTNU, we operate within an environment where NTNU holds authority over the DNS domain `ntnu.no`, and the corresponding Windows domain is represented as `dc=ntnu,dc=no`.

A Windows domain is established by installing a *domain controller (DC)*, which is a Windows Server instance running Active Directory as a service (in rare cases, a Linux server using Samba can serve this role). Any medium-sized or large organization will typically have multiple domain controllers—sometimes tens or even hundreds—and these synchronize portions of (or all) directory data periodically to ensure consistency and redundancy.

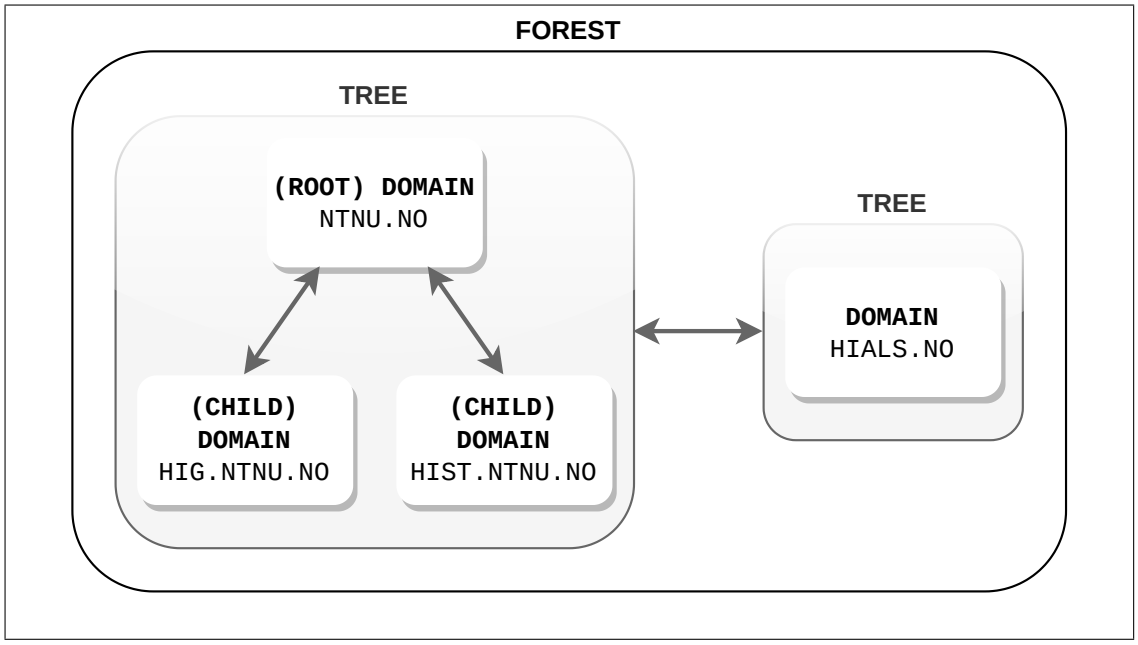


Figure 6.2: Forest, Tree, Domain.

6.2.1 Forest, Tree, Domain

Forest, Tree, Domain in figure 6.2. A domain exists within a *forest*. In the simplest case (such as the one we will use in this course), the forest contains only a single domain. In larger infrastructures, however, multiple domains may exist, belonging to different *trees* within the same forest.

For example, when HiST, HiG, and HiALS were incorporated into NTNU in 2016, their IT infrastructures had to be unified. The author does not know the exact technical details of this process, but for practical reasons it may have been appropriate to integrate Høgskolen i Sør-Trøndelag (HiST) and Høgskolen i Gjøvik (HiG) as domains *within the same namespace (`ntnu.no`) as NTNU*, forming a common domain *tree*. Høgskolen i Ålesund (HiALS), on the other hand, may have had an infrastructure that was more difficult to merge directly, making it more suitable to integrate as a separate tree within the same forest.

What we need to understand about the differences between domains, trees, and forests is the following:

Forest is the *security boundary*. All domains within a forest share configuration data, the schema, the global catalog, and mutual trust relationships. The *global catalog* contains a partial replica of every object in the forest, storing only a selected subset of attributes. This enables fast searches across the entire forest (e.g., for users), regardless of which domain controller handles the request.

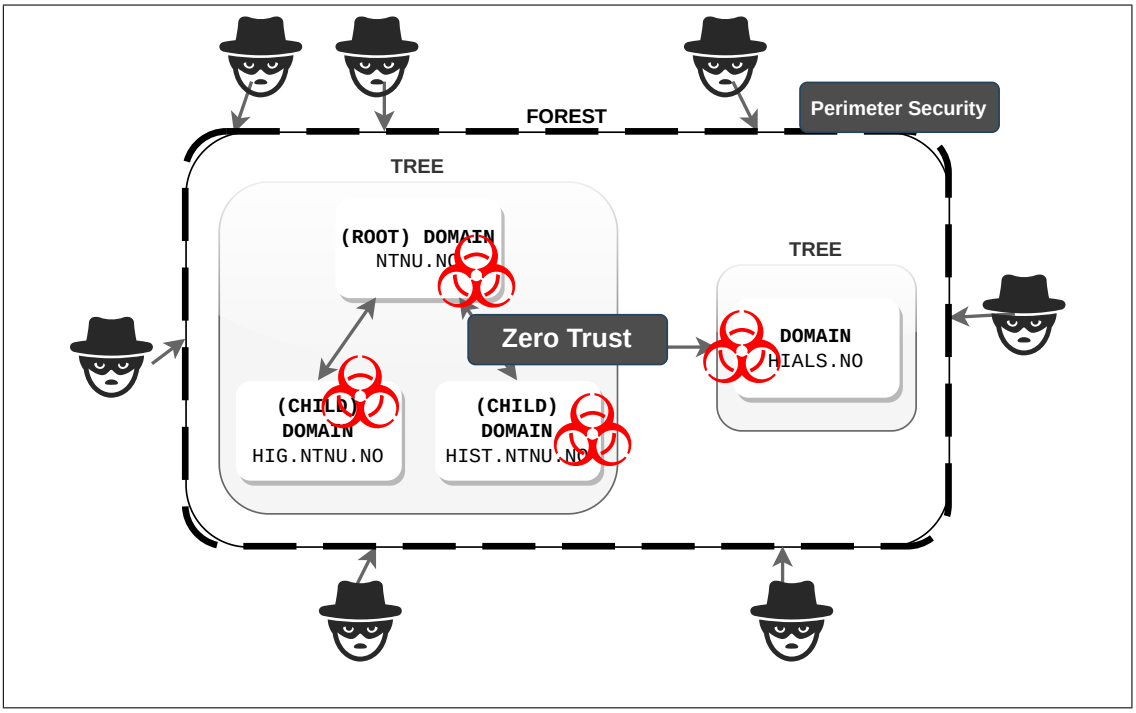


Figure 6.3: Perimeter Security vs Zero Trust.

Tree is not a particularly critical concept in practice. It simply refers to a collection of domains that share the same namespace (i.e., they have the same parent domain name).

Domain is where all directory objects (users, computers, groups, etc.) are stored. All domain controllers within a domain replicate their data to one another, meaning each domain controller holds the same directory information for that domain. Note, however, that a few attributes are *not* replicated [35], such as a user's `lastLogon`.

A company may even operate multiple forests and establish trust relationships between them, although we will not cover multi-forest configurations in this course. In general, we can think of all domain controllers as "equals," but some controllers hold special roles that govern how the most critical directory data is handled during replication. We will not examine these roles in detail here, but interested readers can consult Microsoft's documentation, "Active Directory FSMO roles in Windows" [32].

6.2.2 Thinking Security

Perimeter Security vs Zero Trust in figure 6.3. You have just learned that the Active Directory forest defines the security boundary. This is correct, but whenever we talk about

- Identity and access management
- Asset management
- Policy-based configuration management

Scales to very large organizations

Figure 6.4: Why Active Directory.

a security boundary, we must remember that it is only one component of a broader security model. Protecting the boundary (or perimeter) is important, but given the current and future threat landscape, we must also think in terms of a *Zero Trust* model. In principle, Zero Trust means that no device inside the network is implicitly trusted.

Zero Trust thinking is easiest to adopt if we imagine removing the perimeter entirely and instead operating with *perimeterless security* [72]. Implementing true Zero Trust is, of course, not fully achievable in practice, but the mindset is essential: we should aim to trust no device by default, regardless of its network location.

A simple example illustrates this: a user clicks on an email attachment that downloads and executes malware. Emails are allowed through the perimeter into the internal network. With thousands of employees, some user will eventually run malicious content. The key question then becomes: how can the malware propagate within the internal network? Zero Trust is the generalization of that question—do not assume that any internal device is trustworthy, and ideally remove the notion of "our network" altogether, moving toward a perimeterless infrastructure.

6.2.3 Purpose

Why Active Directory in figure 6.4. Active Directory can function as a shared, enterprise-wide directory service for a multinational company with thousands of employees and hundreds of locations across the world. It stores all IT resources as uniquely identifiable objects (such as users, groups, computers, network devices, and printers) and provides mechanisms for access control and configuration management for these objects. It is a critical component of the IT infrastructure in most medium to large organizations, and it is essential for security professionals to understand how it works and how to manage it effectively.

6.3 Implementing Active Directory

6.3.1 A Case

Organizational Chart in figure 6.5. SEC.CORE is a security consultancy company of-

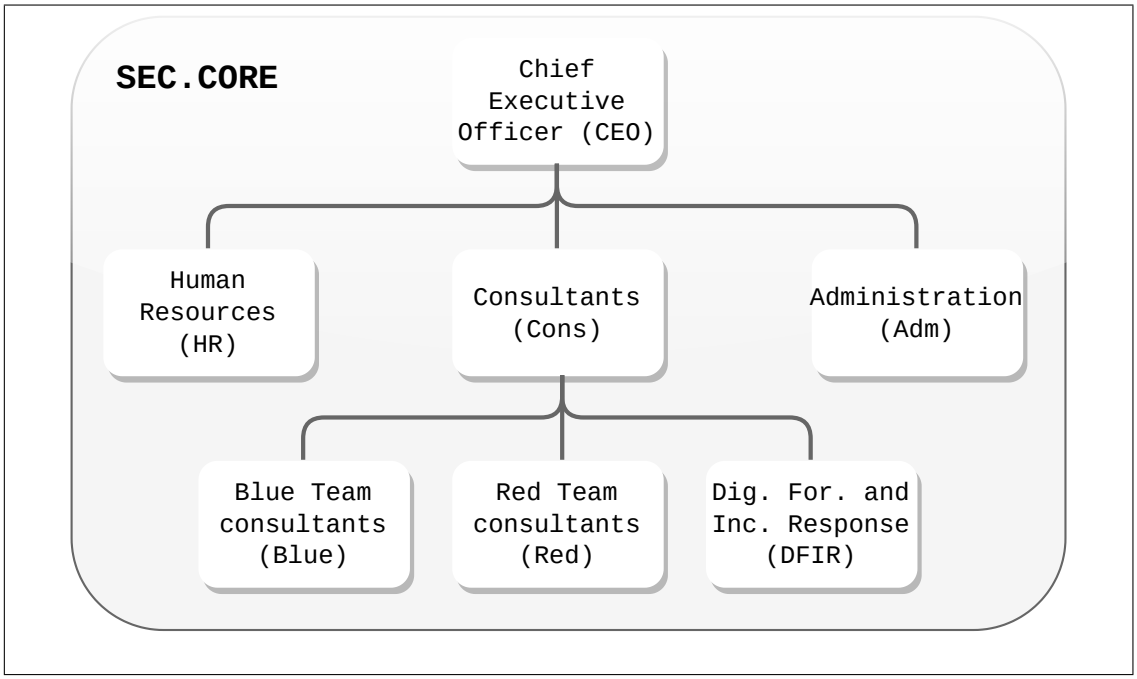


Figure 6.5: Organizational Chart.

fering security testing services in both blue-team and red-team operations, as well as investigation services in the areas of digital forensics and incident response (DFIR). In addition, the company maintains a human resources (HR) department and a general administration unit, which includes the CEO and two system and network administrators (the IT staff).

SEC.CORE hosts all of its general IT services (email, web, accounting, etc.) in a public cloud. As a result, aside from the domain controller `DC1`, the only on-premises server is `SRV1`. The IT staff manage company IT resources using the workstation `MGR` (which should be configured similarly to laptops used by administration staff), while `CL1` is a laptop used by the consultants.

6.3.2 AD ObjectClasses

Important ObjectClasses in figure 6.6. In the Active Directory *schema*, all ObjectClasses are formally defined [34], and we create instances (objects) based on these definitions. For organizing objects, we primarily create *organizational units (OUs)*. If users or computers are joined to a domain without specifying the target OU, they are placed into the default containers `users` and `computers`. Once an OU infrastructure has been established, these objects should be moved into the appropriate OUs.

Default Setup of Users and OUs in figure 6.7. After the installation of Active Directory, the default setup includes the following:

OU organizational units

Container similar to an OU, but cannot have Group Policy Objects

Group contains users and is used for access control (a group has a SID, whereas OUs do not)

In addition, note the root directory server agent service entry ("root of the directory information tree"), *RootDSE*, which is not an ObjectClass.

Figure 6.6: Important ObjectClasses.

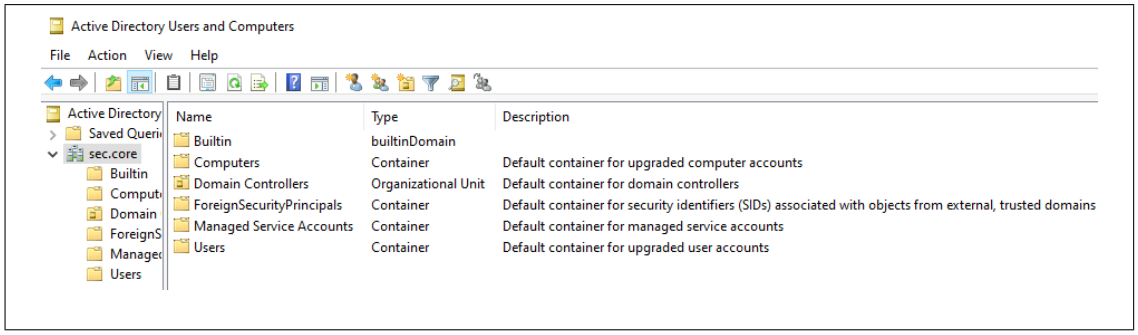


Figure 6.7: Default Setup of Users and OUs.

Domain container the root container of the directory hierarchy

Built-in container contains the default service administrator group accounts

Users container the default location for newly created user accounts and groups

Computers container the default location for newly joined computer accounts

Domain Controllers OU the default location for the computer accounts of domain controllers (i.e., computers serving as domain controllers)

ForeignSecurityPrincipals empty by default; used for security principals from trusted external domains

Managed Service Accounts empty by default; used for managed service account objects

Default Setup of Users and OUs in figure 6.8. Note that for `Users` and `Computers`, `CN` is short for Container, whereas in other contexts `CN` means Common Name (which is unfortunately a bit confusing). When we install Active Directory on `DC1`, the server is registered as a computer account in the only OU that exists by default: the *Domain Controllers* OU. In the figure, we see that the workstation `MGR` has been joined to the domain, and since no specific OU was defined, it is placed in the default `Computers`

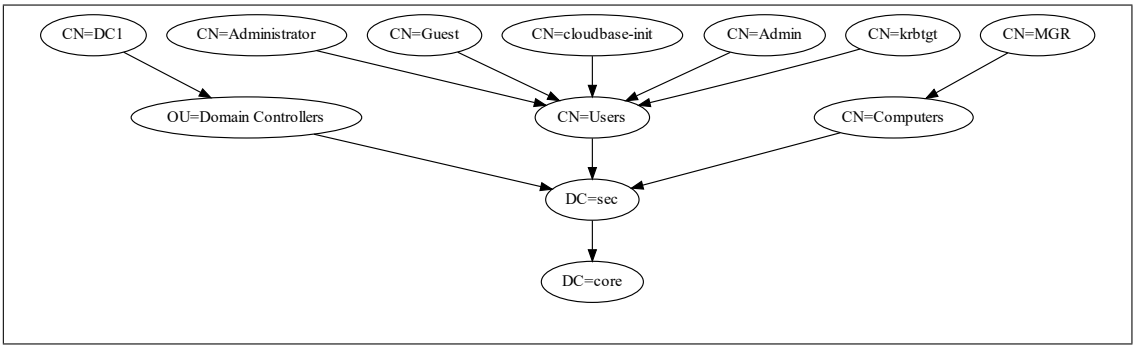


Figure 6.8: Default Setup of Users and OUs.

container (from which it should later be moved, as we will discuss). We also have five user accounts created by default:

Administrator This is the local Administrator account, which becomes a Domain Administrator account after Active Directory is installed.

Guest The built-in Guest account, which (hopefully) remains disabled.

cloudbase-init An account present in Windows cloud instances, used during provisioning (i.e., preparing the system for use).

Admin The default administrative account created automatically in Windows cloud instances (created by Cloudbase).

krbtgt A special service account used by the Kerberos Key Distribution Center (KDC). It is created automatically during the installation of Active Directory.

How does this relate to the local user accounts we discussed earlier in the course? On the domain controller `DC1`, local accounts are migrated into Active Directory at the time AD is installed. (Every other host that later joins the domain will still keep its own local accounts.) Before installing Active Directory, the system looks like this:

```
PS> Get-LocalUser | Select-Object -Property Name,PrincipalSource

Name               PrincipalSource
----               ---------------
Admin                        Local
Administrator                Local
cloudbase-init               Local
DefaultAccount               Local
Guest                        Local
WDAGUtilityAccount           Local
```

After Active Directory has been installed, it looks like this:

```
PS> Get-LocalUser | Select-Object -Property Name,PrincipalSource

Name           PrincipalSource
----           ---------------
Administrator  ActiveDirectory
Guest          ActiveDirectory
krbtgt         ActiveDirectory
cloudbase-init ActiveDirectory
Admin          ActiveDirectory
DC1$           ActiveDirectory
```

In practice, this means that we can log in to any domain-joined host using an account that exists in Active Directory. However, we can also log in to domain-joined hosts using their local user accounts. In other words, when logging in, we must explicitly choose the type of authentication we want to use. You may have encountered this before: you are presented with a choice of domains to log into, and the local hostname appears as one of the available login options.

All system user and group accounts (`Win32_SystemAccount`) are also moved into Active Directory during the installation. The `$` in the account name `DC1$` indicates that this is a computer account rather than a user account. All computer (machine) accounts follow this naming convention and end with a trailing `$`.

6.3.3 OU Design

OU Design in figure 6.9. In the figure, SEC.CORE's organizational chart has been translated into a possible logical structure in Active Directory using organizational units (OUs). This design follows the best practice [30] of avoiding the placement of users and computers in the same OUs, and of creating dedicated OUs for all objects so that they do not remain in the default `Users` and `Computers` containers. Since SEC.CORE is a small organization, our OU design is based primarily on the company's organizational chart. However, we make two intentional deviations:

1. we separate the IT department, since those user accounts require a different set of permissions, and
2. HR does not need a dedicated OU in a small company and can instead be placed under general administration. (Note: this may become a poor design choice if the company grows or merges with others in the future.)

An alternative approach – common in larger organizations – is to base the OU structure on geographical location (called a *site* in Active Directory), or on a hybrid between organizational structure and geographical distribution.

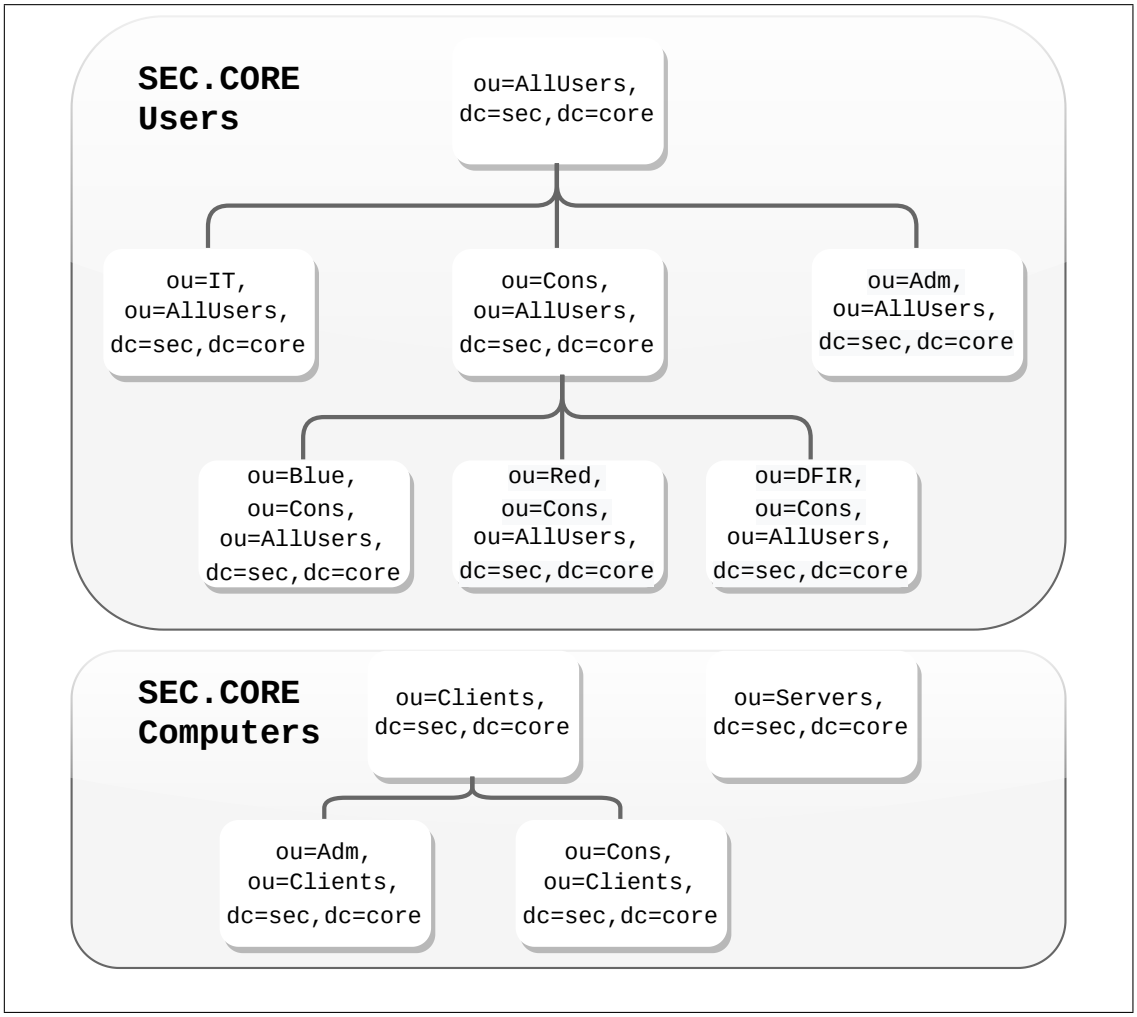


Figure 6.9: OU Design.

The most important aspect of Active Directory design is the structure of Organizational Units (OUs). *Best practices state that OUs should be organized primarily to support administrative delegation, and secondarily to support Group Policy Objects (GPOs)*. In practice, this means that if you have multiple teams of system administrators (as in much larger organizations than SEC.CORE), you should delegate responsibility by allowing each team to administer a defined set of OUs.

This ties directly to the secondary purpose: facilitating GPOs. Users and computers should be grouped in OUs according to which policies need to apply to them. For example, in a university, it is natural to distinguish between students, faculty, and administrative staff. Likewise, computers should be separated—student workstations belong in a different OU from faculty laptops.

Sites in Active Directory represent geographical locations that correspond to distinct physical networks. Sites become relevant when the organization spans multiple physical locations where bandwidth between sites may be limited.

If you have locations that are far apart (for example, a branch office in another country), you may want users at those locations to rely on local services rather than sending all traffic to a remote central site. Active Directory sites are designed specifically for this scenario. They allow you, for instance, to deploy a Read-Only Domain Controller (RODC) at a low-bandwidth site to provide faster local authentication and directory access for users at that location.

6.3.4 Creating the OUs

Let us now create the OUs. If we do not specify the `Path` parameter, the OU will be created at the top level, under `DC=sec,DC=core`.

```
# User OUs (logged in as Domain Administrator on DC1)
New-ADOrganizationalUnit 'AllUsers' -Description 'Contains OUs and users'
New-ADOrganizationalUnit 'IT' -Description 'IT staff' `
  -Path 'OU=AllUsers,DC=sec,DC=core'
New-ADOrganizationalUnit 'Cons' -Description 'Consultants' `
  -Path 'OU=AllUsers,DC=sec,DC=core'
New-ADOrganizationalUnit 'Adm' -Description 'Administration' `
  -Path 'OU=AllUsers,DC=sec,DC=core'
New-ADOrganizationalUnit 'Blue' -Description 'Blue Team' `
  -Path 'OU=Cons,OU=AllUsers,DC=sec,DC=core'
New-ADOrganizationalUnit 'Red' -Description 'Red Team' `
  -Path 'OU=Cons,OU=AllUsers,DC=sec,DC=core'
New-ADOrganizationalUnit 'DFIR' `
  -Description 'Digital Forensics and Incident Response' `
  -Path 'OU=Cons,OU=AllUsers,DC=sec,DC=core'
# Computer OUs
```

```
New-ADOrganizationalUnit 'Clients' `
  -Description 'Contains OUs and users laptops'
New-ADOrganizationalUnit 'Servers' `
  -Description 'Contains OUs and servers'
New-ADOrganizationalUnit 'Adm' -Description 'Adm laptops' `
  -Path 'OU=Clients,DC=sec,DC=core'
New-ADOrganizationalUnit 'Cons' -Description 'Consultants laptops' `
  -Path 'OU=Clients,DC=sec,DC=core'
```

6.3.5 Joining Computers

When joining computers to the domain, we can either join them without specifying a `Path`, in which case the computer account will be placed in the default `Computers` container and must later be moved to the correct OU; or we can specify the path to the desired OU directly when joining, ensuring the computer account is created in the appropriate location in Active Directory from the start.

```
# In our case, all hosts were joined without specifying path
# so we need to move them:
Get-ADComputer "MGR" |
  Move-ADObject -TargetPath "OU=Adm,OU=Clients,DC=sec,DC=core"
Get-ADComputer "CL1" |
  Move-ADObject -TargetPath "OU=Cons,OU=Clients,DC=sec,DC=core"
Get-ADComputer "SRV1" |
  Move-ADObject -TargetPath "OU=Servers,DC=sec,DC=core"

# If e.g. we had not joined cl1 yet, we could have joined it
# directly to its correct OU with
# Add-Computer -Credential $cred -DomainName sec.core `
#   -OUPath 'OU=Cons,OU=Clients,DC=sec,DC=core' -PassThru -Verbose
#
# Manual for Add-Computer states it is smart to use -PassThru:
# "Returns an object representing the item with which you are working.
# By default, this cmdlet does not generate any output."

# Let's check status, query via dedicated Cmdlet
Get-ADComputer -Filter * |
  Select-Object -Property DNSHostName,DistinguishedName

# Same query via generic LDAP Cmdlet
Get-ADObject -LDAPfilter "(ObjectClass=Computer)" |
  Select-Object -Property DNSHostName,DistinguishedName
# notice missing DNSHostName because cmdlets return different objects
```

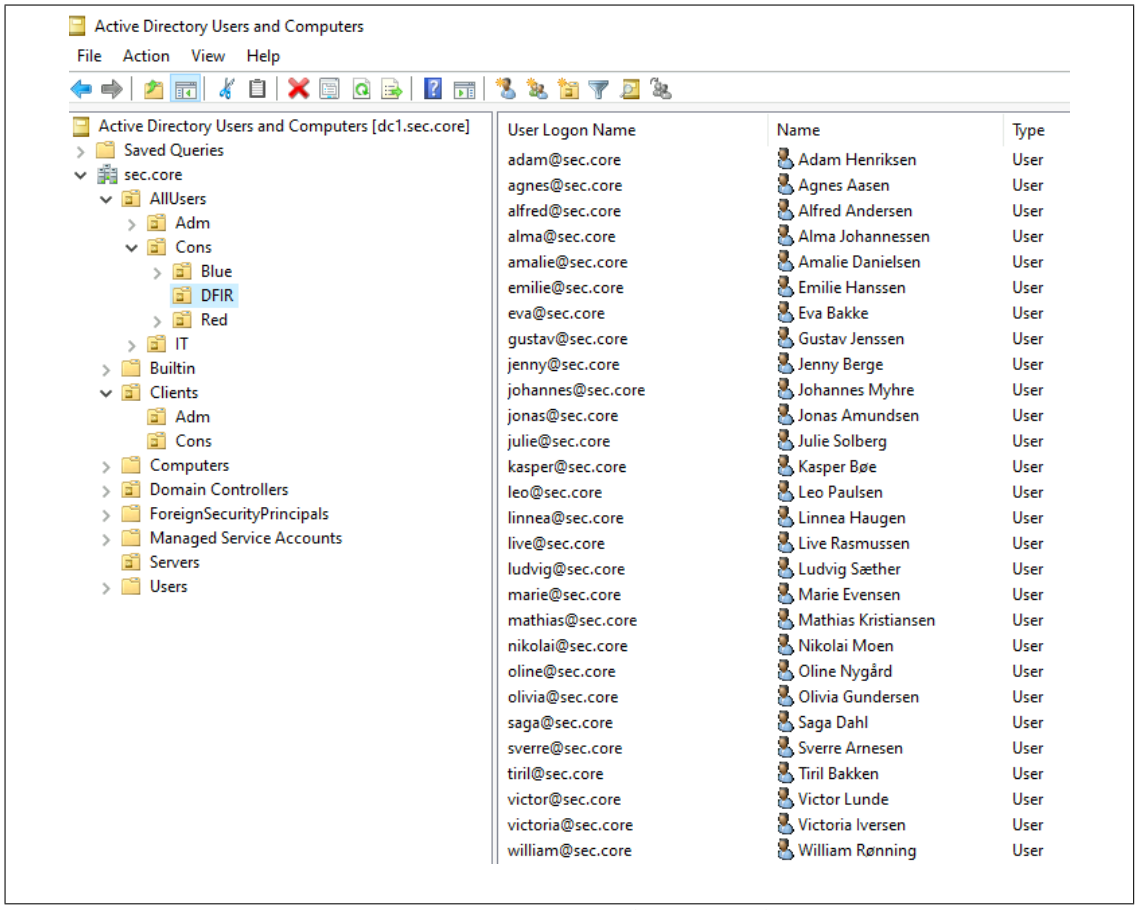


Figure 6.10: SEC.CORE with OUs and Users.

6.3.6 Adding Users

SEC.CORE with OUs and Users in figure 6.10. We want to create a set of random users without any default passwords and place them in the appropriate OUs. For more information, see the introduction of the `CreateUserCSV.ps1` file.

```
# Lets create the users
# if you don't have curl, do choco install curl
# and join these to lines into one URL:
curl -O https://raw.git.ntnu.no/erikhje/sky/refs/heads/
  main/scripts/CreateUserCSV.ps1
.\CreateUserCSV.ps1
$ADUsers = Import-Csv seccoreusers.csv -Delimiter ';'
foreach ($User in $ADUsers) {
  if (!(Get-ADUser -LDAPFilter `
      "(sAMAccountName=$($User.Username))")) {
```

```
    New-ADUser `
    -SamAccountName         $User.Username `
    -UserPrincipalName      $User.UserPrincipalName `
    -Name                   $User.DisplayName `
    -GivenName              $User.GivenName `
    -Surname                $User.SurName `
    -Enabled                $True `
    -ChangePasswordAtLogon $False `
    -DisplayName            $user.Displayname `
    -Department             $user.Department `
    -Path                   $user.path `
    -AccountPassword `
    (ConvertTo-SecureString $user.Password -AsPlainText -Force)
  }
}
# How many users do we have now?
(Get-ADUser -Filter * | Measure-Object).Count
# View all attributes of a user:
Get-ADObject -Filter { SamAccountName -eq 'Jenny' } -Properties *
```

6.3.7 Groups

The AGDLP Pattern in figure 6.11. Access control is hard. In the Unix/Linux world, the standard model is relatively simple: owner, group, and others, combined with read, write, and execute permissions. In the Windows world, the set of possible permissions is much larger, and – notably – objects are not limited to files as they are in Unix/Linux. With more permission types and many kinds of objects, the overall complexity grows, and complexity is always our enemy.

In Active Directory, *it is considered bad practice to assign permissions directly to user accounts*. Doing so quickly becomes difficult to manage and nearly impossible to maintain an overview of. To implement access control correctly, we must think in terms of groups and roles. This gives us a higher-level abstraction that allows us to manage permissions effectively.

> *Note that this is how we "always" solve complex problems in computer science: we manage complexity through hierarchy, layers, and abstractions (i.e., we decompose a large complex problem into smaller, simpler subproblems).*

Active Directory has the following types of groups [33]:

Distribution groups used only for mailing lists (not directly relevant for our purposes).

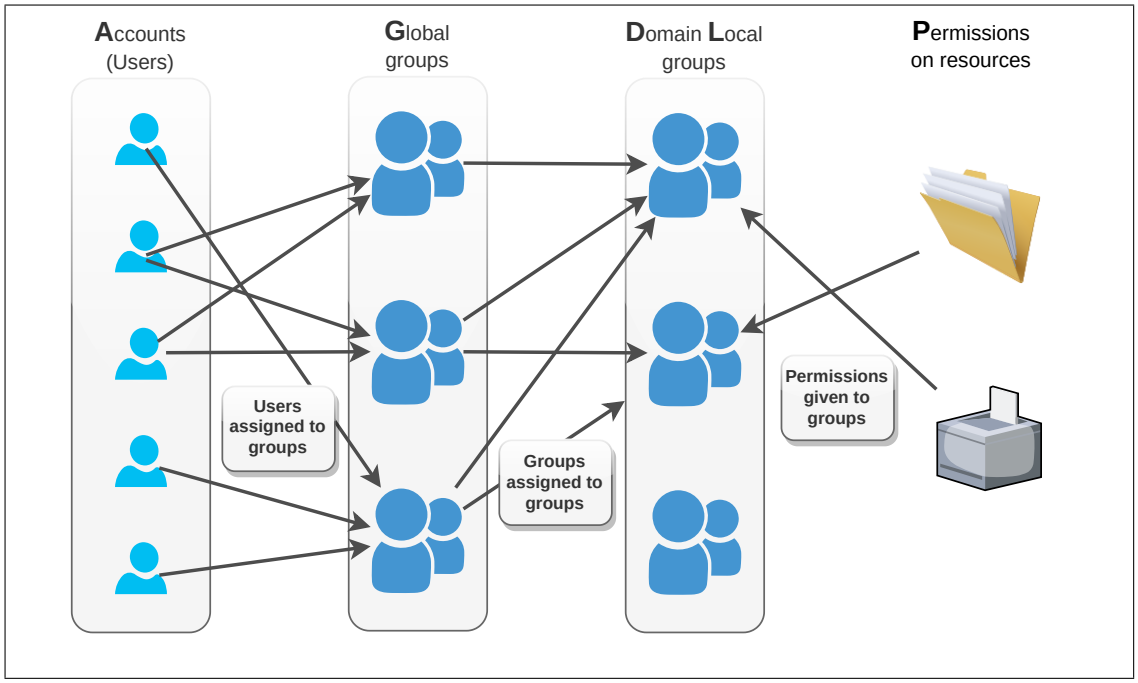


Figure 6.11: The AGDLP Pattern.

Security groups used for assigning permissions and roles ("grouped permissions"), with different *scopes*:

Universal Forest-wide scope.

Global Domain-wide scope.

Domain local Domain-wide scope, but cannot be members of "built-in groups that have well-known SIDs" [51].

The standard pattern for setting up access control in Active Directory is called AGDLP, an acronym for Accounts, Global groups, Domain Local groups, and Permissions.

Typically, we create a global group corresponding to each OU containing user accounts. The user accounts from the OU are added to this global group. We cannot perform access control directly on OUs because OUs do not have SIDs (Security Identifiers) and therefore are not *security principals*.

Computer accounts, user accounts, and groups *are* security principals (they have SIDs) and can therefore be used in access control lists. As mentioned, we create groups based on each OU – usually placing them inside the OU – and then assign the OU's users to these groups.

For each resource that requires controlled access, we create domain local groups and assign permissions to those groups. Using the AGDLP pattern, we then add the global

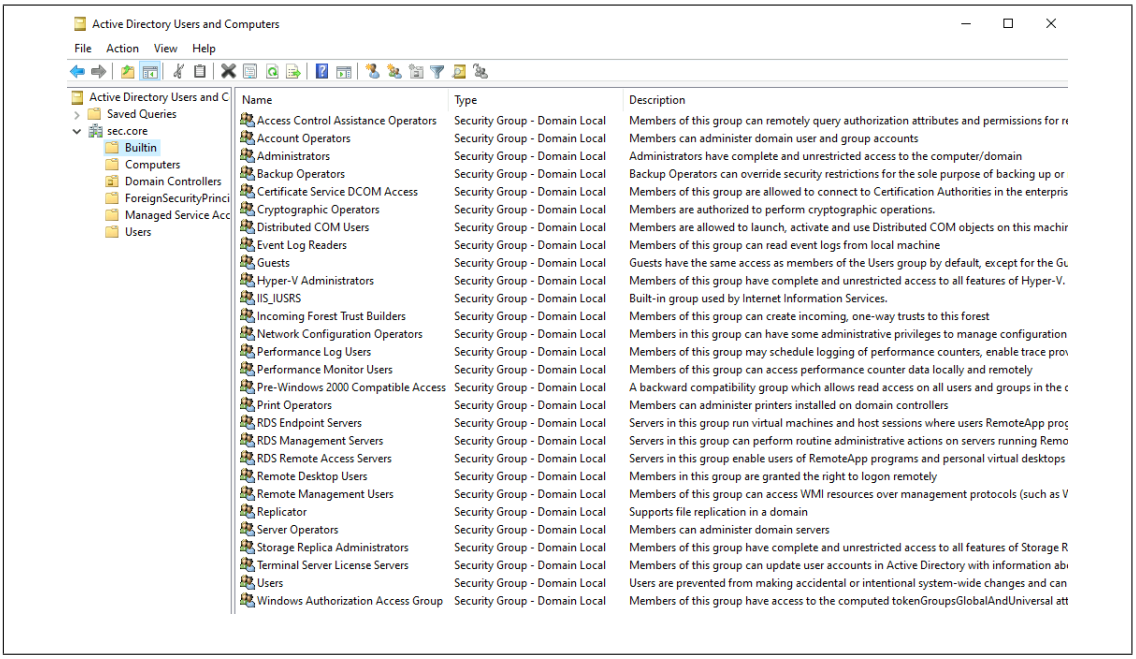


Figure 6.12: Default Domain Local Groups in Builtin .

groups to the appropriate domain local groups. This creates a manageable and clear structure for roles and permissions.

However, this setup is not dynamic: if a new user account is added to an OU, it is not automatically added to the corresponding global group. Whenever new users are created in the domain, proper routines must be in place to update the global groups accordingly.

Default Domain Local Groups in Builtin in figure 6.12. Active Directory includes a set of predefined domain local groups located in the `Builtin` container. These groups represent common administrative roles within a domain. By adding users to these groups, we can assign them the corresponding administrative capabilities (for example, IT staff working with troubleshooting or incident management should be added to the `Event Log Readers` group). However, these groups are not designed for general access control purposes. They are intended for specific administrative functions and should not be used as part of the AGDLP pattern for managing permissions on resources. Instead, we should create our own domain local groups for access control and add the appropriate global groups to them as needed.

Default Groups and Users in Users in figure 6.13. Active Directory also includes a set of predefined universal, global, domain local groups, as well as predefined user accounts, all located in the `Users` container. These groups represent common roles within a domain, similar to the groups found in the `Builtin` container. However, unlike the `Builtin` container, the `Users` container includes universal groups, global groups, and user accounts in addition to domain local groups. Regarding the groups and users in

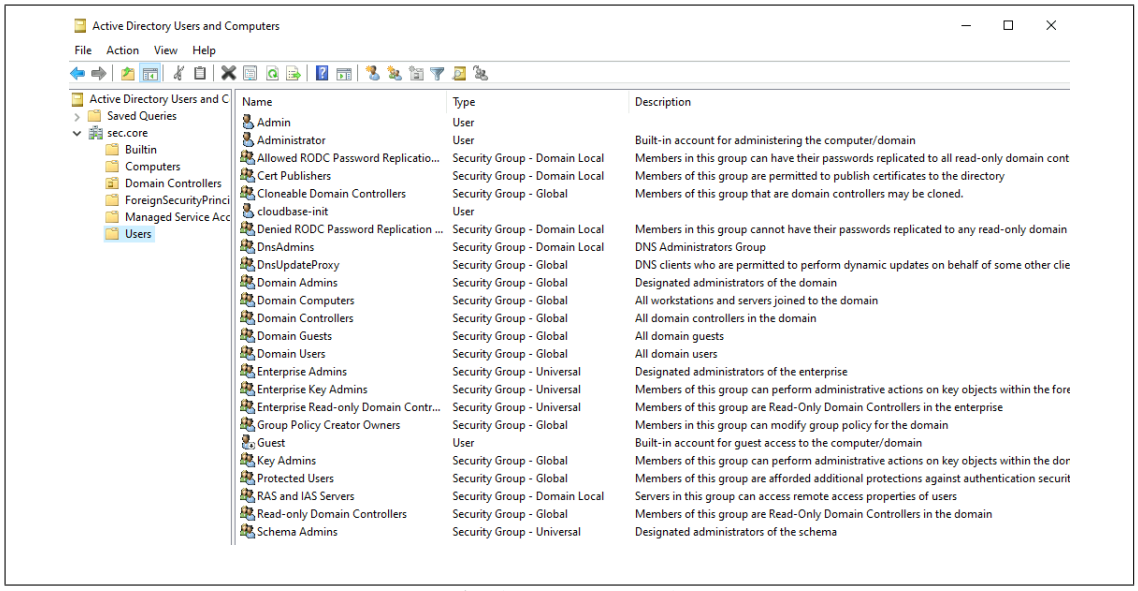


Figure 6.13: Default Groups and Users in Users.

both `Builtin` and `Users`, Microsoft states [33]:

> You can move groups that are located in these containers to other groups or organizational units (OU) within the domain, but you cannot move them to other domains.

RDP group in figure 6.14. By default, only domain administrators are allowed to use RDP (connect to a host using Remote Desktop) on domain-joined machines. If additional users should be granted RDP access, they must be added to the local group `Remote Desktop Users` on the specific host they need to access.

In our case, we want to allow a specific user – for example, `julie` – to RDP into the host `CL1`. Because the local group `Remote Desktop Users` on `CL1` is one of the built-in groups that have well-known SIDs [51], we cannot simply add `julie` to the `Remote Desktop Users` group in Active Directory. The group with that name in Active Directory is a domain local group, which is not the same as the local group that exists only on the `CL1` machine.

To follow the AGDLP pattern (or more precisely, a slightly simplified "AGLP" variant in this scenario), we must:

1. Create a new global group in Active Directory.
2. Add `julie` to this global group.
3. Add the global group to the local `Remote Desktop Users` group on `CL1`.

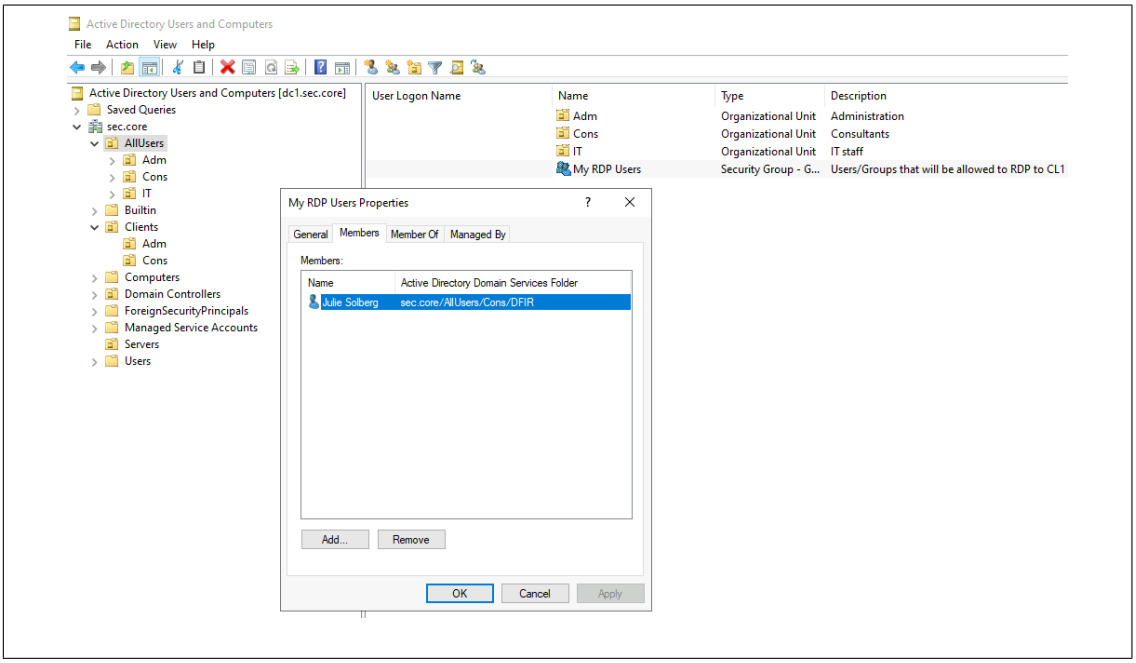


Figure 6.14: RDP group.

This approach keeps privilege assignments manageable, avoids assigning permissions directly to users, and maintains a clear group-based access control structure.

```
# on DC1 (logged in as Domain Administrator)
$GROUP = @{
  Name           = "g_My RDP Users"
  GroupCategory = "Security"
  GroupScope     = "Global"
  DisplayName    = "Local RDP users"
  Path           = "OU=AllUsers,DC=sec,DC=core"
  Description    = "Users/Groups that will be allowed to RDP to CL1"
}
New-ADGroup @GROUP
Add-ADGroupMember -Identity 'g_My RDP Users' -Members 'Julie'
Get-ADGroupMember -Identity 'g_My RDP Users'

# on CL1 ("Run as administrator")
Add-LocalGroupMember -Group "Remote Desktop Users" `
  -Member 'SEC.CORE\g_My RDP Users'
Get-LocalGroupMember -Group "Remote Desktop Users"
```

We prefix all global groups with g_ to make it easy to identify global groups solely by their names.

6.4 Lab tutorials

1. Add OUs, Users, Groups to your sec.core domain (assuming you installed the domain in last weeks exercise) and use the AGDLP pattern to allow selected users to RDP to CL1 (like `Julie`), all according to the description in the chapter text. In other words, implement what you have read in this chapter.

2. Mount Active Directory as a PSDrive and navigate around in it:

```
Import-Module ActiveDirectory
Get-PSDrive
cd AD:\
cd 'DC=sec,DC=core'
cd OU=AllUsers
Get-ChildItem
cd OU=Adm
Get-ChildItem
cd ../..
Get-ChildItem
```

6.5 Review questions and problems

1. What is the key difference between a DNS domain and a Windows domain?
2. Why is a forest considered the security boundary in Active Directory?
3. What is the role of the global catalog?
4. What distinguishes a tree from a domain in AD?
5. Why does Zero Trust remain important even if the forest is a security boundary?
6. Why do organizations deploy multiple domain controllers?
7. What are Organizational Units (OUs) used for?
8. Why shouldn't users and computers remain in the default containers?
9. How can we recognize a computer account in AD?
10. Why is OU design considered crucial?
11. What is an AD site?
12. Why is assigning permissions directly to user accounts discouraged?
13. What are the two main AD group types?
14. What does AGDLP stand for?
15. Why can't OUs be used directly for access control?
16. Why must a global group be created to give Julie RDP access to CL1?
17. What is the purpose of prefixing global groups (e.g., g_)?
18. Do the lab tutorial before you try to solve this problem.
    (a) Use `Get-ADUser -Filter` to find all user accounts that contain the letter 'M' in the `SamAccountName`. Pipe to `Format-Table` and print only `SamAccountName` and `Name`.
    (b) Use `Get-ADUser -LDAPFilter` to find all user accounts that contain the letter 'M' in the `SamAccountName`. Pipe to `Format-Table` and print only `SamAccountName` and `Name`.
    (c) Without piping to `Format-Table`, measure how much time each of the two different commands use with `Measure-Command`. Is one of them faster than the other?
19. Do the lab tutorial before you try to solve this problem.

(a) Delete all the users in the OU `DFIR`.

(b) Remove the OU `DFIR` with

```
Remove-ADOrganizationalUnit -Identity `
  'OU=DFIR,OU=Cons,OU=AllUsers,DC=sec,DC=core'
```

(hint: you will not be allowed, search the internet to find our how what you need to do to remove an OU)

(c) Create the OU DFIR again.

(d) Create the users again by modifying the `foreach` loop to you used in the lab tutorial to create them (modify the loop to only loop over the users in the OU DFIR).

7

Remoting, Config Management and Group Policy

Configuration Management is about *centrally controlling and maintaining the configuration ("settings") of many hosts*. A wide range of approaches and tools exist for this purpose, including Cfengine, Puppet, and Ansible. In the context of Windows infrastructures, however, the most widely used and enduring technology is Group Policy. In this chapter, we focus primarily on Group Policy as the core configuration management mechanism in Windows environments. We also briefly discuss alternative approaches and complementary technologies, with a particular emphasis on PowerShell Remoting.

NSM Grunnprinsipper for IKT-sikkerhet 2.1 in figure 7.1. Configuration management is the basis for several basic security measures, so the relevant chapters in NSM is at least 2.3, 2.6 and 2.8.

7.1 Push vs Pull

Push vs Pull in figure 7.2. There are two major approaches to controlling the configuration of numerous hosts from a central server [70]:

Push refers to situations where the central server initiates the connection to the managed hosts. The server actively "pushes" configuration changes to them. Examples of push-based systems include PowerShell Remoting and SSH, where the management server connects directly to each host to execute commands or apply settings.

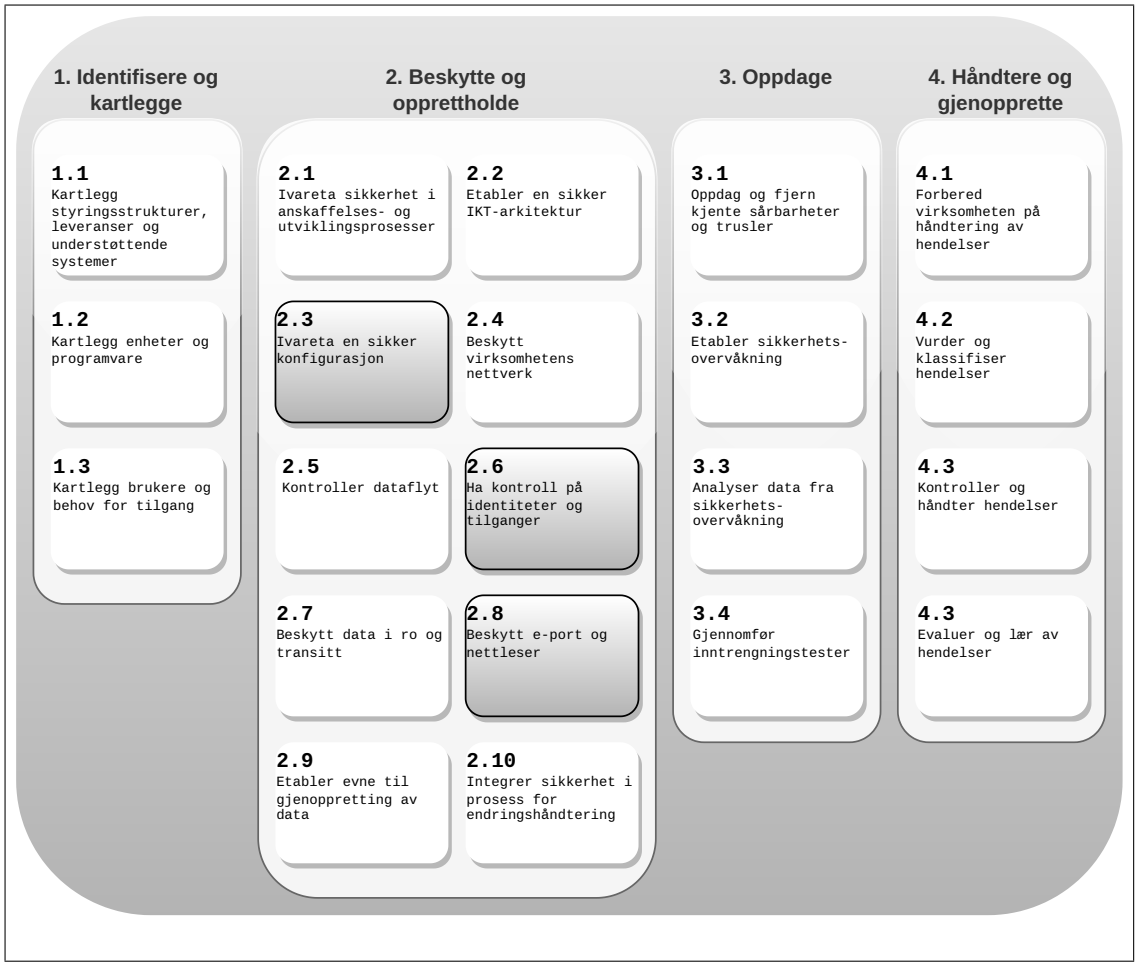


Figure 7.1: NSM Grunnprinsipper for IKT-sikkerhet 2.1.

Pull refers to systems where the hosts themselves autonomously contact the central server to retrieve configuration data. They download the configuration and apply it according to their own schedule or triggering conditions. Examples of pull-based approaches include Group Policy and PowerShell Desired State Configuration (DSC).

Different configuration management technologies rely on different network ports and communication channels. In some environments, these systems operate on dedicated and trusted management networks ("management LANs"). However, even in such setups, a strict Zero Trust security model (as introduced in earlier chapters) requires that no host be inherently trusted. Instead, all communication should be mutually authenticated to ensure that only legitimate clients and servers exchange configuration data.

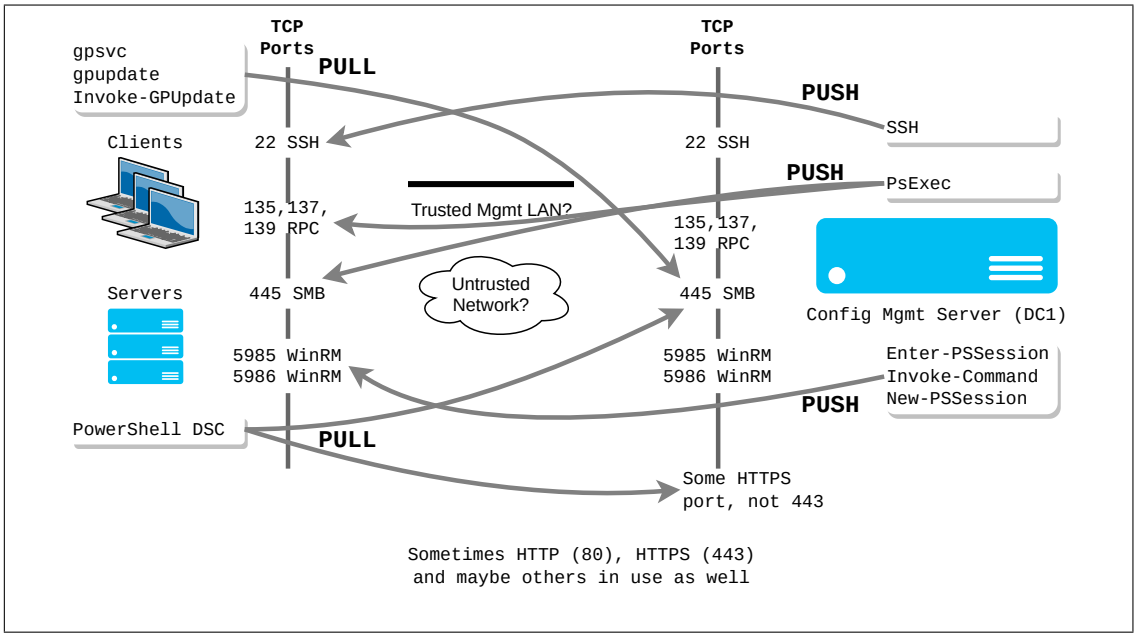


Figure 7.2: Push vs Pull.

7.2 PowerShell Remoting

In PowerShell, we can connect to other hosts using either WSMan or SSH as the underlying protocol. (Some cmdlets can also use DCOM/WMI/RPC (port 135) for remoting, but we will not cover those methods here.) SSH must be used if we want to perform PowerShell remoting to non-Windows hosts. In our environment, we use PowerShell remoting over the WSMan protocol (also known as Windows Remote Management, or WinRM).

Using PowerShell remoting is straightforward as long as all hosts belong to the same domain. Connections to hosts outside the domain—or between two hosts that are not domain-joined—are also possible, but require some initial configuration to establish trust and enable remote communication.

There are three standard ways to use PowerShell remoting (the examples below show MGR connecting to DC1, where both machines are in the same domain):

1. An interactive session:

```
Enter-PSSession DC1
```

2. Executing a set of commands on a set of remote host:

```
Invoke-Command -ComputerName DC1,SRV1 `
```

```
{Get-LocalUser -Name Guest | Select-Object -Property Enabled}
```

3. Establish a persistant session:

```
$s = New-PSSession -ComputerName DC1,SRV1
Invoke-Command -Session $s {Get-HotFix}
Get-PSSession
Remove-PSSession $s
```

Establishing a persistant session can sometimes be faster than using `Invoke-Command`.

7.3 SMB and File Shares

Windows can share parts of its file system using the Server Message Block (SMB) protocol (also known as the Common Internet File System, CIFS). By default, the SMB service listens on port 445. SMB can also support additional functionality beyond simple file sharing, including Remote Command Execution (RCE) when used in combination with Remote Procedure Calls (RPC) on port 135 (and possibly ports 137 and 139). For the purposes of this chapter, however, we focus only on SMB's file-sharing capabilities.

Managing file shares in PowerShell can be accomplished using cmdlets from the `SmbShare` module:

```
Get-Command -Module SmbShare
```

SMB File Shares in figure 7.3. File shares can be either *hidden* or visible. A hidden share is not displayed in Windows Explorer or in "My Computer," although it can still be easily discovered using other tools. Hidden shares are indicated by appending a `'$'` to the end of the share name. The `ADMIN$`, `C$`, and `IPC$` shares are administrative shares that exist on all Windows hosts. The `NETLOGON` and `SYSVOL` shares are created automatically on domain controllers.

To access a file share, we do not use a regular file path. Instead, we use a *Universal Naming Convention (UNC)* path [49]. A typical UNC path looks like this:

```
\server\share\filepath
e.g.
Get-ChildItem '\DC1\ADMIN$\System32\calc.exe'
```

Local file shares can be listed using the `Get-SmbShare` cmdlet. To view shares on a remote computer from the command line, we can use the legacy `net view` program, for example:

```
net view \dc1
```

Normal Windows host:

```
[mgr]: PS C:\Users\Administrator\Documents> Get-SmbShare

Name   ScopeName Path       Description
----   --------- ----       -----------
ADMIN$ *         C:\windows Remote Admin
C$     *         C:\        Default share
IPC$   *                    Remote IPC
```

Domain Controller:

```
[dc1]: PS C:\Users\Administrator\Documents> Get-SmbShare

Name     ScopeName Path                                        Description
----     --------- ----                                        -----------
ADMIN$   *         C:\windows                                  Remote Admin
C$       *         C:\                                         Default share
IPC$     *                                                     Remote IPC
NETLOGON *         C:\windows\SYSVOL\sysvol\sec.core\SCRIPTS Logon server share
SYSVOL   *         C:\windows\SYSVOL\sysvol                    Logon server share
```

Figure 7.3: SMB File Shares.

```
[dc1]: PS C:\Users\Administrator\Documents> Get-SmbShare | Get-SmbShareAccess

Name     ScopeName AccountName                    AccessControlType AccessRight
----     --------- -----------                    ----------------- -----------
IPC$     *         BUILTIN\Administrators         Allow             Full
IPC$     *         BUILTIN\Backup Operators       Allow             Full
IPC$     *         NT AUTHORITY\INTERACTIVE       Allow             Full
C$       *         BUILTIN\Administrators         Allow             Full
C$       *         BUILTIN\Backup Operators       Allow             Full
C$       *         NT AUTHORITY\INTERACTIVE       Allow             Full
NETLOGON *         Everyone                       Allow             Read
NETLOGON *         BUILTIN\Administrators         Allow             Full
SYSVOL   *         Everyone                       Allow             Read
SYSVOL   *         BUILTIN\Administrators         Allow             Full
SYSVOL   *         NT AUTHORITY\Authenticated Users Allow           Full
ADMIN$   *         BUILTIN\Administrators         Allow             Full
ADMIN$   *         BUILTIN\Backup Operators       Allow             Full
ADMIN$   *         NT AUTHORITY\INTERACTIVE       Allow             Full
```

Figure 7.4: SMB File Share Access.

SMB File Share Access in figure 7.4. The default administrative shares – `ADMIN$`, `C$`, and `IPC$` – are accessible only to users with local administrator privileges. In a domain environment, this also means that domain administrator accounts have access to these shares by default. For our purposes, the `NETLOGON` and `SYSVOL` shares are the most important:

SYSVOL This is where the group policy objects are.

NETLOGON This is where user logon scripts are (but the share is really just a link to a subfolder of SYSVOL).

```
PS> (Get-SmbShare -Name NETLOGON).Path
C:\windows\SYSVOL\sysvol\sec.core\SCRIPTS
PS> (Get-SmbShare -Name SYSVOL).Path
C:\windows\SYSVOL\sysvol
```

`SYSVOL` and `NETLOGON` both grant `Read` access to `Everyone`. These permissions apply only to the *file share* itself. In addition, users must also have the appropriate permissions on the underlying directory they want to access. In other words, access operates at two levels: 1. Access to the file share, and 2. Access to the directory within that share. Directory access is controlled by the file system's access control lists (ACLs), which can be viewed and modified using `Get-ACL` and `Set-ACL`. On a domain controller, we can observe that although `Everyone` has `Read` access to the `SYSVOL` file share, they do not have permissions to the underlying directory:

```
PS> (Get-Acl C:\Windows\SYSVOL\sysvol).Access |
 Select-Object -Property IdentityReference,AccessControlType,FileSystemRights

IdentityReference          AccessControlType  FileSystemRights
-----------------          -----------------  ----------------
CREATOR OWNER              Allow              -536084480
NT AUTHORITY\Auth. Users   Allow              -1610612736
NT AUTHORITY\Auth. Users   Allow              ReadAndExecute, Synchronize
NT AUTHORITY\SYSTEM        Allow              268435456
NT AUTHORITY\SYSTEM        Allow              FullControl
BUILTIN\Administrators     Allow              -536084480
BUILTIN\Administrators     Allow              Write, ReadAndExecute,
                                              ChangePermissions,
                                              TakeOwnership, Synchronize
BUILTIN\Server Operators   Allow              -1610612736
BUILTIN\Server Operators   Allow              ReadAndExecute, Synchronize
```

(where it says `-536084480`, `268435456` or `-1610612736`, these are *Special* permissions which is a grouping of permissions without any specific name, they are just called "special".)

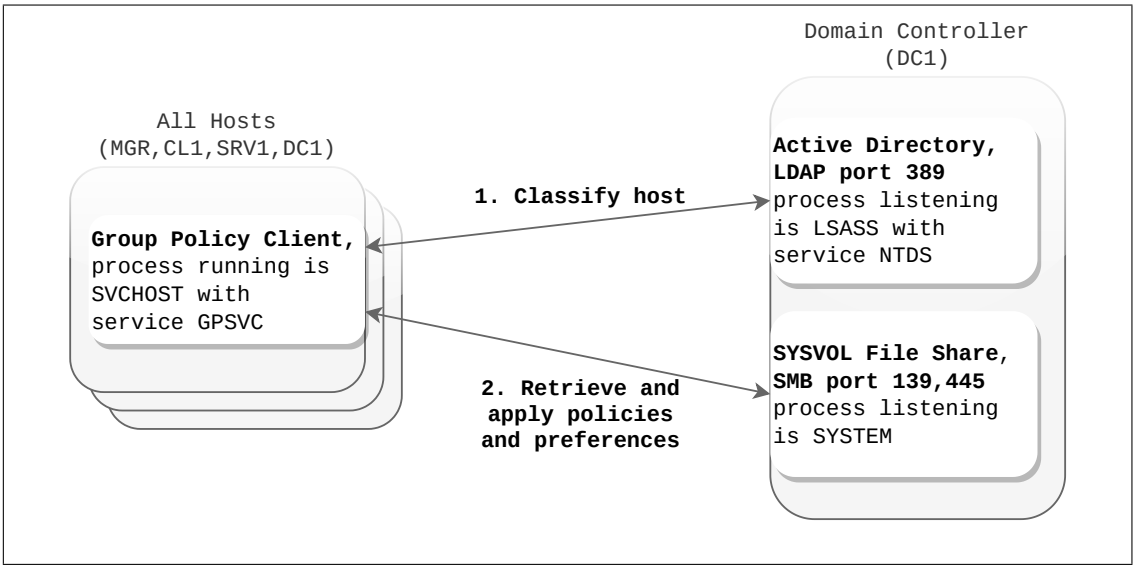


Figure 7.5: Group Policy Architecture.

We primarily rely on permissions set in the file system, and we generally do not focus on file share permissions – as long as the underlying file system supports access control (which NTFS, the default Windows file system, does). If you want to control access to a directory, it is best to manage the directory's access control list directly. File system permissions apply regardless of how the directory is accessed and are therefore enforced for any protocol, not just access via SMB.

Share-level permissions become most relevant when sharing a file system that does not support its own permission structures – for example, the FAT file system. In such cases, share permissions are the only mechanism available to restrict access.

7.4 Group Policy

7.4.1 Architecture

Group Policy Architecture in figure 7.5. The Group Policy client is the `gpsvc` service, which runs inside a service-hosting process (`svchost`). Its configuration is stored in the registry at:
`HKLM:\SYSTEM\CurrentControlSet\services\gpsvc`
The Group Policy client contacts Active Directory to determine the host's location in the LDAP directory (specifically, which OU it belongs to). Based on this information, it retrieves the appropriate Group Policy Objects (GPOs) from the `SYSVOL` file share and then applies the policies and preferences contained within those GPOs.

On domain controllers, the Group Policy client runs by default every five minutes. On all other domain-joined hosts, it runs every 90 minutes, with a random offset of up to

When a host is joined to a domain:

1. Local GPO
2. GPO linked to site (do not do this)
3. GPO linked to domain
4. GPO linked to OU (do this)

GPO's are created, tested, then linked to site, domain or OU.
Last writer wins! In other words, OU-GPOs overwrites conflicting GPOs from previous steps (unless `Enforced` is set).

Figure 7.6: GPO Processing Order.

30 minutes to distribute the load across time. These update intervals can be modified by adding registry entries (typically configured via Group Policy itself) in: `HKLM:\Software\Policies\Microsoft\Windows\System`

- `GroupPolicyRefreshTime` (and corresponding `GroupPolicyRefreshTimeDC` for domain controllers)
- `GroupPolicyRefreshTimeOffset` (and corresponding `GroupPolicyRefreshTimeOffsetDC` for domain controllers)

A Group Policy update can also be forced by running `Invoke-GPUpdate`. If you want the update to occur immediately, you can use: `Invoke-GPUpdate -RandomDelayInMinutes 0`. If `Invoke-GPUpdate` is not available as a cmdlet, you can use the older command-line tool `gpupdate` instead. Remember that we can find all processes and services directly involved in network communication using the method from Chapter one:

```
Get-NetTCPConnection |
Select-Object -Property LocalAddress,LocalPort,State,OwningProcess,`
@{Name='ProcessName';Expression={(Get-Process `
  -Id $_.OwningProcess).ProcessName}},`
@{Name='ServiceName';Expression={(Get-CimInstance -ClassName `
  Win32_Service -Filter "ProcessID=$($_.OwningProcess)").Name}} |
  Format-Table -AutoSize
```

7.4.2 Processing Order

GPO Processing Order in figure 7.6. A site is a defined IP subnet, typically representing a geographical location. In some cases, managing hosts based on their physical location is useful, but in most environments Group Policy is not applied according to geography. For that reason, we will not cover site-based Group Policy processing in this chapter.

- Policy settings are grouped into GPOs

  Computer applies to all users

  Users only for specific users/user groups

- Subcategories are
  - Software settings (installations)
  - Windows settings (login scripts, folder redirection, printers, ...)
  - Administrative templates
- Options for each setting
  - Not configured
  - Enabled
  - Disabled, can mean
    - Reverse the setting from a previous level
    - Force disabling of an OS default

Almost everything is changes to the Windows Registry (HKLM and HKCU)

Figure 7.7: Group Policy.

7.4.3 Policy Settings

Group Policy in figure 7.7. Settings are grouped into Software Settings, Windows Settings, and Administrative Templates, each available under both the Computer and User categories. Some settings are not meaningful to apply to individual users (or user groups), and therefore appear only under the Computer category; likewise, some user-specific settings appear only under the User category. When you configure a setting under the Computer category, it will apply to all users on that system.

Administrative Templates under the Computer category correspond to changes in the `HKLM` portion of the registry, whereas Administrative Templates under the User category correspond to changes in `HKCU`.

If a computer is not joined to a domain, you can still manage it using *Local Group Policy*. This allows standalone hosts to be configured through the same mechanisms and policy structure as domain-based Group Policy, but with settings stored and applied locally.

DEMO LOCAL GROUP POLICY: disable task manager after CTRL-ALT-DEL:

```
User configuration
 -> Administrative templates
 -> System
 -> CTRL-ALT-DEL options
```

http://technet.microsoft.com/en-us/library/cc754948%28v=ws.10%29.aspx:

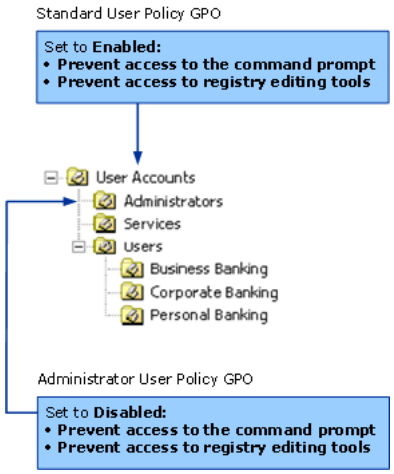


Figure 7.8: Inheritance.

(Note how this is only available to Users, and not under Computer configuration.)

DEMO LOCAL GROUP POLICY: allow bginfo.exe from sysinternals:

```
User configuration
 -> Administrative templates
 -> Desktop
 -> Desktop
 -> Desktop wallpaper
```

If we were to make this change manually by modifying the registry using PowerShell, we would do something like the following:

```
Set-ItemProperty -Path 'HKCU:\Control Panel\Desktop\' `
  -Name WallPaper `
  -Value 'C:\Users\ADMINI~1\AppData\Local\Temp\BGInfo.bmp'
```

Inheritance in figure 7.8. Since GPOs are applied in a specific order, we can implement broad or general policies early in the processing sequence (for example, at the domain level or in a higher-level OU), and then override those same settings for specific groups of users in lower-level OUs.

DEMO: Group Policy in a Domain — "Force Disabling an OS Default" Windows enables its firewall by default. This default is not configured through Group Policy, but we can use Group Policy to turn it off for demonstration purposes:

1. Creating a Group Policy Object is usually best done using the graphical interface (GUI). While it is technically possible to create a GPO using PowerShell—provided we know the exact registry settings we want to configure—it is generally preferable to use the GUI during the design phase of a new GPO.

```
# On DC1
gpme.msc

# Create a new GPO called 'MyFWSettings'
Computer configuration
 -> Administrative templates
 -> Network
 -> Network connections
 -> Windows Defender firewall
 -> Domain profile
 -> Protect all network connections (disable)
# exit gpme
```

2. A good strategy is to maintain a repository containing all GPOs and to separate these from the policies currently applied in the environment. You can then import the relevant GPOs when needed. In practice, this means creating GPOs, giving them a consistent naming convention—such as prefixing them with "My"—and treating them as your configuration "code." These GPOs can be stored in a Git repository or, at the very least, placed under version control and backed up somewhere reliable.

```
# On DC1

# copy the one we created to one that we are going to use
Copy-GPO -SourceName "MyFWSettings" -TargetName "FWSettings"

# link it to the Clients OU so it will be applied to CL1
Get-GPO -Name "FWSettings" |
 New-GPLink -Target "OU=Clients,DC=sec,DC=core"

# btw we can view all GPOs with
Get-GPO -All -Domain $env:USERDNSDOMAIN
# or
Get-GPO -All | Format-Table -Property displayname

# Remember also that GPOs are just objects in an AD LDAP tree:
Get-ADObject -LDAPFilter "(ObjectClass=groupPolicyContainer)" |
 ForEach-Object {Get-GPO -Id $_.Name}
```

```
# And they are just a file structure made available
# to hosts through a share
Get-ChildItem C:\Windows\SYSVOL\domain\Policies
Get-SmbShare
```

3. Let's see how this GPO affects CL1. Keep the Firewall control panel (`firewall.cpl`) visible alongside PowerShell.

```
# On CL1

# Install if not present:
Get-WindowsCapability -Online `
 -Name Rsat.GroupPolicy.Management.Tools~~~~0.0.1.0 |
 Add-WindowsCapability -Online

Invoke-GPUpdate -RandomDelayInMinutes 0

# or just
gpupdate /force

# We can view report with
Get-GPOReport -All -Domain $env:USERDNSDOMAIN -ReportType HTML `
 -Path ".\GPOReport1.html"
.\GPOReport1.html

# or
gpresult.exe /h GPOReport.html
```

A Group Policy Object (GPO) in a domain consists of two parts: 1. The *container*, which is stored in Active Directory and contains metadata (such as version information), and 2. The *template*, which is stored in the file system under the SYSVOL folder and contains the actual policy data and settings.

> *Note that we typically use the GUI to create GPOs. Although it is possible to create GPOs with PowerShell, GPO creation is usually a one-time task and is most efficiently performed using the graphical interface. Instead of trying to create GPOs entirely through PowerShell, we rely on the GUI during the design stage and then use PowerShell for tasks such as linking, backing up, applying, and monitoring GPOs. Use the right tool for the job!*

7.4.4 Settings vs Preferences

GP Settings vs GP Preferences in figure 7.9. In a domain-based Group Policy (as op-

- Settings are enforced
- Preferences can be changed by the user after they have been applied

Figure 7.9: GP Settings vs GP Preferences.

- Everyone is managing Windows servers and laptops, isn't there some common best practice we can all adopt and adapt?
- Microsoft Security Baselines at techcommunity.microsoft.com/t5/microsoft-security-baselines/bg-p/Microsoft-Security-Baselines

Figure 7.10: Are we all doing the same?.

posed to Local Group Policy), we have both traditional settings and *preferences* [40]. Preferences resemble standard settings in appearance, but the key difference is that preferences can be modified by the user after being applied. Preferences extend the capabilities of Group Policy by providing a broader set of configurable options, but they cannot be enforced in the same strict manner as standard policy settings.

Because preferences are not enforced, they are typically used for less critical configurations—tasks where flexibility is acceptable. Common examples include creating shortcuts on a user's desktop.

7.4.5 Professional Practice

Are we all doing the same? in figure 7.10. Microsoft Security Baseline is a collection of GPOs that you can download and import from Microsoft to use as a starting point for establishing a solid baseline security configuration for your hosts. Separate baseline GPOs exist for each Windows edition, and Microsoft also provides baselines for specific functionalities or applications – most notably the Microsoft Edge browser, which is a critical component on all client systems.

When you apply one or more security baselines to a Windows host, one of the first noticeable effects is often that the screen automatically locks after a period of inactivity. This is one of many default hardening measures included in the baselines.

In the following demonstration, we will import the Windows server 2025 Security Baseline and apply it to our Servers OU. This baseline includes a comprehensive set of security settings that can be used as a starting point for securing Windows servers in a domain environment. By applying this baseline, we can quickly implement a range of security configurations that are recommended by Microsoft for Windows Server 2025. Note that you should take a snapshot of your `SRV1` host before applying the baseline, as it will make significant changes to the system's security settings.

```
# We have to use 7zip instead of Expand-Archive because 7zip
# handles weird long file and directory names better
# (trust me :)
choco install -y 7zip

# Download the baseline for Server 2025 from Microsoft
# (the following should be on one line):
curl -O https://download.microsoft.com/download/
  e99be2d2-e077-4986-a06b-6078051999dd/
  Windows%20Server%202025%20Security%20Baseline%20-%202602.zip

# Extract the baseline using 7zip:
7z x Windows%20Server%202025%20Security%20Baseline%20-%202602.zip

# Import the baseline GPOs into Active Directory
cd '.\Windows Server 2025 Security Baseline - 2602\'
cd Scripts
Get-GPO -All | Format-Table -Property displayname
.\Baseline-ADImport.ps1
Get-GPO -All | Format-Table -Property displayname

# We can now open gpmc.msc and see the new GPOs in the Group Policy
# Objects container. From there we can view all settings in the GUI,
# but we will not do that now. Instead, we will link the baseline
# GPOs to the Servers OU and see how they affect SRV1. We can also
# use gpme to edit the GPOs and change some settings if we want to,
# but for now we will just link them as they are.

# We need the OU distinguished name several times
$OU = "OU=Servers,DC=sec,DC=core"

# Get all currently linked to OU Servers
# (Repeat this after you have linked to GPOs below)
Get-ADOrganizationalUnit $OU |
 Select-Object -ExpandProperty LinkedGroupPolicyObjects
# if you want to see the names of the GPOs
# from https://community.spiceworks.com/topic/
       2197327-powershell-script-to-get-gpo-linked-
       to-ou-and-its-child-ou
$LinkedGPOs = Get-ADOrganizationalUnit $OU |
 Select-Object -ExpandProperty LinkedGroupPolicyObjects
$LinkedGPOGUIDs = $LinkedGPOs | ForEach-Object{$_.Substring(4,36)}
$LinkedGPOGUIDs |
```

```
 ForEach-Object {Get-GPO -Guid $_ | Select-Object Displayname }

# link the security baseline for member server to OU Servers
Get-GPO -Name "MSFT Windows Server 2025 v2602 - Member Server" |
 New-GPLink -Target $OU

# On SRV1 apply
Invoke-GPUpdate -RandomDelayInMinutes 0
# or
gpupdate /force

# See changes in report, Policies, Settings, Windows Settings,
# Security Settings, Who is the "Winning GPO"?
gpresult.exe /h GPOReport.html
./GPOReport.html
# Change the Maximun Password Age to 999 days and
# see how it changes the report
# (in gmpe.msc, Computer configuration, Policies, Windows Settings,
# Security Settings, Account Policies, Password Policy,
# Maximum password age)
rm GPOReport.html
gpresult.exe /h GPOReport.html
./GPOReport.html
# The winning GPO is the one that is applied last, in this case
# the one we just edited, and it overwrites the previous setting
# from the baseline GPO. This is because the GPO processing order
# is Local, Site, Domain, OU, and the last one wins.
```

7.4.6 Tools

Group Policy Tools Overview in figure 7.11. In a domain environment, `gpme.msc` is our most important graphical tool. It is where we design and configure our GPOs. After the GPOs have been created and structured in the GUI, the remaining tasks – such as linking, backing up, restoring, monitoring, and applying them – are best handled using PowerShell.

Some Important Notes in figure 7.12. These are some useful notes collected from the excellent book by Jeremy Moskowitz [55]:

- GPOs cannot be applied to the default categories Users and Computers since these are containers not OUs.
- Higher level policies (domain) can be set to ”enforced”, meaning they can not be overridden by lower level (OU) (Enforced takes precedence over Block policy

- Local GPO: `gpedit.msc`
- AD: `gpmc.msc`, `gpme.msc`
- PowerShell:
  `Import-Module GroupPolicy`
  `Get-Command -Module GroupPolicy`
- Group Policy Results
  `Get-GPResultantSetOfPolicy`
  Resultant Set of Policy - RSoP

Remember manual client pull with `Invoke-GPUpdate`

Figure 7.11: Group Policy Tools Overview.

- Settings for a set of users or computers, managed by the same administrators: *should be in same GPO*
- Filtering by WMI can be slow
- Enforce settings with a new GPO at the domain level to avoid changes by OU-administrators
- Link order can be manipulated
- *Max 999 GPOs*

Figure 7.12: Some Important Notes.

inheritance).

- Settings that apply to the same set of users or computers and are managed by the same administrators, should be in the same GPO.
- A GPO (not individual policy settings) can be filtered by security (e.g. instead of all authenticated users, choose a specific user group) or WMI to only apply to a subset of users or computers.
- Use WMI filters only when necessary (exceptions), can be time-consuming or do not time out at all (long logon times...) (WMI filters can be though of as conditional statements).
- Max 999 GPOs can be applied, if you have more, NONE WILL BE APPLIED!
- Do not change default domain or domain controller policy, instead create a new GPO and set it to Enforce.

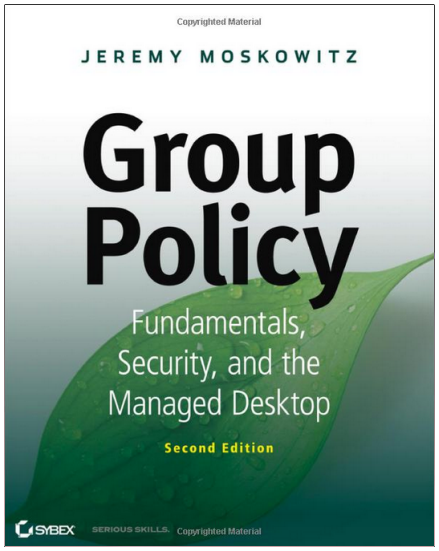


Figure 7.13: Great Book!.

- You create Enforce settings at the Domain level to avoid changes by OU-administrators.
- You can manipulate the link order of GPOs (e.g. the GPOs linked to an OU).

Great Book! in figure 7.13.

7.5 Other Techniques

7.5.1 PsExec

PsExec in figure 7.14. `PsExec` from Sysinternals is one tool that can be used to run remote commands:

```
# On MGR, as Domain Administrator
#
PS C:\Users\Administrator> PsExec.exe \\dc1 ipconfig

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Windows IP Configuration
```

From PSExec Demystified at
www.rapid7.com/blog/post/2013/03/09/psexec-demystified

> PSExec has a Windows Service image inside of its executable. It takes this service and deploys it to the Admin$ share on the remote machine. It then uses the DCE/RPC interface over SMB to access the Windows Service Control Manager API. It turns on the PSExec service on the remote machine. The PSExec service then creates a named pipe that can be used to send commands to the system.

Figure 7.14: PsExec.

Example of PowerShell DSC code:

```
Node SRV1
{
  User MyUsers
  {
      Ensure   = "Absent"
      UserName = "Mysil"
  }
}
```

Figure 7.15: Desired State Configuration.

```
Ethernet adapter tapd1f5f055-ad:

   Connection-specific DNS Suffix  . : openstacklocal
   Link-local IPv6 Address . . . . . : fe80::f816:3eff:fe3a:fbc1%15
   IPv4 Address. . . . . . . . . . . : 192.168.111.134
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.111.1
ipconfig exited on dc1 with error code 0.
```

7.5.2 PowerShell DSC

Desired State Configuration in figure 7.15. The code ensures that the user `Mysil` does not exist on host `SRV1`. PowerShell Desired State Configuration (DSC) is a way of writing *declarative* code in PowerShell, and it has its own framework for applying that code.

Declarative code specifies the desired end state, and the framework takes responsibility for achieving and maintaining it. This contrasts with the more familiar *imperative* style of programming, which focuses on the exact steps required to reach the end state rather than simply describing the state itself.

7.5.3 SSH

SSH is a protocol you are already familiar with from earlier courses. We mention it here because it can also be used between Windows hosts, and PowerShell is able to use SSH as its underlying transport protocol for remoting. This makes it possible to use PowerShell remoting not only with Windows hosts, but also with Linux and macOS systems.

7.5.4 Intune and Endpoint Configuration Manager

Microsoft provides additional tools for configuration management that we will not cover in this course, but they are worth being aware of. Microsoft Intune is a cloud-based service primarily used for managing mobile devices. Microsoft Endpoint Configuration Manager (previously known as System Center Configuration Manager, SCCM) is a GUI-based configuration management platform that uses HTTPS as its main communication protocol between nodes [43].

7.6 Lab tutorials

1. Do the entire demo in chapter 7.4.5 ”Professional practice” where you download the Windows Server 2025 Security Baseline - Member Server, import it as a GPO and apply it to an OU. Import the GPO on DC1, and apply the GPO to the Servers OU so you can check the effect of it on SRV1, in other words, just follow the demo completely step by step.

7.7 Review questions and problems

1. What is the fundamental difference between a *push*-based and a *pull*-based configuration management model?
2. When performing PowerShell remoting to a non-Windows host, why must SSH be used instead of WSMan?
3. What are the two levels of access that must be satisfied when accessing a directory through an SMB share?
4. Why are SYSVOL and NETLOGON especially important in a Windows domain?
5. What is the responsible service for processing Group Policy on Windows hosts?
6. Why do non-domain controller hosts refresh Group Policy every 90 minutes *with a random offset*?
7. What is the correct order of GPO processing for a domain-joined machine?
8. Why is creating GPOs best done through the GUI, even if PowerShell can modify them?
9. What is the main difference between Group Policy *settings* and Group Policy *preferences*?
10. How is Local Group Policy different from Group Policy in a domain?
11. What is the difference between choosing ”Not configured” and ”Disabled” for a policy setting?
12. Why is version-controlling GPO definitions (e.g., prefixing with “My” and storing in a Git repo) considered professional practice?
13. What is the purpose of Microsoft Security Baselines?
14. Why will the following command fail?

```
Get-GPO -Name "MSFT Windows 11 24H2 - User" |
  New-GPLink -Target "CN=Users,DC=sec,DC=core"
```

15. Log in as domain administrator (`SEC\Administrator`, if you are unsure who you are logged in as, write `whoami`) on MGR, write a PowerShell command line that uses `Invoke-Command` to execute `Get-HotFix` on the hosts DC1 and SRV1.

8 Software Package Management

8.1 NSM Grunnprinsipper

NSM Grunnprinsipper for IKT-sikkerhet 2.1 in figure 8.1. Managing software on hosts is a critical part of protecting them against attacks. Any piece of software may contain vulnerabilities, meaning that simply installing it can increase the attack surface of a host. Software may also include malware, and because software installation typically requires administrator privileges, installing malicious software with such privileges can cause severe and far-reaching damage to the entire infrastructure.

In other words, installing and updating software must follow a careful, controlled, and well-defined process. Secure software package management is emphasized across several chapters of the NSM Grunnprinsipper, with the most relevant being principles 1.2, 2.1, and 2.3.

8.2 What is Software?

What Is a Software Application? in figure 8.2. A software application is much more than just the executable file you run to start it. Sometimes we install small, simple programs that may consist of only a single executable file. Other times, we install modules or libraries that extend the functionality of existing tools (for example, Visual Studio Code extensions or Python modules). In many cases, we install large and complex software applications that rely on numerous dependencies, all of which must be installed and configured as part of the process.

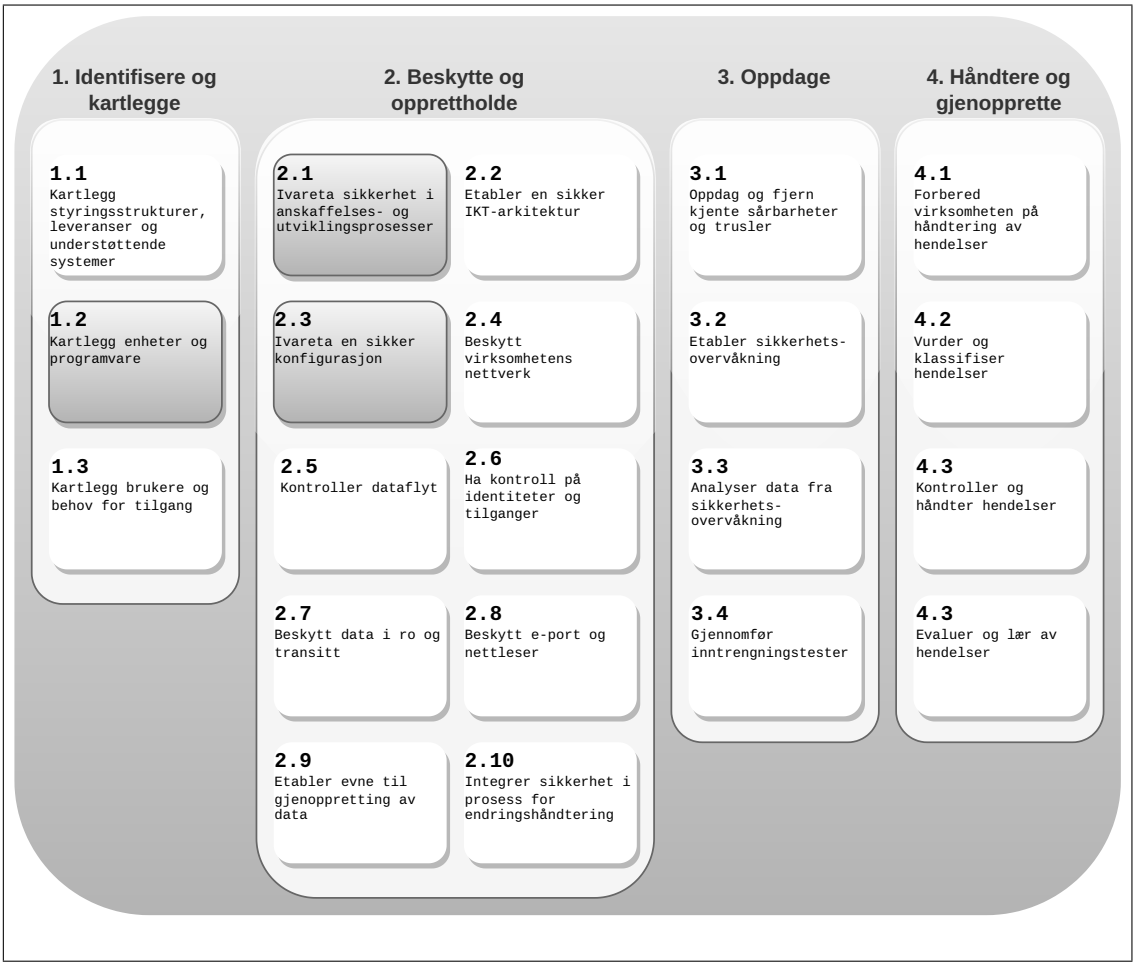


Figure 8.1: NSM Grunnprinsipper for IKT-sikkerhet 2.1.

When working in the open-source ecosystem of Linux, we are often spoiled by rarely having to think about licensing issues. The software we use is typically "free," where free usually refers to freedom (as in open-source licensing), but it can also mean "free of charge." In Windows-based infrastructures we still use a significant amount of free software, but it is far more common to encounter commercial applications that require proper license management. Handling software licenses can be complex, and if it is not done correctly, it is easy to end up in legal or compliance problems.

8.3 Where is Software?

NSM's "Grunnprinsipper for IKT-sikkerhet 2.1" [1] state in chapter 1.2 "Kartlegg enheter og programvare":

> Kartlegging av enheter og programvare er viktig for å få oversikt over

- *Executables*
- Shared and non-shared libraries, modules
- Images (icons), sound, help files, directories
- Config files and registry/database entries
- Menu entries, shortcuts, environment variables
- License
  - Open Source/Free software
  - EULA's
  - Dual licensing
  - Product keys and (time-restricted) activation
  - Per user/Per host licenses
  - License servers

Figure 8.2: What Is a Software Application?.

hva som befinner seg i virksomheten. Kartleggingen av enheter bør avdekke både virksomhetsstyrte enheter, legitime enheter med begrensede rettigheter (for eksempel IoT-enheter) og ukjente enheter (kan være f.eks. ansattes privat utstyr eller ondsinnede enheter). Tilsvarende bør kartlegging av programvare dekke all programvare som benyttes i virksomheten, både installert av IT-avdelingen og uautorisert programvare. Det er viktig at virksomheten selv får oversikt over enheter, programvare og deres sårbarheter før angripere gjør det.

Just this? in figure 8.3. In a perfect world, software would be packaged and managed in a fully standardized way. For example, on Windows it would be ideal if all applications were packaged as MSIX[1]. Unfortunately, this is not the case. Several competing standards and packaging systems are in use, and we simply have to make the best of the situation. On a single Windows host, the easiest way to get an overview of installed software is to use the GUI and inspect Apps & Features. However, in an infrastructure with many hosts, manually checking GUIs is not an option. Instead, we must rely on command-line tools to build an inventory, and in practice, the best overview typically comes from a combination of the following[2]:

1. List installed Roles & features on a server:

[1] docs.microsoft.com/en-us/windows/msix

[2] Perhaps you can combine some of these and expand them into the ultimate Windows software inventory tool—and become famous?

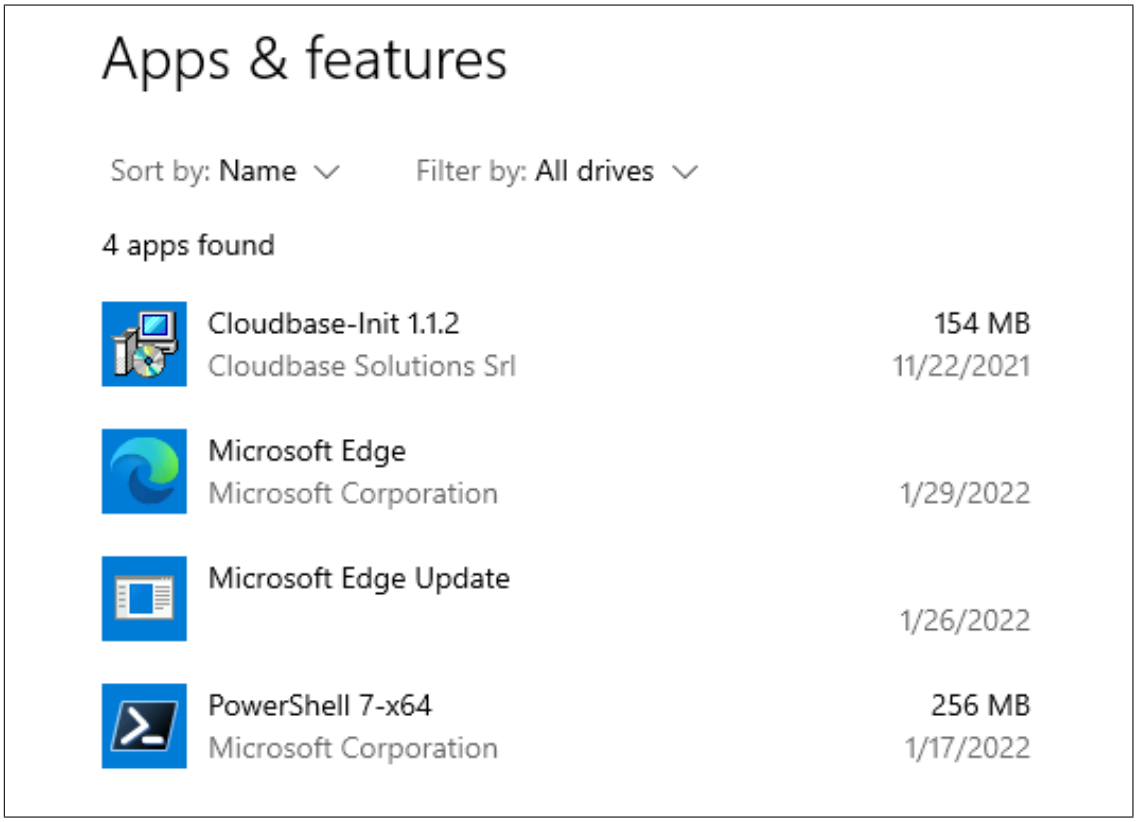


Figure 8.3: Just this?.

```
Get-WindowsFeature |
  Where-Object {$_.InstallState -eq 'Installed'} |
  Format-Table -Property Name,FeatureType
```

2. List installed Capabilities on a Windows 10/11:

```
Get-WindowsCapability -Online |
  Where-Object {$_.State -eq 'Installed'}
```

3. Query what is registered in Windows Management Instrumentation (WMI/CIM) as installed products:

```
Get-CimInstance Win32_Product |
  Format-Table -Property Name,Version
```

4. Query with the Chocolatey package manager (should list everything you have installed with `choco` and maybe more):

```
choco list --local-only # --local-only will be deprecated 2.0
```

5. Query with PowerShell's PackageManagement:

```
Get-Package | Format-Table -Property Name,Version,Providername
# which "Providers" contribute to the list?
# Get-PackageProvider
```

6. Query with the DISM (Deployment Image Servicing and Management) tool:

```
dism /online /get-packages
```

7. Query updates ("fixes" to "Hot" systems, "hot" meaning systems in production)

```
Get-HotFix
```

8. Query the registry for what is registered as possible to uninstall (inspired by "How to use PowerShell to List installed Software"[3])

```
$registrylocations = (
  "HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall",
  "HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall",
  "HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall"
  )
$registrylocations | Get-ChildItem | ForEach-Object {
Write-Output `
"$($_.GetValue('DisplayName')) $($_.GetValue('DisplayVersion'))"
} | Where-Object { $_.Trim() -ne '' }
```

9. The new command line tool that is currently only available as a preview (install from Microsoft Store) on Windows 10/11 looks promising (when this enters a final version available for server as well, and gets PowerShell cmdlets, it will be nice):

```
winget list
```

10. List all appx packages installed:

[3] adamtheautomator.com/powershell-list-installed-software

```
# maybe need to do this:
# Import-Module -UseWindowsPowerShell Appx
Get-AppxPackage -AllUsers |
  Format-Table -Property Name,PackageFullName
```

In other words, obtaining a complete overview of all software installed on a Windows host is not straightforward. Some software may exist only as executable files placed in a folder, without being registered as an installed software package at all. What we do in practice depends heavily on the context. If our organization has standardized on using Chocolatey, then we rely on the `choco` tool. If the organization has standardized on using only AppX packages, we use the corresponding AppX cmdlets, such as `Get-AppxPackage`.

8.4 Supply Chain Security

NSM's "Grunnprinsipper for IKT-sikkerhet 2.1" [1] state in chapter 2.1 "Ivareta sikkerhet i anskaffelses- og utviklingsprosesser":

> 2.1.4 Reduser risiko for målrettet manipulasjon av IKT-produkter i leverandørkjeden. . . . . d) Programvare-produkter bør kun lastes ned fra leverandørens offisielle webside (kun via https). Virksomheten bør oppbevare all installasjonsprogramvare i fil-mapper hvor kun installasjonsansvarlig har skriverettigheter. . . .

Software Supply Chain in figure 8.4. A *supply chain* ("leverandørkjede" in Norwegian) consists of all the steps from raw material (source code) to a finished product delivered to a customer (an installed software package). A supply chain can be visualized in far more detail depending on the context, but these six steps are sufficient for our purposes.

The supply chain begins with one or more programmers writing source code (*can everyone be trusted?*). This code is then combined—typically compiled into one or more executable files – together with libraries (*code supplied by others!*). The resulting executables are integrated with additional system components (*also supplied by others!*) to form an application.

An application is a type of "software artifact," often distributed as a software package (our focus here) or as a container image (e.g., a Docker container). The application is then published to a remote repository (*managed by someone!*). From there, we download it into our own local repository for testing (*can everyone with access be trusted?*). Finally, an installation mechanism is used to deploy the application (*can the installation mechanism be trusted?*).

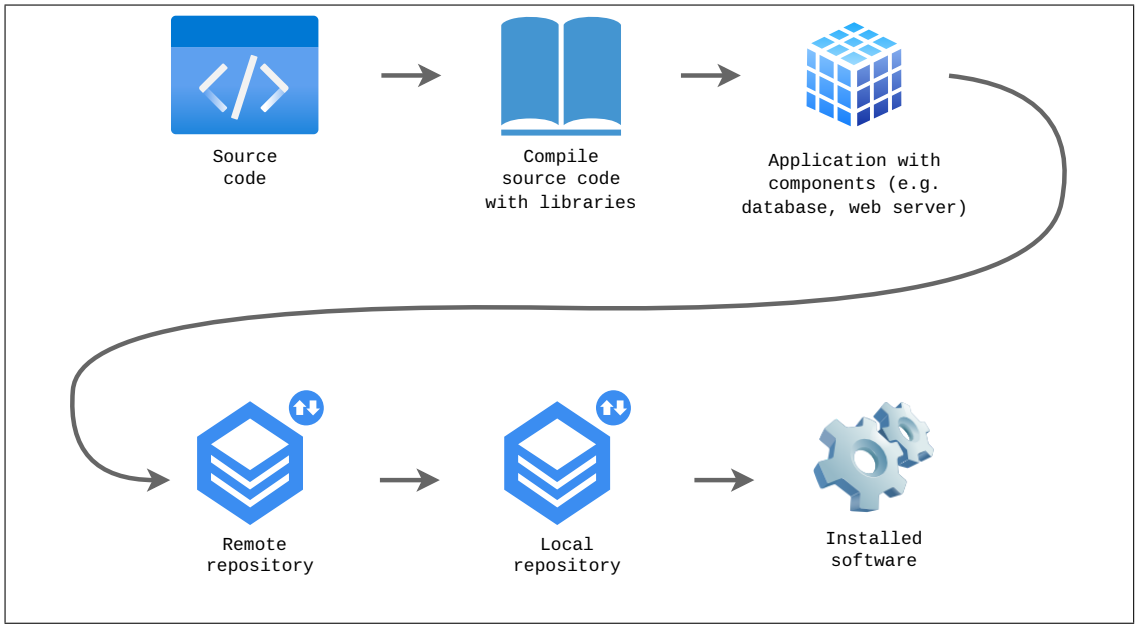


Figure 8.4: Software Supply Chain.

event-stream (2018) Handing over responsibility for a widely used library to someone else...
github.com/dominictarr/event-stream/issues/116

SolarWinds (2020) "They gained access to victims via trojanized updates to SolarWind's Orion IT monitoring and management software"
www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html

Figure 8.5: Supply Chain Attacks.

8.4.1 Threats

Supply Chain Attacks in figure 8.5. Attacks on the supply chain of software packages typically occur in one of four ways:

Typesquatting where an attacker creates software packages with names similar to those of popular packages, hoping that users misspell the name during installation. For example, installing `coffe-script` instead of `coffee-script` [71].

Account hijacking through credential stuffing (attempting to log in using lists of known or previously compromised credentials) or through credential theft. This is believed to be how outsiders may have gained access to SolarWinds' build system, although the exact details remain uncertain.

Social engineering by tricking someone into handing over maintainer privileges for a package, as happened with `event-stream`. Once an attacker gains maintainer access, they can introduce malicious changes directly into the package.

Man In The Middle (MITM) where an attacker manages to place themselves between two steps of the supply chain. An example is DNS cache poisoning, where users are redirected to a malicious package repository controlled by the attacker.

The malicious payload inserted through these attacks often focuses on credential theft, ransomware deployment, sabotage, installing backdoors, or crypto-mining.

Duan et al. [19] conducted an extensive study between 2018 and 2020 in which they analyzed more than one million packages across the PyPi, npm, and RubyGems ecosystems. They identified 339 previously unknown malicious packages, three of which had been downloaded more than 100,000 times.

Samuel and Cappos [64] performed an influential study on package manager security in 2009. They examined the security mechanisms of the widely used Linux package managers `apt` and `yum` (Chocolatey was not included in their analysis). Package managers operate by retrieving software packages from a package repository. Such a repository contains the packages themselves, associated metadata (including version information, authors, and dependencies), and a root metadata file that acts as an index of all packages available.

Although the specific attacks described by Samuel and Cappos are unlikely to succeed today due to improved mitigations, the way they conceptualize threats to package management systems remains highly relevant. Their work helps us understand how attackers might target different points in the package management workflow:

Replay Attack If the attacker is Man-In-The-Middle, they can serve an old version of the repository even though the root metadata is signed ... *Protect by making sure you don't accept metadata older than what you already have*.

Freeze Attack If the attacker is Man-In-The-Middle, they can avoid updating the repository ... *Protect by limiting how long signed root metadata is valid*.

Metadata Manipulation Attack If metadata is not signed (not root nor package metadata), Man-In-The-Middle can easily offer newer versions of packages which are actually older version (with vulnerabilities the attacker know how to exploit) ... *Protect by requiring signed metadata*.

Endless Data (DOS) Attack As root metadata the Man-In-The-Middle attacker will just serve an endless file ... *Protect by monitoring system resources, setting hard limits or possibly by keeping package management cache on a separate partition*.

8.4.2 Protection

The Security Technical Advisory Group (STAG) of the Cloud Native Computing Foundation (CNCF) maintains a ”living” document titled Software Supply Chain Best Practices [22]. This document outlines four themes that are fundamental to securing the software supply chain:

Verification Every step in the supply chain should be verified through cryptographic signing. In practice, this means that artifacts such as Git commits must include a valid digital signature. There should be no ”trust without cryptographic verification”.

Automation As many processes as possible should be automated using scripts – whether command-line scripts or infrastructure-as-code tools. Automation reduces the risk of human error and increases consistency by ensuring that processes produce the same result every time.

Authorization in Controlled Environments All participants in the supply chain, whether human users or software processes, must be granted access based on the principle of least privilege. This applies both to user accounts on systems and to automated processes performing tasks in the pipeline.

Secure Authentication All identities in the supply chain must use strong authentication. Human users should rely on multifactor authentication, while services and automated processes should mutually authenticate using cryptographic keys. And, of course, no service should rely on ”default passwords” such as `abc123` (“but I only used it for testing— it wasn’t supposed to end up in production...”).

8.5 Framework and Process

NSM’s ”Grunnprinsipper for IKT-sikkerhet 2.1” [1] state in chapter 2.3 ”Ivareta en sikker konfigurasjon”:

> 2.3.1 Etabler et sentralt styrt regime for sikkerhetsoppdatering. Installer sikkerhetsoppdateringer så fort som mulig. a) Etabler en prioriteringsliste for oppdateringer. Operativsystem og applikasjoner på de ansattes klienter bør prioriteres. Videre bør man oppdatere servere som inneholder standard applikasjoner og operativsystem, programvaren i skrivere, samt enheter som styrer virksomhetens nettverk (svitsjer, rutere). b) Etabler en rutine med klare ansvarsforhold for i) hvor ofte oppdateringer skal utføres (mye bør kunne automatiseres) og ii) ansvarlig rolle for oppfølging hvis en oppdatering ikke kan gjennomføres eller må utsettes. c) Isoler servere

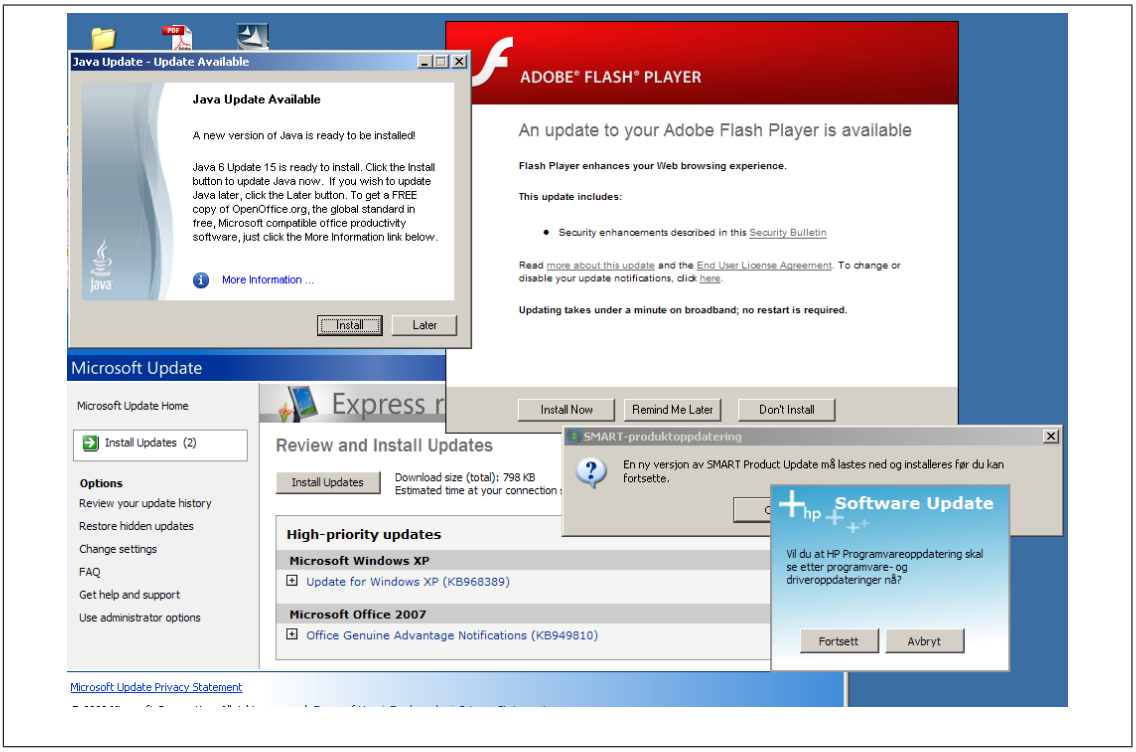


Figure 8.6: We need standards!.

og annet som oppleves som vanskelig å holde oppdatert, se 2.5.4. d) Virksomheter bør automatisere og forenkle prosessen for å implementere nye sikkerhetsoppdateringer.

In other words, we need a clearly defined process for how software enters our infrastructure. We should aim to keep this process as simple as possible and automate it wherever we can (always remember: ”humans make mistakes; computers only make mistakes when we program them to”). To understand how such a process can be implemented, let us examine the last three steps of the supply chain – remote repository, local repository, and installed software.

8.5.1 Package Managers, Installers and Formats

We need standards! in figure 8.6. In the old days – and unfortunately still today – you will often encounter applications that implement their own mechanisms for updating themselves. This leads to chaos. We must strive to ensure that as much software as possible is installed and updated in a standardized manner. By standard, we mean that software should be packaged in a standard file format and managed (installed, queried, updated, and removed) using standard installers or package managers. Our options are:

- Microsoft package file formats: msi, msix, msu, appx, nupkg (plus roles, features, capabilities)
- Installers/Package Managers: Add-AppxPackage, dism, msiexec, choco, scoop, appget, winget, ninite
- (Language-specific package managers: npm (NodeJS), pypi (Python), ppm (Perl), rubygems (Ruby))

Normally, we distinguish between an *installer* and a *package manager*. The installer is the software responsible for installing, updating, and uninstalling the software package. The package manager, on the other hand, is responsible for downloading packages from repositories on the Internet (or from local servers), resolving dependencies, verifying signatures, and performing other checks before handing the package over to the installer.

> *A package manager will typically install dependencies along with the package you request. In other words, you may think you are installing only one package, but in practice, several packages may be installed to satisfy dependencies.*

For example, in Debian-based Linux distributions (such as Ubuntu), `apt` functions as the package manager, while `dpkg` acts as the installer.

On Windows, we have the following installers:

- `msiexec` is an older but mature tool used for installing MSI packages.
- `dism` (Deployment Image Servicing and Management) is a tool for managing offline disk images, but it can also be used for online images – meaning your currently running Windows host. Among its many capabilities, DISM can manage packages.
- `Add-AppxPackage` is the PowerShell cmdlet used to install AppX and MSIX packages. Internally, it acts as a wrapper around DISM.

A Package is not Self-Contained in figure 8.7. A very important characteristic to be aware of with software packages is that even if they are packaged in a standardized format, they may simply act as wrappers around a vendor-specific installation mechanism that executes its own custom scripts. For example, if we install the most popular package in the Chocolatey community repository as of February 23rd, 2023 – Adobe Acrobat Reader – we observe that the installation process downloads several hundred megabytes directly from `adobe.com`.

The most widely known package managers that work on Windows and their default repositories are:

```
PS C:\Users\Administrator> choco install -y adobereader
Chocolatey v1.2.1
Installing the following packages:
adobereader
By installing, you accept licenses for the packages.
Progress: Downloading adobereader 2022.003.20322... 100%

adobereader v2022.003.20322 [Approved]
adobereader package files install completed. Performing other installation steps.
WARNING: No registry key found based on  'Adobe Acrobat Reader DC*'
True
Configuring manual update checks and installs.
Downloading adobereader (update)
  from 'https://ardownload2.adobe.com/pub/adobe/reader/win/AcrobatDC/2200320322/AcroRdrDCUpd2200320322_MUI.msp'
Progress: 100% - Completed download of C:\Users\Administrator\AppData\Local\Temp\chocolatey\AcroRdrDCUpd2200320322_MUI.msp (287.01 MB).
Download of AcroRdrDCUpd2200320322_MUI.msp (287.01 MB) completed.
Hashes match.
Downloading adobereader
  from 'https://ardownload2.adobe.com/pub/adobe/reader/win/AcrobatDC/1500720033/AcroRdrDC1500720033_MUI.exe'
Progress: 100% - Completed download of C:\Users\Administrator\AppData\Local\Temp\chocolatey\AcroRdrDC1500720033_MUI.exe (146.89 MB).
Download of AcroRdrDC1500720033_MUI.exe (146.89 MB) completed.
Hashes match.
Installing adobereader (installer)...
adobereader (installer) has been installed.
Progress: 100% - 2/2 completed
 adobereader may be able to be automatically uninstalled.
 The install of adobereader was successful.
  Software installed to 'C:\Program Files (x86)\Adobe\Acrobat Reader DC\'

Chocolatey installed 1/1 packages.
 See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
```

Figure 8.7: A Package is not Self-Contained.

- Chocolatey.

  Installer https://chocolatey.org/install

  Default repo https://community.chocolatey.org

- Scoop.

  Installer https://scoop.sh

  Default repo https://github.com...

- Ninite. Create a mix of packages, can be updated as a unit, see https://ninite.com

- WinGet. See "Use the winget tool to install and manage applications"[4]

  Installer https://github.com...

  Default repo https://winget.azureedge.net/cache

- AppGet.

  Installer https://appget.net/download

  Default repo https://appget.net/packages

Microsoft probably got many ideas from AppGet when they created WinGet, the story "The Day AppGet Died"[5] is an interesting read.

[4] docs.microsoft.com/en-us/windows/package-manager/winget
[5] keivan.io/the-day-appget-died

- Where are all the files to be updated?
- What if the update fails?
- *Why are updates different from fresh installs?*
  - no physical access required?
  - host may not be in a "known state"
  - host may have "live" users
  - host may be gone
  - host may be dual-boot

Installations/updates have to be transactional/atomic in behavior for both install and uninstall.

Figure 8.8: Fresh Install vs Update.

8.5.2 Updates

Fresh Install vs Update in figure 8.8. There must be a reliable way to locate the files installed by previous versions of a package, and during an update process all relevant data must be carefully backed up so it can be restored if something fails (for example, a power outage while files are being copied). This is one of the reasons why Windows Update often takes a long time. An update process is inherently complex and involves many modifications to files and registry settings. All these changes need to be transactional (atomic), meaning that Windows creates a backup copy of every file and configuration change before proceeding, allowing a full rollback if something goes wrong. These backup copies are stored in the WinSxS folder ("side-by-side"), which often consumes a significant amount of disk space:

```
PS> du C:\Windows\WinSxS

DU v1.62 - Directory disk usage reporter
Copyright (C) 2005-2018 Mark Russinovich
Sysinternals - www.sysinternals.com

Files:        61355
Directories:  24452
Size:         7,022,037,443 bytes
Size on disk: 7,425,290,240 bytes
```

Challenges for Reliability in figure 8.9. In today's world, which is filled with cybersecurity challenges, we must also remember that reliability is essential. Unfortunately, software installations and updates are inherently risky. Microsoft releases a collection of updates (also known as patches) on the second Tuesday of every month—*Patch Tuesday*.


NEWS ANALYSIS

Patch expert calls on Microsoft execs to fix Windows updating

IT admins surveyed by Susan Bradley, who moderates the PatchMangement.org mailing list, said Microsoft's current strategy is failing. So she took their concerns to Microsoft leaders, including CEO Satya Nadella, and asked them to make changes.

By Gregg Keizer

Senior Reporter, Computerworld | AUG 7, 2018 12:26 PM PDT

Figure 8.9: Challenges for Reliability.

Even though Microsoft performs extensive testing, errors can still occur. Therefore, we should always follow the *One, Some, Many* technique described by Limoncelli, Hogan, and Chalup [29]: begin by installing, updating, or upgrading a single test host; then apply the update to your own machine and those of close colleagues; next, apply it to a dedicated lab environment; and only after that roll it out to an entire user group, and so on. In other words, test thoroughly and never update everything at once. This process is often referred to as a ”staged rollout” (or ”update rings”), and it is a common industry practice for reducing the risk associated with software updates.

8.6 Installations and Updates: How to do it?

How to do it in figure 8.10. If you are responsible for installing or updating (”patching”) software in a Windows infrastructure, your work will primarily focus on the last three steps of our defined supply chain. If we assume that your organization has adopted Chocolatey as its package manager, in addition to the Windows Update mechanism, then the procedure you follow may look as described in the following chapters.

8.6.1 Acquiring Packages

We will get packages from the Chocolatey community repository and/or from creating Chocolatey packages ourselves [11]:

> Chocolatey allows you to create packages easily using the package builder, but it also allows you to take packages from the Chocolatey Community Repository and recompile them for internal use - this is a process known as *package internalization*.

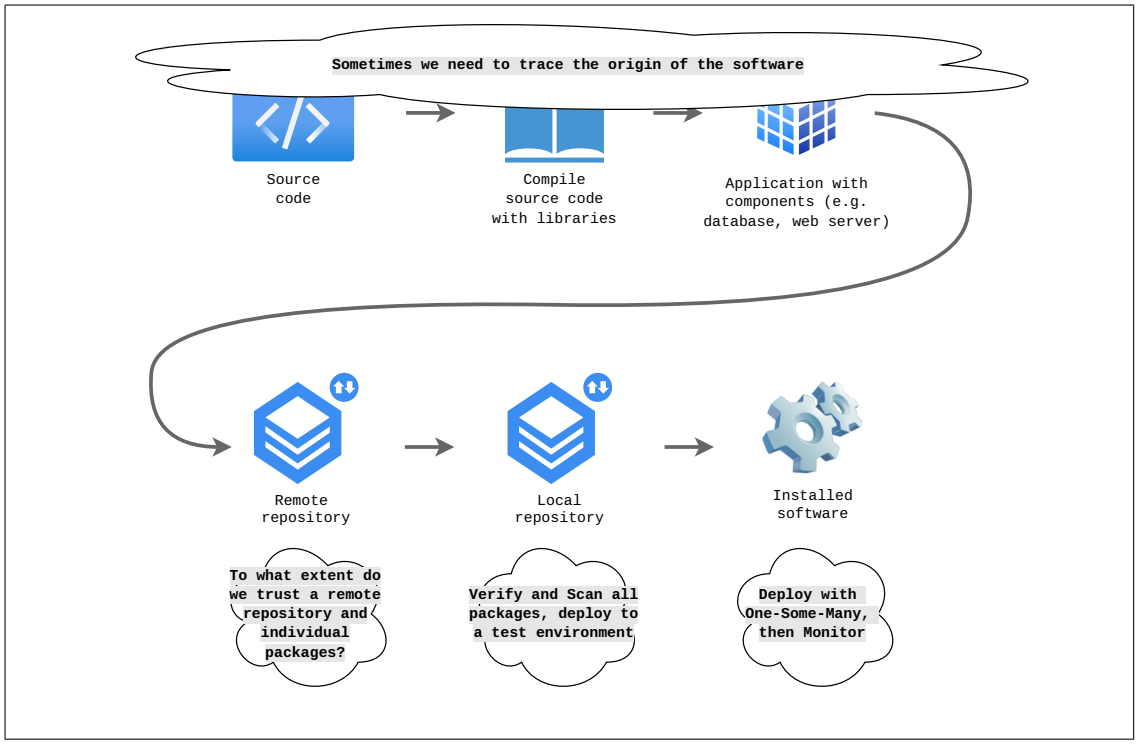


Figure 8.10: How to do it.

By default, Chocolatey installs software from its community repository

```
PS> choco source list
chocolatey - https://community.chocolatey.org/api/v2/
```

To what extent do we trust this package repository? The web pages of Chocolatey state:

> As an organization, you want 100% reliability (or at least that potential), and you may want full trust and control as well. This is something you can get with internally hosted packages, and you are unlikely to achieve from use of the Community Package Repository. If your use of Chocolatey is for an organization/business, you are likely to have a low tolerance for production breakages and/or low trust for the greater internet. You probably would not want to give control of your infrastructure over to community members and volunteers. Organizational use of the community repository is not recommended.

In other words, we should set up our own internal repository. But it is important to note that Chocolatey already performs many of the same actions with the packages in its public repository that we would need to implement in our internal repository:

Validation is about checking that a package adheres to required standards. It is similar to linting and unit testing in programming: performing basic compliance checks such as whether the version number follows the correct format, whether author information is included, whether a contact email is provided, and whether all expected files are present in the correct locations.

Verification is about testing that the installation process actually works as intended.

Security scanning can cover many activities, but the most common approach is to check packages for known malware by submitting them to VirusTotal [75] (which is what Chocolatey does).

We can place some trust in the Chocolatey community repository, but we should always review the results of validation, verification, and security scanning for each package and for each specific version before downloading them into our own internal repository for further inspection.

Downloading Windows Updates is a much more restricted and simpler domain than the "Chocolatey universe", since updates originate solely from Microsoft.

8.6.2 Host Internal Repository

We want to host our own internal package repository for both security and performance reasons. We do not want to depend on remote repositories where uptime, rate limits, or network performance could affect our organization's ability to perform installations and updates. Thus, performance—alongside security—is a strong motivation for maintaining an internal repository.

In our internal repository, we perform validation, verification, and security scanning. The most important reason for hosting our own repository is security. We must scan packages for vulnerabilities using tools such as Snyk [65]. We may also use VirusTotal, a cloud-based service to which we can submit files for analysis. According to their website,

> VirusTotal inspects items with over 70 antivirus scanners and URL/domain blocklisting services, in addition to a myriad of tools to extract signals from the studied content.

We must be careful with what we submit to VirusTotal, since anything we upload will be shared with many security companies [6].

It is extremely important to deploy packages in a test environment to observe what actually happens at runtime ("when it actually installs"). Some packages are simple enough to inspect manually by examining the files in the package and using tools such

as `Select-String` to search for URLs or keywords that may indicate runtime behavior. However, most packages are large and difficult to analyze manually, and the only reliable way to understand their behavior is through a controlled test deployment.

As mentioned earlier, *some packages download code and data from remote servers during installation, and we need to verify this to ensure that what is being downloaded is trustworthy*. Packages that contain scripts which download external code or data are good candidates for repackaging into new packages that we create ourselves (*package internalization*, as discussed previously). This gives us greater control and makes it easier to trust the packages we deploy.

To create an internal package repository for Chocolatey, their documentation lists the following options:

Folder/UNC share is to have all the packages in a folder in the file system which of course can be shared on the network (a SMB share just like the one Group Policy uses to download GPOs).

Simple server is a simple web server setup which allows for access with https.

Package gallery is an advanced web server (https access here as well) setup with a database and a structure that scales for numerous packages and clients.

For Windows Updates, we should use Windows Server Update Services (WSUS) [38]. WSUS is the standard internal repository for Microsoft updates (HotFixes). While we can place a high level of trust in the security of Microsoft's updates, we must still consider reliability and performance – two key reasons for integrating WSUS into our infrastructure.

Updates must be thoroughly tested in our test environment for every Windows version on which they will be deployed. Microsoft has released several updates over the years that have caused issues for users, which makes testing essential.

From a performance standpoint, it would be highly inefficient if all Windows hosts worldwide contacted the same Microsoft servers to download the same updates. Instead, it is far more efficient for each organization to download updates once to their WSUS server and then have their clients retrieve updates from that local WSUS instance.

8.6.3 Install or Update ("Patch")

We can specify where choco installs from. This can either be a file path or a https location, e.g.

```
choco install zoomit -s https://community.chocolatey.org/api/v2/
```

We can also ask choco to use DISM to show and install Windows features

```
choco list --source windowsFeatures
```

For Windows Update, we typically use Group Policy to redirect all Windows Update clients (the service `wuauserv` running inside an `svchost` service hosting process) to the WSUS server instead of Microsoft's public update servers.

In any case, it is wise to remember the One, Some, Many principle. Ideally, you have already caught all potential problems during testing in your dedicated test environment. But if issues still remain, performing a careful step-by-step rollout using the One, Some, Many approach can save you from significant trouble.

If you want to learn more from a broader, higher-level perspective (a more "risk-based approach" aimed at managers), take a look at the NIST Special Publication Guide to Enterprise Patch Management Planning: Preventive Maintenance for Technology [66].

8.7 Lab tutorials

1. No lab tutorials this week.

8.8 Review questions and problems

1. Describe the typical content of a software application (in other words: what kind of components have to be part of a typical software package).
2. Explain the difference between a package manager and an installer in the context of Windows and Linux.
3. What is ”typesquatting”?
4. What threat scenarios do “freeze attacks” and “replay attacks” try to exploit in package repositories?
5. Why would you want to maintain your own software repository in your infrastructure (and direct all your clients package managers to this repository)?
6. Describe the ”one, some, many” technique of software updates.
7. What do you have to consider / ”be concerned about” when doing software updates on a host compared to doing fresh installation of a host?
8. Describe the software supply chain and how security can be implemented in each step.
9. Why should organizations avoid installing software directly from the Chocolatey Community Repository in production environments?
10. Why is it risky to upload arbitrary software packages to VirusTotal?
11. Consider the following two cases:

    (a) You want to install version 22.0 of 7-zip from the Chocolatey community package repository at
    community.chocolatey.org/packages/7zip.install/22.0

    (b) You want to install the PowerShell module PSWindowsUpdate from the PowerShell Gallery at
    www.powershellgallery.com/packages/PSWindowsUpdate/2.2.0.3

    Try to find our as much as possible about these two cases, are there any problems? Would you trust these install locations and the packages? Which procedure would you use for installing them?
12. Use PowerShell to list installed software using both `choco` and `winget`. Compare the output from the two tools. What differences do you observe, and why do these differences occur?

9 Logging and Monitoring

9.1 Introduction

From NRK Innlandet[1]:

> Østre Toten har vært uten datasystemer en måned etter hacking ØSTRE TOTEN (NRK): PST mener dataangrep er en av de største truslene i 2021. I Østre Toten innrømmer ordføreren at sikkerheten ikke var god nok.
> ...Ordføreren erkjenner at datasikkerheten ikke har vært god nok, men sier at de ennå ikke har konklusjonen på hvorfor det skjedde.

Several "famous" people (search the internet for references) have stated some variation of the phrase: *there are two types of companies—those that have been compromised, and those that do not yet know they have been compromised*. This mindset highlights an important principle in modern security: we must always assume that our infrastructure will be attacked, and that at least some attacks will eventually succeed. Preventive measures are essential, of course, but they are not sufficient on their own. We must be equally prepared to respond effectively when an incident occurs. In practice, this raises several critical questions:

- How long did the attacker have access?
- Are they truly out of the environment now?
- Are the systems and backups in a known "clean" state?

[1] www.nrk.no/innlandet/kan-ta-et-halvt-ar-for-ostre-toten-a-rette-opp-dataangrep-1.15364106

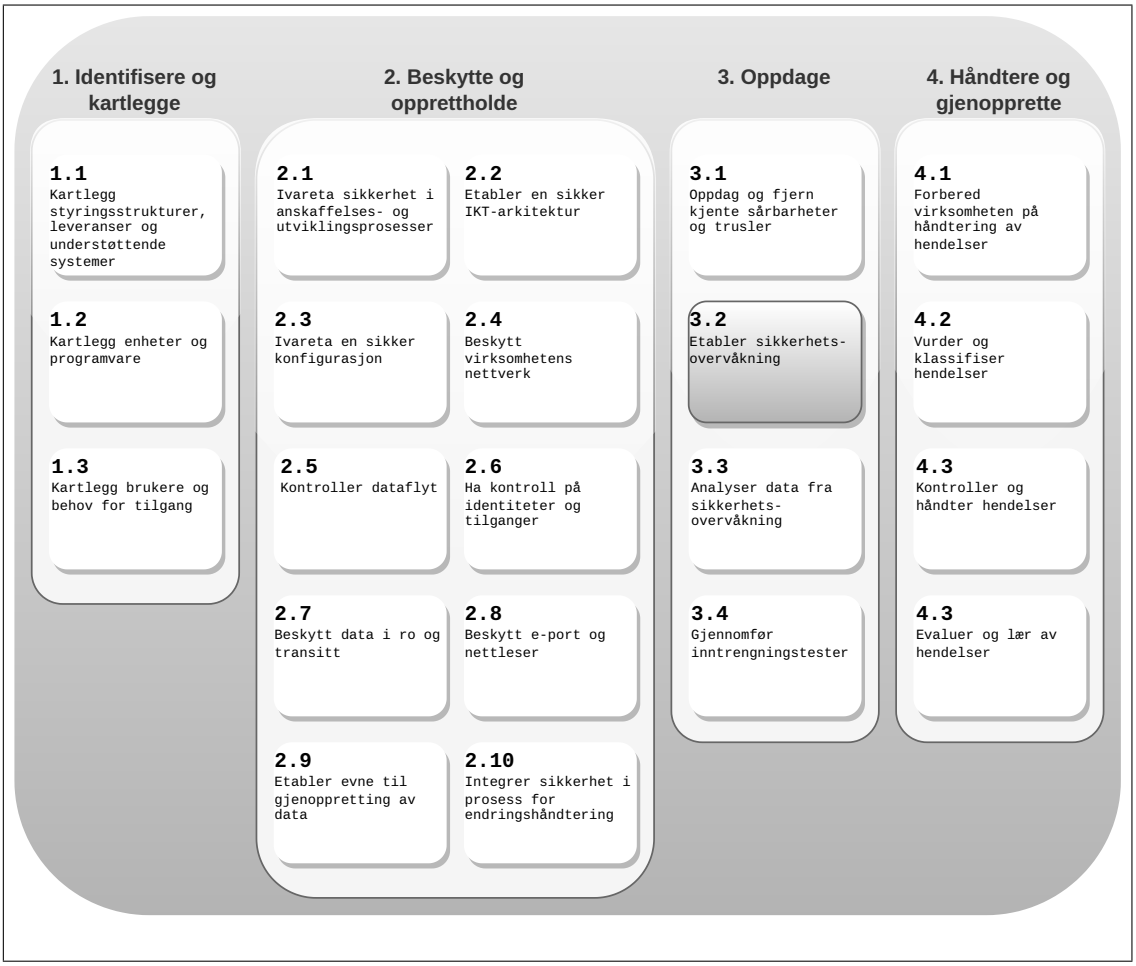


Figure 9.1: NSM Grunnprinsipper for IKT-sikkerhet 2.1.

9.2 NSM Grunnprinsipper

NSM Grunnprinsipper for IKT-sikkerhet 2.1 in figure 9.1. An important element of "Get the basics right" is establishing a proper setup for logging and monitoring. We need mechanisms that allow us to detect whether we are currently under attack or have already been compromised. In addition, we must ensure that the necessary data is available to support a thorough investigation (forensics) after an incident has occurred. In NSM's "Grunnprinsipper for IKT-sikkerhet 2.1" [1], these requirements are addressed in Chapter 3.2, "Etabler sikkerhetsovervåkning".

Two categories of data:

- Counters (numeric values: *periodic* or *accumulating*)
  - *CounterSet* e.g. `Process`
  - *Counter* e.g. `Working Set`
  - *Instance* e.g. `pwsh#1` (instances are numbered if there are more than one)
  - *Path* e.g. `\Process(pwsh#1)\Working Set`
- Log events (text messages)

Sometimes log events are generated from counters

Figure 9.2: Terminology.

9.3 Counters

In this chapter, we will at times use the excellent module `PSScriptTools`, as it provides helpful cmdlets such as `Get-MyCounter` (an enhanced version of `Get-Counter`) and `Convert-EventLogRecord` (which improves the output produced by `Get-WinEvent`). From the previous chapter, we know that we should think carefully before installing software from the PowerShell Gallery. Therefore, you should review the PSScriptTools package[2] before running:

```
Install-Module -Name PSScriptTools
```

The author of PSScriptTools is Jeff Hicks, a well-known and respected contributor in the PowerShell community. Given this, installing the module in our "low-risk" SkyHiGh infrastructure is considered safe.

Terminology in figure 9.2. For logging and monitoring, we generally work with two categories of data: *counters* (numeric values) and *log events* (text messages). We will begin with counters.

Counters can be either *periodic* (for example, memory usage) or *accumulating* (for example, system uptime). A periodic counter reports a value at a specific moment in time, typically calculated over the last second or last few milliseconds. Because a periodic counter resets to zero between each measurement, its observed values may rise and fall over time.

An accumulating counter, on the other hand, only increases until something explicitly resets it to zero. For example, when a host reboots, the uptime counter is reset, and when a service process restarts, its total CPU usage returns to zero.

[2] www.powershellgallery.com/packages/PSScriptTools

- `Get-Counter` (we sometimes use the wrapper `Get-MyCounter`)
- `Get-CimInstance`
- Use .NET directly

Figure 9.3: Implementation.

Windows provides a subsystem called Performance Logging and Alerting (PLA), which defines *counter sets* that contain individual *counters*. A counter may be either *single-instance* (representing a single value) or *multi-instance* (for example, one instance per CPU, plus an instance called `_Total`). Each counter is identified by a *path*. When a counter has multiple instances – such as the "Processor" counter on a system with two CPUs—Windows provides a special instance called *_Total*, representing the aggregate value across all instances.

9.3.1 Implementation

Implementation in figure 9.3. We have three options for accessing counters from the command line:

Get-Counter is the PowerShell cmdlet that we will use, e.g.

```
Get-MyCounter -Counter '\Processor(_Total)\% Processor Time'
```

Get-CimInstance is using the Windows Management Instrumentation (WMI) described in chapter one, e.g.

```
(Get-CimInstance -ClassName Win32_PerfFormattedData_PerfOS_Processor |
  Where-Object {$_.Name -eq '_Total'}).PercentProcessorTime
```

Performance Logs and Alerts (PLA) directly create a PowerShell object from the .NET-API, e.g.

```
New-Object System.Diagnostics.PerformanceCounter("Processor",
  "% Processor Time", "_Total")
```

Some examples (note the use of `ExpandProperty`, since a property can be an object itself sometimes it is necessary to use `ExpandProperty` to see all available properties/"sub-values"):

- All CounterSets

```
Get-Counter -ListSet * |
 Sort-Object CounterSetName |
 Select-Object -Property CounterSetName |
 Out-GridView
```

- How many Counters are there?

```
Get-Counter -ListSet * |
 Select-Object -ExpandProperty Counter |
 Measure-Object
```

- All Counters in a specific CounterSet

```
Get-Counter -ListSet Process |
 Select-Object -ExpandProperty Counter
```

We are primarily interested in the value called `CookedValue`. Windows also provides a raw value and a secondary value, but these two are combined into a final, human readable form – the CookedValue.

A useful way to think about this is to imagine the raw value representing full seconds (for example, "2 seconds"), and the secondary value representing additional milliseconds (for example, "205 milliseconds"). The CookedValue then becomes the combined, meaningful value: 2.205 seconds. This is the value we typically want to work with.

Some examples of retrieving values:

- Show the counter

```
Get-Counter -Counter '\Process(_Total)\Working Set'
```

- The CounterSamples-object

```
(Get-Counter -Counter `
  '\Process(_Total)\Working Set').CounterSamples
```

- The values

```
(Get-Counter -MaxSamples 3 -Counter `
  '\Process(_Total)\Working Set').CounterSamples.CookedValue
```

- With the wrapper `Get-MyCounter` the `Value` is the `CookedValue`, so this produces the same output

```
(Get-MyCounter -MaxSamples 3 -Counter `
  '\Process(_Total)\Working Set').Value
```

If we want to retrieve counters from remote hosts, we should use the following approach:

```
# Do this:
$scriptblock = {
  Get-Counter '\Processor(_total)\% Processor Time'
  }
Invoke-Command -ComputerName dc1,srv1 `
               -ScriptBlock $scriptblock
```

You will find many posts on the Internet that describe the old way of retrieving data from remote hosts. In this context, old way refers to connecting to the host on port 135 using the DCOM (Distributed Component Object Model) protocol. This method is deprecated and should not be used today. We mention it here only because you may encounter references to it and wonder why we do not rely on this approach:

```
Get-Counter -ComputerName dc1,srv1 `
  '\Processor(_total)\% Processor Time'
```

If you want to see all cmdlets that support this old, deprecated method of accessing remote hosts, you can run the following command:

```
Get-Command |
  Where-Object {
    $_.Parameters.Keys -contains "ComputerName" -and
    $_.Parameters.Keys -notcontains "Session"}
```

We should only access remote hosts using PowerShell's remoting capabilities, which rely on the WinRM protocol and communicate over ports 5985 and 5986. This is the mechanism used when we run the `Invoke-Command` cmdlet. See also Chapter Seven for more details about remoting.

Sometimes it is necessary to analyze counter values in an application like Excel or another spreadsheet tool. To export counter values for use in a spreadsheet, we can run:

```
$counters = '\Processor(0)\% Processor Time',
            '\Process(_Total)\Working Set'

# Export-Counter is only in Windows PowerShell (5.1)
Get-Counter -Counter $counters -MaxSamples 10 |
 Export-Counter -Path C:\PerfLogs\cap.csv -FileFormat csv

# Need to do this in PowerShell Core instead
Get-Counter -Counter $Counters -MaxSamples 10 |
ForEach-Object {
  $_.CounterSamples | ForEach-Object {
    [pscustomobject]@{
      TimeStamp = $_.TimeStamp
      Path = $_.Path
      Value = $_.CookedValue
    }
  }
} | Export-Csv -Path $home\out.csv -NoTypeInformation

Get-Content out.csv | Set-Clipboard
# paste in Excel, Data, Text to Columns)
```

9.3.2 GUI Tools

GUI Tools in figure 9.4. On Windows, we can view counters graphically using tools such as Task Manager, Performance Monitor, Resource Monitor, and several others. In this book, we use Windows Admin Center, since it is Microsoft's modern, officially supported tool for managing small Windows environments. At a larger scale, counter data is typically collected by a central monitoring system. Such systems often use visualization tools like Grafana[3] to present data in real time. For monitoring the servers that make up the SkyHiGh private cloud, NTNU uses Zabbix[4].

9.4 Log Events

9.4.1 Terminology

Terminology in figure 9.5. A log entry (also called a log record) consists of many fields. Unfortunately, there is no universal standard for how log entries should be structured.

[3] grafana.com
[4] zabbix.com

Task Manager etc., we use *Windows Admin Center*

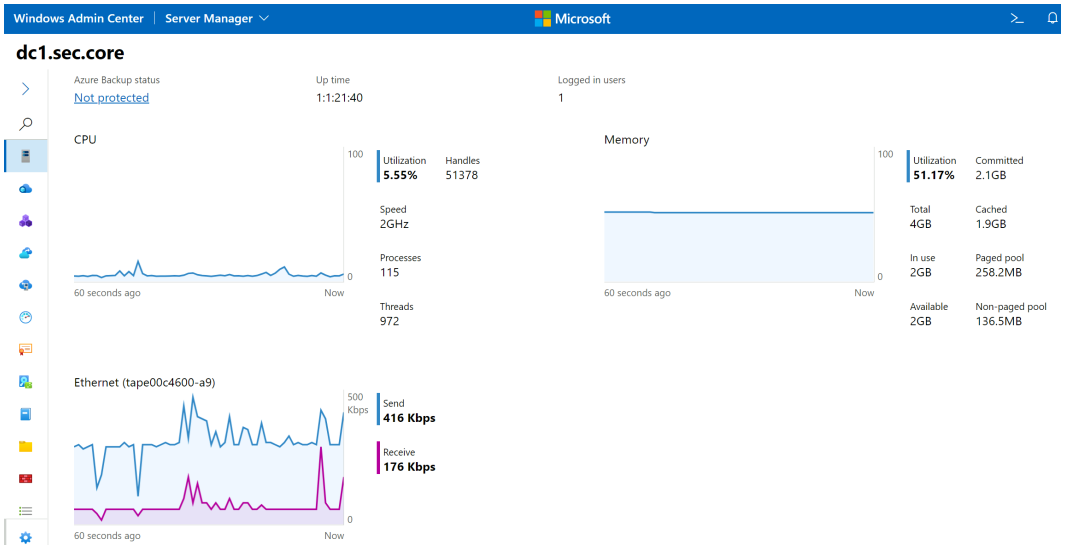


Figure 9.4: GUI Tools.

Event provider/source An application that generates events

Event A log entry

Event log A file containing events (Application, Security, System plus many app specific)

Event type Critical, Error, Warning, Information, Verbose, Debug, Success Audit, Failure Audit

Event ID A number representing specific event (*only unique per source/provider*)

Log mode Autobackup, Circular, Retain

Figure 9.5: Terminology.

However, most logs typically include certain commonly used fields, such as timestamp, hostname, process name or source/provider, and a message.

On Windows (as opposed to Linux), log entries also include an event ID and an event type. Some event logs (categories), such as Application, Security, and System, collect events from many different sources/providers, while many applications define and write to their own dedicated event logs, e.g.
'Microsoft-Windows-WindowsUpdateClient/Operational'

9.4.2 Log Files and Mode

Eventlogs can be in one of three different[5] `EventLogMode` (sometimes represented as the name e.g. "AutoBackup" and sometimes represented with a value corresponding to "AutoBackup" which is "1"):

AutoBackup (1) Archive the log when full, do not overwrite events. The log is automatically archived when necessary. No events are overwritten.

Circular (0) New events continue to be stored when the log file is full. Each new incoming event replaces the oldest event in the log.

Retain (2) Do not overwrite events. Clear the log manually rather than automatically.

Being aware of the log mode is important because log files can easily fill up a disk partition, especially when we increase the logging level for frequently used services (such as a web server). Logging every detail related to the server's processing consumes significant resources (both CPU time and disk space) and generates a very large number of events.

Log files are special files that typically have either a fixed size or a maximum size, giving us some control over how much disk space they occupy. However, each log mode behaves differently and comes with its own risks. Log mode AutoBackup creates new files when the log becomes full, which can significantly increase disk usage. Log mode Circular overwrites old events, which may result in losing historical information needed for analysis. Log mode Retain prevents overwriting, but this can lead to new events not being logged at all if the file reaches its maximum size.

Most companies want to keep log events for at least 90 days (three months) to ensure there is sufficient history for forensic investigations when an incident occurs.

We can see which logs use which mode with:
`Get-WinEvent -ListLog * | Format-Table -Property LogName,LogMode`
But how many logs are in each log mode? (Notice how we can use `Group-Object` to answer this instead of using a combination of `Where-Object` and `Measure-Object`)

[5] docs.microsoft.com/en-us/dotnet/api/system.diagnostics.eventing.reader.eventlogmode

```
PS>  Get-WinEvent -ListLog * | Group-Object -NoElement -Property LogMode
  Count Name
  ----- ----
    413 Circular
      2 Retain
```

In other words, most logs use the Circular log mode by default. Log entries are stored as records inside log files, and these log files are located in C:\Windows\System32\winevt\Logs\:

```
# How many logfiles?
Get-ChildItem C:\Windows\System32\winevt\Logs\ |
 Measure-Object

# How many of each size?
Get-ChildItem C:\Windows\System32\winevt\Logs\ |
   Group-Object -Property Length

# Which one most recently written to?
Get-ChildItem C:\Windows\System32\winevt\Logs\ |
 Sort-Object -Property LastWriteTime
```

The log files are managed and written to by the EventLog service which several other services depend on.

```
PS> (Get-Service EventLog).DependentServices

Status   Name               DisplayName
------   ----               -----------
Stopped  Wecsvc             Windows Event Collector
Stopped  NcdAutoSetup       Network Connected Devices Auto-Setup
Stopped  AppVClient         Microsoft App-V Client
Running  netprofm           Network List Service
Running  NlaSvc             Network Location Awareness
```

We can see that one of the services that depends on the EventLog service is the Windows Event Collector. This service can be used both to send log events to another host and to receive and store events from remote hosts. A professional logging system should collect log data from all hosts and forward it to a centralized processing server.

In NSM's "Grunnprinsipper for IKT-sikkerhet 2.1" [1], Chapter 3.2.4 states: "Beslutt hvilke data som er sikkerhetsrelevant og bør samles inn." In other words, security-relevant log events should be collected from all hosts.

```
  $id=(Get-WinEvent -LogName Security -MaxEvents 1).RecordId
  Get-EventLog -LogName Security |
    Where-Object {$_.Index -eq $id}

  Get-WinEvent -LogName Security |
    Where-Object {$_.RecordId -eq $id}

  Get-CimInstance Win32_NTLogEvent |
    Where-Object {$_.RecordNumber -eq $id}


We will use Get-WinEvent
```

Figure 9.6: Same Data from Three Cmdlets.

9.4.3 Three cmdlets

Same Data from Three Cmdlets in figure 9.6. There are three approaches in PowerShell to retrieving log events:

- `Get-EventLog`
- `Get-WinEvent`
- `Get-CimInstance Win32_NTLogEvent`

Sometimes it is faster and/or easier to use one cmdlet over another. The newest cmdlet – and probably the one you should try first – is `Get-WinEvent`, since it supports retrieving all event logs, unlike the older alternatives:

```
# How many logs does Get-WinEvent process?
PS> (Get-WinEvent -ListLog * |
      Where-Object {$_.RecordCount -gt 0} | Measure-Object).Count
109

# How many logs does Get-CimInstance process?
PS> (Get-CimInstance Win32_NTLogEvent |
     Select-Object -Property Logfile |
      Sort-Object -Property Logfile |
       Get-Unique -AsString | Measure-Object).Count
8

# How many logs does Get-EventLog process?
# (btw Get-EventLog is only in Windows PowerShell)
PS> (Get-EventLog -LogName * | Measure-Object).Count
11
```

- Most recent entries
- Find specific EventIDs
- Search all logs
- Ignore ”Information” level
- Search a specific time period

Figure 9.7: Typical Usage.

`Get-WinEvent` is the recommended cmdlet to use in PowerShell Core.

The three cmdlets return different types of objects, which also means that their property names differ. Because of this, `Get-Member` and `Select-Object -Property *` are useful tools for exploring and understanding these differences. If you want to retrieve a specific log entry, you must use different property names depending on which of the three cmdlets you are using. Below is an overview of the differences in the most important property names:

Field	Get-EventLog	Get-WinEvent	Get-CimInstance
Time stamp	`TimeGenerated`	`TimeCreated`	`TimeGenerated`
Host	`MachineName`	`MachineName`	(missing)
Source	`Source`	`ProviderName`	`SourceName`
Message	`Message`	`Message`	`Message`
Type	`EntryType`	`LevelDisplayName`	`Type`
Event ID	`InstanceId`	`Id`	`EventCode`

9.4.4 Using Get-WinEvent

Typical Usage in figure 9.7.

Most Recent

Most recent in the common logs (notice how `Get-WinEvent` merges the log events from all logs it processes):

```
Get-WinEvent -MaxEvents 10 -LogName `
  Application,System,Security,"Windows PowerShell" |
  Format-Table -Property `
   LogName,TimeCreated,RecordID,ID,LevelDisplayName,Message
```

Specific EventIDs

Finding specific EventIDs:

```
# Note the use of 'ExpandProperty'
# to see entire message

Get-WinEvent  -MaxEvents 10 -FilterHashtable `
  @{ LogName='Security'; Id='4672','4624' } |
  Select-Object -Last 1 -ExpandProperty Message
```

EventIDs are unique only within a source/provider. E.g. eventID 26 means FileDeleteDetected in the source/provider Sysmon, while it means "Connection to Exchange has been lost" in the source/provider Outlook. Some sources/providers write to the same logs, which means that the same eventID can represent different things within a single log. Because of this, it is advisable to include the source/provider when searching for a log entry with a specific eventID.

You can find lists of important event IDs in many places:

- https://adsecurity.org/?p=3299
- https://www.malwarearchaeology.com/cheat-sheets
- https://www.ultimatewindowssecurity.com/securitylog
- Threat Detection with Windows Event Logs[6]

If you find a resource that contains a list of event IDs you want to search for in your logs, you can copy and paste that list into a file – e.g. `a.txt` – and extract all the event IDs[7] into an array like this:

```
$IDs = Select-String -Pattern '\d{4}' .\a.txt -AllMatches |
 Select-Object -ExpandProperty Matches |
  Select-Object -Property Value
```

Later in this chapter you will learn about *regular expressions* like `\d{4}`.

[6] medium.com/adarma-tech-blog/threat-detection-with-windows-event-logs-59548f4f7b85

[7] This will only capture event IDs with four digits. There are also event IDs with fewer digits, as we will see when we discuss Sysmon.

Search all

Query all logs:

```
# This fails due to 256 max
Get-WinEvent -MaxEvents 10

$logs = (Get-WinEvent -ListLog * |
  Where-Object {$_.RecordCount} |
  Select-Object -ExpandProperty Logname)

# 25 most recent Warning and Information
Get-WinEvent -MaxEvents 25 `
  -FilterHashtable @{Logname=$logs;Level=3,4}
```

Ignore

Log entries are categorized according to different levels of "seriousness," which typically map to a corresponding numeric value (accessible through the `LevelDisplayName` property when using `Get-WinEvent`):

Verbose	5
Informational	4
Warning	3
Error	2
Critical	1
LogAlways	0

Ignore Information-level:

```
$Date = (Get-Date).AddDays(-2)

# Ignore "Information" by suppress level 4
$filter = @{
  LogName='Application'
  StartTime=$Date
  SuppressHashFilter=@{Level=4}
}

Get-WinEvent -FilterHashtable $filter
```

Time period

Find entries in a specific time period (remember *Filter left, Format right*):

```
$start = (Get-Date).AddDays(-1)
$end   = Get-Date

# DO THIS
Get-WinEvent -FilterHashtable `
 @{ logname="System"; starttime=$start; endtime=$end }

# DO NOT DO THIS (INEFFICIENT!)
Get-WinEvent -LogName System |
 Where-Object { $_.TimeCreated -gt $start `
  -and $_.TimeCreated -lt $end }
```

NSM's "Grunnprinsipper for IKT-sikkerhet 2.1" [1] chapter 3.2.4 is worth reciting in full:

> Beslutt hvilke data som er sikkerhetsrelevant og bør samles inn. For delene nevnte i 3.2.3 bør man som minimum samle inn a) data relatert til tilgangskontroll (vellykkede og mislykkede påloggинger) på enheter og tjenester og b) administrasjons- og sikkerhetslogger fra enheter og tjenester i IKT-systemene. For klienter bør man i tillegg som minimum registrere c) forsøk på kjøring av ukjent programvare (ref. 2.3.2) og d) forsøk på å få forhøyede systemrettigheter ("privilege escalation").

9.4.5 Extend logging with Sysmon

To improve detection of "forsøk på kjøring av ukjent programvare" (attempts to run unknown or unauthorized software), it is common on Windows hosts to enable additional logging using Sysmon from Sysinternals [47]. Sysmon installs as both a system service and a device driver, and it provides detailed logging of process creation, network connections, and file changes. Sysmon uses event IDs ranging from 1 to 28 (plus event ID 255 for "sysmon error") [48]. For example, event ID 1 is "Process Creation," and event ID 2 is "A process changed a file creation time." Sysmon can be configured with very fine-grained control, and a widely used configuration – either directly or as a starting point – is the configuration file created by SwiftOnSecurity [69]. To install Sysmon using SwiftOnSecurity's configuration file:

```
# Prerequisite is choco install sysinternals.

# download SwiftOnSecurity's configuration file
curl -O https://raw.githubusercontent.com/SwiftOnSecurity/sysmon-config/
        master/sysmonconfig-export.xml

# install with the config file we downloaded
```

```
sysmon.exe -accepteula -i sysmonconfig-export.xml

# check status of Sysmon
Get-Service sysmon

# query sysmon log
Get-WinEvent -MaxEvents 10 `
  -LogName "Microsoft-Windows-Sysmon/Operational"
```

9.5 Regular Expressions

Regular Expressions in figure 9.8. You may have noticed that the author has occasionally mentioned *regular expressions* (RegEx) without actually explaining what they are. It is now time to do so.

9.5.1 RegEx vs Wildcards

A regular expression is a combination of normal characters and special characters that we use to search through text. It is similar to using *wildcards*, but far more powerful and flexible.

Let's illustrate with an example: suppose we want to search for all filenames that end with the character `l` followed by exactly two additional characters[8]. In other words, we want to match filenames ending in `lnk`, `lds`, `lnt`, and so on. If we were to use wildcards, we could do this:
`Get-ChildItem -Recurse -File | Where-Object {$_.Name -like '*l??'}`
The only wildcards we actually use are

***** Match any character zero or more times.

? Match one character in that position.

If we were to use regular expressions, we would do this:
`Get-ChildItem -Recurse -File | Where-Object {$_.Name -match '.*l.{2}$'}`
Wildcards are essentially a very simple subset of the more general concept of regular expressions. We can interpret the regular expression `.*l.{2}$` as follows:

. The special character `.` matches any character (but just one character).

.* The special character `*` is a *quantifier*, it means "the previous character zero or more times". This means that the regular expression `.*` is the same as the wildcard `*`

[8] Note that this exact example would be most efficiently solved with `Get-ChildItem -Recurse -File -Filter '*l??'`, but efficiency is not the point here.

https://xkcd.com/208
https://creativecommons.org/licenses/by-nc/2.5

Figure 9.8: Regular Expressions.

Describing *a single* character:

Operator	Meaning
`.`	Any single character
`[abcd]`	One of these characters
`[^abcd]`	Any one but these characters
`[a-zA-Z0-9]`	A character in these ranges

Figure 9.9: Single Characters.

`.*l` This means the character `l` with anything in front of it.

`.*l.{2}` The special characters `{n}` is also a quantifier, it means "the previous character exactly `n` times".

`.*l.{2}$` The special character `$` is an *anchor*, it means "end of the line". There is also an anchor for "beginning of the line" of course, that is the special character `^`

You can read all the details about wildcards[9] and about regular expressions[10] in the documentation at Microsoft. But note the following important difference when it comes to which *operator* to use when applying wildcards or regular expressions:

> When using wildcards we use the operator `-like` (and the corresponding case-sensitive `-clike`), but if we use regular expressions we use the operator `-match` (and the corresponding case-sensitive `-cmatch`).

9.5.2 Basics of Regular Expressions

First, when learning regular expressions, consider using the excellent website regular expressions 101[11], which is created and maintained by Firas Dib from Sweden[12]. On this website, you can enter any regular expression and see, step by step, exactly how it is interpreted by the regular expression engine.

Note: in the left-hand menu, choose PCRE2 as the flavor and RegEx Debugger under Tools. Also make sure to disable all regex options in the text field where you enter the expression.

Single Characters in figure 9.9. This is how we can describe a single character. Say we want to describe a sequence of four characters:

1. The first character should be `A` or `B`.

[9] docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_wildcards
[10] docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_regular_expressions
[11] regex101.com
[12] github.com/sponsors/firasdib

2. The second character should not be `C`.
3. The third character should be a digit.
4. The fourth character can be anything but there must be a fourth character.

The regular expression `[AB][^C][0-9].` fulfills these requirements. Let's have some fun data to work with. Here is a list of ten million typical passwords:

```
curl -O https://erikhje.folk.ntnu.no/pw.txt
# Let's count how many lines there are by using
# a RegEx that matches anything:
PS> (Select-String '.*' pw.txt | Measure-Object).Count
1000000
```

If we were on Linux, we would use `grep` (which of course also supports regular expressions). In PowerShell, the equivalent of `grep` is `Select-String`. If we need to use the old `cmd` command line, the corresponding command is `findstr`. Using `Select-String`, we can apply the example above like this:

```
Select-String -Pattern '[AB][^C][0-9].' -Path pw.txt

# but we typically omit the parameters and just type
Select-String '[AB][^C][0-9].' pw.txt

# if we want the search to be casesensitive we have to do
Select-String '[AB][^C][0-9].' pw.txt -CaseSensitive
```

Since regular expressions contain special characters, we place them in single quotes (`'`) rather than double quotes (`"`). This ensures that none of the special characters are interpreted or expanded by PowerShell before the regular expression engine processes them.

`Select-String` also provides the option `-NotMatch`, which matches the opposite of the pattern defined by the regular expression. If we want to see whether there is more than one match in the input, we can use the `-AllMatches` option (since `Select-String` by default returns only the first occurrence of the pattern).

Sometimes you will encounter aliases for describing different character classes. We say "sometimes" because regular expression dialects differ across languages; here we include only what PowerShell supports. For example, you can use `\d` ("digit") instead of `[0-9]`, and `\w` instead of `[a-zA-Z_0-9]` ("word character"). To match the opposite of these, you can use the uppercase versions: `\D` ("not a decimal digit") and `\W` ("not a word character").

Anchoring:

Operator	Meaning
^	Beginning of line
$	End of line

Grouping:

Operator	Meaning
()	Group
|	OR

Figure 9.10: Anchoring and Grouping.

In addition, `\s` matches any whitespace character, `\S` matches any non-whitespace character, and `\t` matches the tab character.

Anchoring and Grouping in figure 9.10. The previous example `[AB][^C][0-9].` generated numerous matched lines, so maybe we need to narrow our search:

- The match should be at the beginning of the line:
  `^[AB][^C][0-9].`
- The match should be at the end of the line:
  `[AB][^C][0-9].$`
- The entire line should be only these four characters:
  `^[AB][^C][0-9].$`

Note that `^` at the beginning of a regular expression means "beginning of the line," while `^` inside square brackets (i.e. directly after `[`) means "none of these characters."

Let's modify our example slightly: instead of requiring the first character to be `A` or `B`, we now want the first three characters to be either `ABC` or `DEF`:
`(ABC|DEF)[^C][0-9].`

Quantifiers in figure 9.11. We have already seen the quantifier (sometimes called a repetition operator or modifier) `*` used in `.*`, meaning "any character zero or more times." Alternatives to `*` include `?` ("zero or one time") and `+` ("one or more times"). For example, when matching a valid email address, there must be at least one character both before and after the at-sign (`@`), so we use `+` instead of `*`.

The braces `{}` can be used to match an exact number of occurrences. In our context, this can be useful for checking password length:

- Any passwords that are at least twelve characters long:
  `^.{12,}$`

Repetition operators/Modifiers/Quantifiers:

Operator	Meaning
`?`	0 or 1 time
`*`	0 or more times
`+`	1 or more times
`{N}`	N times
`{N,}`	At least N times
`{N,M}`	At least N but not more than M

e.g. finding URLs:
`http[s]?://[^"]+`

e.g. each line should be an email address:
`^[A-Za-z0-9._-]+@[A-Za-z0-9.-]+$`

Figure 9.11: Quantifiers.

or
`.{12}`

- Any passwords that are at exactly twelve characters long:
  `^.{12}$`
- Any empty passwords:
  `^.{0}$`
  or
  `^$`
- Any passwords with seven characters or fewer:
  `^.{0,7}$`

One pitfall when using regular expressions is attempting to match everything that is allowed in a string instead of *inverting* the match – that is, describing what should not be matched. A classic example is matching URLs in a webpage. A URL begins with `http://` or `https://` followed by many possible characters that are difficult to describe precisely. However, what we do know is that the URL will typically end at the next double quote. Therefore, instead of trying to list all the characters that can appear in a URL, we simply state that after `http[s]?://` there will be one or more characters that are not a double quote. This allows us to write a simple and effective regex for matching URLs: `http[s]?://[^"]+`

Sometimes it is very useful to extract specific parts of the match. We can do this by marking the part we want with parentheses `( ... )` and then referring to it afterwards using `$matches[1]`, `$matches[2]`, and so on. The variable `$matches[0]` always contains the entire matched expression:

```
'zalo@oppvask.com','a@b@' | ForEach-Object {
if ($_ -match
  '^[A-Za-z0-9._-]+@([A-Za-z0-9.-]+)$') {
    Write-Output "Valid email: $($matches[0])"
    Write-Output "Domain is $($matches[1])"
  } else {
    Write-Output "Invalid email address!"
  }
}
# or
'zalo@oppvask.com','a@b@' -match '^[A-Za-z0-9._-]+@([A-Za-z0-9.-]+)$'
$Matches[1]
```

Regular expressions can also be used with the operators replace[13] and split[14], and with the switch statement with the `-regex` option[15]. When using the `-replace` operator we can *replace text based on captured text* similar to the use of the `matches` hashtable, but we refer to the captured text with `$1`, `$2`, ... instead, e.g.

```
PS> 'my dog 1 is a cat 2' -replace '(\d)(\D*)(\d)', '$3$2$1'
my dog 2 is a cat 1
```

How do we match the special characters themselves in PowerShell?

> If you want a literal match for one of the special characters, e.g. you want to search for passwords that contain `$` or `*`, you have to protect the special character with `\` or `` ` ``: `[\$\*]` or `[`$`*]`

9.5.3 LookAround Regular Expressions

"LookAround" Operators in figure 9.12. What is different about these "LookAround" regular expressions compared to what we have covered so far is that the regular expression engine does **not** move forward in the text when the match occurs. Instead, it matches a position – *a space between two characters*. This is such a fundamental point that it is worth repeating:

> *The regular expression engine does not move when doing the match; it simply matches a space between two characters.*

This is why they are called "Lookahead" and "Lookbehind." A *lookahead* checks what comes after the current position, while a *lookbehind* checks what comes before it — without consuming any characters in the match.

[13] docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_comparison_operators#replacement-operator

[14] docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_split

[15] learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_switch

Operator	Meaning
`(?=<pattern>)`	Lookahead. Matches the space between the character that comes before where that character is followed by `<pattern>`.
`(?<=<pattern>)`	Lookbehind. Matches the space between the character that comes after it where that character is preceded by `<pattern>`.
`(?!<pattern>)`	Negative lookahead.
`(?<!<pattern>)`	Negative lookbehind.

Figure 9.12: "LookAround" Operators.

The first use case for lookarounds is avoiding the need to "trim" (remove characters at the beginning or end of a string) from a regex result. Because lookarounds match only the position between characters and do not include anything in the matched text, they allow us to focus on exactly the part of the string we want.

For example, let's extract the digits (the price) from the string `BigCat is NOK15 at CC`. We know the digits are preceded by NOK, so we could write a regex like `NOK\d+`:

```
PS> 'BigCat is NOK15 at CC' -match 'NOK\d+'
True
PS> $Matches.Values
NOK15
```

The problem here is that NOK is included in the match. We could remove it afterwards by "trimming" the string using `($Matches.Values).Substring(3)`, which removes the first three characters from the matched value. However, it is better to extract only the digits directly from the match itself. We can achieve this by using a capture group:

```
PS> 'BigCat is NOK15 at CC' -match 'NOK(\d+)'
True
PS> $Matches.Values
15
NOK15
```

Now the digits 15 are stored in `$Matches[1]`, while the entire match is stored in `$Matches[0]`. The neat feature of lookarounds is that they allow the entire match to consist only of the part we are interested in:

```
PS> 'BigCat is NOK15 at CC' -match '(?<=NOK)\d+'
True
PS> $Matches.Values
15
```

In other words, the entire match is now only '15', because lookarounds (a lookbehind in this case) do not include any characters in the match itself — they simply "look around".

The second use case for lookarounds is checking for the presence of certain content within strings. We know that under normal circumstances we should **not use password composition requirements** [42]. However, suppose we are conducting research and want to evaluate the current state of an existing password database — for example, does most of the password dataset we downloaded earlier in the chapter contain uppercase letters, lowercase letters, and digits?

In this situation, we cannot use a regex like `[a-z]+[A-Z]+\d+`, because we cannot assume any specific sequence of characters. An uppercase letter may appear before a digit, after it, or anywhere else in the string. This is where lookahead expressions come to the rescue:

```
Select-String `
  -Pattern '(?=[^a-z]*[a-z])(?=[^A-Z]*[A-Z])(?=\D*\d)' `
  -Path pw.txt `
  -CaseSensitive |
    Select-Object -First 10
```

In other words, we use three lookaheads:

Lowercase `(?=[^a-z]*[a-z])` (from start of the string, match not a lowercase character zero or more times followed by a lowercase character).

Uppercase `(?=[^A-Z]*[A-Z])` (from start of the string, match not an uppercase character zero or more times followed by a lowercase character).

Digit `(?=\D*\d)` (from start of the string, match not a digit zero or more times followed by a digit).

These three checks can appear in any order, because the regex engine does not move when using lookarounds. Therefore, the regular expression `(?=[^a-z][a-z])(?=[^A-Z][A-Z])(?=\D*\d)` is equivalent to asking: "*Does the string contain at least one lowercase letter, at least one uppercase letter, and at least one digit?*"

If we do not want to use lookaheads, we could of course solve this directly in code instead, using something like the following (and remember to use `-cmatch` rather than `-match` when you need a case-sensitive comparison):

```
Get-Content .\pw.txt |
  ForEach-Object {
    if ($_ -cmatch '[a-z]') {
      if ($_ -cmatch '[A-Z]') {
```

```
        if ($_ -match '\d') {
          $_
        }
      }
    }
} | Select-Object -First 10
```

Also note that this could be expressed as one very long regular expression without lookarounds by listing all six permutations of the sequence lowercase, uppercase and digit with "or" as the separator:
`(.*[a-z].*[A-Z].*\d.*)|(.*[A-Z].*[a-z].*\d.*)|(...` and so on.
In these examples we have used `.*` instead of `[^a-z]`, `[^A-Z]` and `\D` to make it more readable. This does not scale well. With three conditions like we have here, this means $3! = 6$ possible combinations (permutations). If we increase to four conditions (for example, adding a requirement to check for special characters) the number jumps to $4! = 24$ permutations!

9.5.4 RegEx Performance

There are many more details involved in regular expressions than what we cover here, but understanding how the regular expression engine works is a topic well worth exploring. One important aspect to understand is that regular expressions are processed by a regular expression engine, which is a piece of software that takes the regular expression and the input text and determines whether there is a match. The way the engine processes the regular expression can have a significant impact on performance.

The quantifiers `*` and `+` are by default *greedy* meaning they will match as much as possible, e.g. `.*\d` will match the entire string including any digit before moving on and "backtracking" looking for a digit. You can make it *lazy* instead of greedy by adding a `?`, e.g. `.*?\d` will make `.*?` only match up until the first occurrences of a digit.

An even better approach in this case is to use contrast matching, which we briefly mentioned earlier. Instead of relying on greediness or laziness, we can write the pattern as `\D*\d`. This pattern matches any number of non-digit characters followed by a digit, which is more efficient because it avoids unnecessary backtracking.

If we use the RegEx Debugger at regular expressions 101[16] (note: in the left menu choose PCRE2 as flavor and RegEx Debugger under tools, and make sure to turn off all regex options in the text field where you enter the regex), we can see that if we match the test string `abc1def`, `.*\d` will match in seven steps, `.*?\d` will match in six steps and `\D*\d` will match in three steps. In other words, on paper this represents a significant difference in performance. In practice, performance rarely matters with today's computers, but we should still aim to follow best practices and avoid unnecessary computations

[16] regex101.com

whenever possible. For further reading, consider browsing the excellent resource *"The Elements of Good Regex Style"* [63].

9.5.5 RegEx and Logs

Regular expressions are extremely useful for searching through logs. They are especially valuable on Linux and when working with log data that lacks a defined structure, but they are also very useful for the Windows event log. We know that event log entries on Windows can be retrieved in PowerShell as objects with properties, and in most cases we can search these properties without needing regular expressions. However, the `Message`-property behaves differently from many other properties and is therefore a good candidate for the use of regular expressions.

The `Message`-property is slightly different because it is a *NoteProperty*, meaning it is created by PowerShell. In this context, it also means that certain values from the message can be extracted directly, without having to apply a regular expression to the entire message string. Consider the following PowerShell session:

```
PS> $LogEntry = Get-WinEvent -LogName Security -MaxEvents 1

PS> ($LogEntry | Get-Member -Name Message).MemberType
NoteProperty

PS> $LogEntry.Message
An account was logged off.

Subject:
        Security ID:            S-1-5-18
        Account Name:           DC1$
        Account Domain:         SEC
        Logon ID:               0x48B853A

Logon Type:                     3

This event is generated when a logon session is destroyed.
It may be positively correlated with a logon event using
the Logon ID value. Logon IDs are only unique between
reboots on the same computer.

PS> $LogEntry.Properties.Value[2]
SEC
```

The message appears to have some internal structure — and indeed it does. The values for *Security ID*, *Account Name*, *Account Domain*, and *Logon ID* can be retrieved from the `Value`-array inside the `Properties`-property. Which values exist depends on the `EventID` (the author is still searching for definitive documentation on this; there should be a Microsoft reference table somewhere describing which `EventID`s correspond to which values). We can observe that the number of values differs across event IDs:

```
PS> Get-WinEvent -LogName Security -MaxEvents 10 |
    Select-Object -Property `
    Id,@{Name="NumValues";Expression={$_.Properties.Value.Length}}

  Id NumValues
  -- ---------
4672         5
4624        27
5379        11
...
```

This means that in some cases we need to search the entire `Message`-property:

```
Get-WinEvent -MaxEvents 100 -LogName Security |
   ForEach-Object {
     if ($_.Message -match "Account Name:\s+Administrator.*")
       { Write-Output "$($_.Id): $($Matches[0])" }
   }
```

A very useful tool for processing the `Message`-property of log events is `Convert-EventLogRecord` (from the PowerShell module `PSScriptTools`, mentioned at the beginning of this chapter). This cmdlet extracts all values embedded in the `Message`-property and adds them to the LogEvent object as separate properties:

```
$LogEntry = Get-WinEvent -LogName Security -MaxEvents 1 |
              Convert-EventLogRecord

# instead of doing
PS> $LogEntry.Properties.Value[2]
SEC
# we can now do
PS> $LogEntry.TargetDomainName
SEC
```

With `Convert-EventLogRecord` we can now rewrite the previous example using `Get-WinEvent` in a much simpler and more readable way. While the logic is similar (but not identical), it could look like this:

```
Get-WinEvent -MaxEvents 100 -LogName Security |
  Convert-EventLogRecord |
    Where-Object {$_.SubjectUserName -match '[^\$]+\$' -or
                  $_.TargetUserName -match 'Administrator'} |
      Format-Table -Property Id,SubjectUserName,TargetUserName
```

The `SubjectUserName` property contains the name of the account that owns the process which generated the event, typically a system account. The `TargetUserName` property contains the username involved in the event itself. For example, in a logon event, the `SubjectUserName` might be `DC1$` (the domain controller's computer account), while the `TargetUserName` (the user who is actually logging on) might be `Administrator`.

9.6 Monitoring

Monitoring is about gaining an overview of the most important values and messages you have learned about in this chapter. A good rule of thumb for any general monitoring system is to begin by viewing the system from the user's perspective. For example: *"Test whether a product can be purchased in the webshop."*

Any monitoring system begins by retrieving data from hosts. This is done by placing *agents* (typically service processes) on the hosts, which collect and send data to a central service. Examples of such central services include Splunk and Elasticsearch. Notice how counters and log messages are related: if a counter exceeds a certain threshold, an agent-plugin may generate a log message, which is then collected by the central service. An agent-plugin may either read counter values directly or perform its own tests, such as checking whether a port is reachable (and generating a log message if it is not).

You will learn more about monitoring in the course *DCSG2003 - Robuste og skalerbare tjenester*. In this course, we will simply use Windows Admin Center as our "monitoring interface" for counters and log events. Windows Admin Center is a web-based interface for managing Windows servers, and it includes a built-in monitoring system that allows you to view performance counters and log events in real time.

9.7 Lab tutorials

1. **NOW LET'S USE A GUI!** Do this on MGR. Log in as domain administrator.

   (a) Install and launch Windows Admin Center

   ```
   choco install windows-admin-center
   & 'C:\Program Files\Windows Admin Center\SmeDesktop.exe'
   # note that first time it will search for updates
   # to extensions and install them, this takes
   # a few minutes
   ```

   (b) Add all the Windows hosts

      i. Add
      ii. Search Active Directory
      iii. Enter * in the search box (for servers) and DC1 and SRV1 should show up in the search, add both of them

   (c) Choose one of the hosts you have in Windows Admin Center, go to Performance Monitor

      i. Blank workspace
      ii. Add counter
      iii. Select object, `Processor`
      iv. Select instance, `_Total`
      v. Select counter, `% Processor Time`

   (d) Choose one of the hosts you have in Windows Admin Center, go to Events

      i. Choose "System" under "Windows Logs"
      ii. Sort by Level, can you find any log event at Error-level
      iii. View the details of an Error log event

9.8 Review questions and problems

1. What are the two main categories of data used in logging and monitoring, and how do they differ?
2. What is the difference between a periodic counter and an accumulating counter?
3. What is meant by the `CookedValue` of a counter, and why is it usually the most useful value?
4. What is the meaning of a counter path, and what is the special role of the instance `_Total`?
5. Which fields are commonly found in log entries, and which additional fields are especially important on Windows?
6. What are the three log modes discussed in the chapter, and what is the main risk associated with each one?
7. Why do organizations often want to retain logs for at least 90 days?
8. Why is centralized log collection considered important in a professional logging system?
9. Why is it not sufficient to search only for an eventID when analyzing logs?
10. What is Sysmon, and why is it useful in security monitoring?
11. What is the difference between the PowerShell operators `-like` and `-match`?
12. What do the anchors `^` and `$` mean in regular expressions, and why are they useful?
13. Why can lookaround expressions be useful compared to ordinary matching?
14. What is the difference between greedy and lazy quantifiers in regular expressions?
15. Why can contrast matching sometimes be better than relying on greedy or lazy matching?
16. What is the benefit of using `Convert-EventLogRecord` when working with Windows event logs?
17. How are counters and log messages related in practical monitoring systems?
18. (Search the Internet to solve this) Which group policy setting can you use to set the maximum file size for the Security event log on your hosts? What is the maximum file size you can set?

19. Use `Get-Counter -ListSet` to list all the counters in the CounterSet `Event Log`

20. Use `Get-Counter` to show the value in the counter `Events/Sec` from the CounterSet `Event Log`.

21. Use `Get-Counter` to show only the `CookedValue` in the counter `Events/Sec` every second for 10 seconds. The output should be like this (where xxxxx is your command line):

```
PS> xxxxx

CookedValue
-----------
10.9904350243983
6.76204890854735
9.88813063407143
8.87613268082581
12.8631527765561
                0
7.89924590836342
                0
15.6938306276679
7.77853121495407
```

    Repeat this exercise using `Get-MyCounter`.

22. Write a PowerShell script or function `Get-IOPS` which outputs the sum of disk reads and writes for the last second. You can find these two Counters in the CounterSet `PhysicalDisk`. If you want to know more about what IOPS is see https://en.wikipedia.org/wiki/IOPS

23. Use `Get-WinEvent` to list the 25 newest Security events. Choose one of them and output all properties in full text from that event.

24. Use `Get-WinEvent` to find all Security events that was logged the first hour after midnight on March 22. Hint: you can create a date object with
    `[datetime]$x = "03/22/2022"`

25. Use `Invoke-Command` to execute a search on `dc1` for System events during the last five days with `LevelDisplayName` Warning.

26. We know from Event Log Analysis Part 2 — Windows Forensics Manual 2018[17] that interesting events for Windows updates are Event IDs 19, 20, 43 and 44.

    (a) Create an array with the these Event IDs.

[17] medium.com/@lucideus/event-log-analysis-part-2-windows-forensics-manual-2018-75710851e323

(b) Use Get-WinEvent with the parameter -FilterHashtable to search the System eventlog for these Event IDs.

27. Download the file with ten million passwords mentioned in the chapter. Use `Select-String` to find passwords that

    (a) have more than 24 characters

    (b) have either the sequence `$$` or `!!`

    (c) have at least two `$`

    (d) only contain alphabetic characters (a..z and A..Z)

    (e) only contain alphabetic characters (a..z and A..Z) and are six characters or less long

28. Generate 1000 files with random filenames with the command lines

```
$tmpdir = New-Item -Type Directory -Name ('mytmp' + (Get-Random))
cd $tmpdir
(1..1000) | ForEach-Object {
  New-Item -Type File -Name ([System.IO.Path]::GetRandomFileName())
}
```

    (a) Replace the file endings with `.dat` using the -replace operator[18]

    (b) How many of these filenames contain at least two digits and both the characters x and y? Use the regex construction lookahead to solve this.

29. The scary biker gang MC-Donalds are considering selling a new drug. Our intelligence unit has been able to retrieve a file with secret internal communication from their network. You can download this file with
    `curl -O https://erikhje.folk.ntnu.no/MC.txt`
    The new drug has a familiar name and is mentioned quite often in the file, but many times in a slightly obfuscated way, but *what we do know is it is always surrounded by double quotes* (`"`). Your job is to write the regular expression using lookaround constructions and complete the Select-String command with the correct options in the following PowerShell session (in other words, you need to replace the dots).

```
PS> $MyMatches = Select-String .....
PS> $MyMatches.Matches.Value
pizza
pitsa
pitza
pit5a
```

[18] learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_comparison_operators

```
pizza
pitsa
pitza
pit5a
MC-Pizza
Pitza5$
MC-Pizza
PS>
```

30. On DC1, use `Get-WinEvent` to search for all EventID 4624 in the Security eventlog (restrict to the last 200 log entries to avoid too long output), then add the following pipeline to the command:

    - `Convert-EventLogRecord`
    - use `Where-Object` to select only those log events that have `$_.TargetUserName` that starts with `Adm`,
    - expand the Message property of those log events,
    - pipe to `Select-String` which should search for the text "Logon process:" followed by one or more white space followed by either the text "Kerberos" or "NtLmSsp", add the parameter `-AllMatches` to Select-String
    - finally use ForEach-Object to print `$_.Matches.Value`

    The output should be something like:

```
Logon Process:          Kerberos
Logon Process:          Kerberos
Logon Process:          Kerberos
Logon Process:          NtLmSsp
Logon Process:          Kerberos
Logon Process:          Kerberos
Logon Process:          Kerberos
Logon Process:          Kerberos
Logon Process:          Kerberos
Logon Process:          Kerberos
```

10 Security: Attacks

10.1 NSM Grunnprinsipper

NSM Grunnprinsipper for IKT-sikkerhet 2.1 in figure 10.1. To protect an ICT infrastructure we need to know which threats we face and how we can mitigate those. This is covered in NSM chapter 3.1 ”Oppdag og fjern kjente sårbarheter og trusler”.

10.2 Introduction

10.2.1 Asset Value

The starting point for addressing security is to ask: *which assets are we trying to protect, and what is the value of each asset?* We have previously mentioned that one of the goals of Active Directory is asset management, which makes it a natural place to begin. What would happen to the company if Active Directory became unavailable? What if its data were stolen? What if its data were deleted? The Active Directory service is itself an asset that must be protected, and it is likely to be a high-value asset.

This also highlights an important point: we must protect both data (whether at *rest* or in *transit*) and services/processes. Technical assets (e.g. an operating system process, such as the services that make up Active Directory) may be valuable in their own right, but they may also be highly valuable because they can be used to access or affect other high-value assets (e.g. a water supply system). In our context, examples of assets that we are trying to protect include:

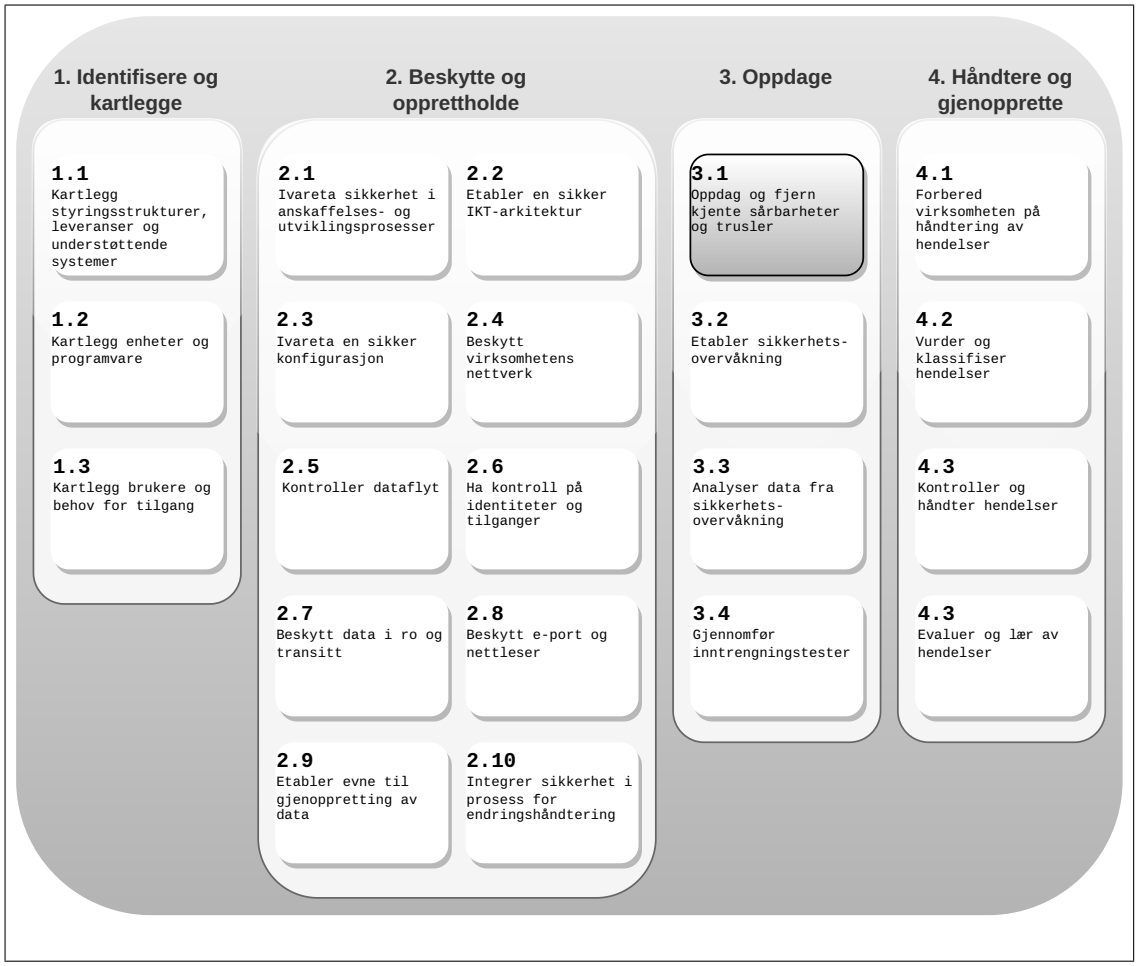


Figure 10.1: NSM Grunnprinsipper for IKT-sikkerhet 2.1.

- Accounts.
- Services/processes.
- User and company data (including backups).

10.2.2 Risk Assessment

Once you know which assets need to be protected and understand their value, the next step is to conduct a risk assessment in order to determine where protection efforts and resources should be prioritized. An *information security risk* is the result of the interaction between three factors: *threat*, *vulnerability*, and *asset*. More generally, risk is often described as the product of likelihood and consequence. Based on risk assessments, an organization develops an information security management system.

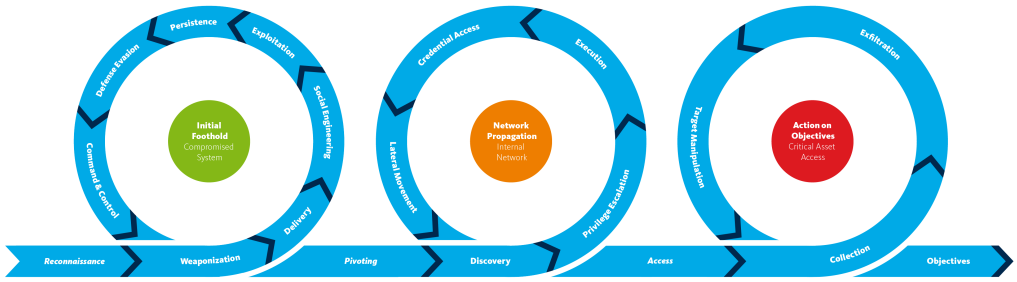


Figure 10.2: The Unified Kill Chain.

You will learn more about this in later courses, but we introduce it here so that you understand the broader context. In this course, we focus on one part of that larger picture: understanding threats and how to mitigate them in a Windows infrastructure.

10.3 Threats

10.3.1 Cyber Kill Chain

The Unified Kill Chain in figure 10.2. (Source of figure[1]). For us to be effective at protecting a Windows infrastructure, we need to understand how attacks are carried out. A well-known model for describing this is the *Cyber Kill Chain*, originally developed by Lockheed Martin in 2011 and later extended and combined with Mitre Att&ck in the Unified Kill Chain by Paul Pols [73]. The kill chain gives us an overview of the typical sequence of an attack. To defend against such attacks, however, we need to understand in more detail how they are actually implemented. This is where Mitre Att&ck is useful: it is a lower-level model than the Cyber Kill Chain and helps explain what lies ”behind” each step of the kill chain.

In practice, this means that a typical attack might proceed roughly as follows (in chronological order and in a highly simplified form):

1. Gain access to a user account on a computer (perhaps only a local account without administrator privileges).
2. Escalate privileges in order to become a local administrator.
3. As a local administrator, install software that runs as a service and waits for a domain user to log in. When a domain user logs in, steal that user's credentials.

[1] en.m.wikipedia.org/wiki/File:The_Unified_Kill_Chain.png

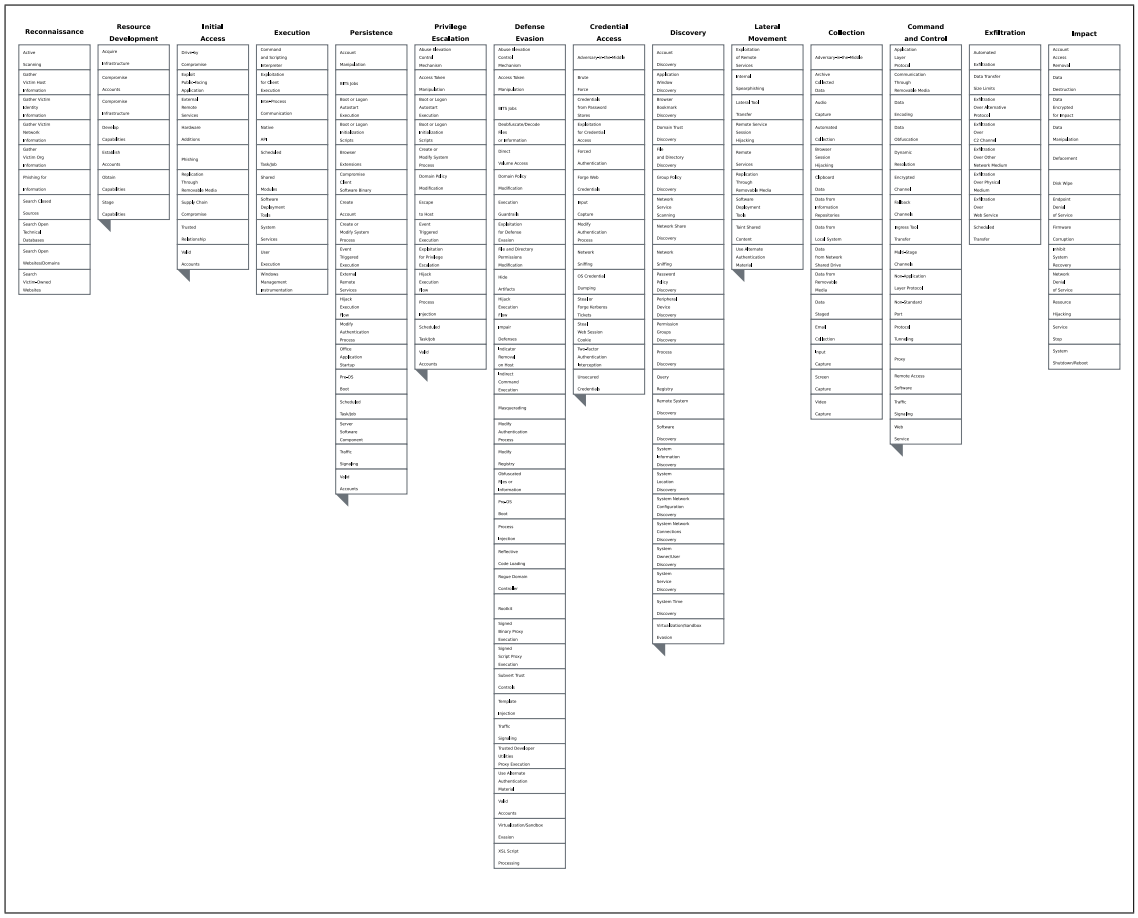


Figure 10.3: Mitre Att&ck Matrix.

4. Once you have access to a domain user account, you can access other machines (move laterally).

5. Find a way to escalate from a domain user account to domain administrator.

6. As a domain administrator, deploy malware through Group Policy or another remote management mechanism (or simply steal data).

10.3.2 Mitre Att&ck

Mitre Att&ck Matrix in figure 10.3. To understand and gain an overview of the threat landscape, we can use the Mitre Att&ck knowledge base [67]. Mitre Att&ck is based on observed activities from known threat actors (groups of people), including those referred to as *APTs (Advanced Persistent Threats)*. Someone who may attack us—such as an APT group or another type of actor (individuals, groups, companies, organizations, or countries)—is referred to as an *adversary* (a synonym for "opponent" or "enemy").

Mitre collects information from known attacks and incorporates that information into the knowledge base. We can use Mitre Att&ck to learn about common attack patterns that have been used against infrastructures similar to ours.

Mitre distinguishes between different *technology domains*: enterprise, mobile, and ICS (industrial control systems). In this book, we focus on the enterprise technology domain, since that is the relevant domain for our Windows infrastructure. The knowledge base is structured around *TTPs (Tactics, Techniques, and Procedures)* (note: the example texts are quotations copied directly from the Mitre Att&ck knowledge base[2]):

Tactic represents the reason for performing an action, as of January 2024 there are 14 enterprise tactics in Mitre Att&ck, e.g. TA0001 Initial Access[3]:

> Initial Access consists of techniques that use various entry vectors *to gain their initial foothold within a network*. Techniques used to gain a foothold include targeted spearphishing and exploiting weaknesses on public-facing web servers. Footholds gained through initial access may allow for continued access, like valid accounts and use of external remote services, or may be limited-use due to changing passwords.

Technique represents how to achieve a tactical objective, as of March 2022 there are 188 enterprise techniques in Mitre Att&ck, e.g. T1195 Supply Chain Compromise[4]:

> Adversaries may manipulate products or product delivery mechanisms prior to receipt by a final consumer for the purpose of data or system compromise. Supply chain compromise can take place at any stage of the supply chain including:
> - Manipulation of development tools
> - Manipulation of a development environment
> - Manipulation of source code repositories (public or private)
> - Manipulation of source code in open-source dependencies
> - Manipulation of software update/distribution mechanisms
> - Compromised/infected system images (multiple cases of removable media infected at the factory)
> - Replacement of legitimate software with modified versions
> - Sales of modified/counterfeit products to legitimate distributors
> - Shipment interdiction

(Sub-technique) is present when there are multiple ways of performing a specific technique, as of January 2024 there are 424 enterprise sub-techniques in Mitre Att&ck, e.g. T1195.001 Supply Chain Compromise: Compromise Software Dependencies and Development Tools[5]:

[2] attack.mitre.org
[3] attack.mitre.org/versions/v14/tactics/TA0001
[4] attack.mitre.org/versions/v14/techniques/T1195
[5] attack.mitre.org/versions/v14/techniques/T1195/001

> Adversaries may manipulate software dependencies and development tools prior to receipt by a final consumer for the purpose of data or system compromise. Applications often depend on external software to function properly. Popular open source projects that are used as dependencies in many applications may be targeted as a means to add malicious code to users of the dependency.
>
> Targeting may be specific to a desired victim set or may be distributed to a broad set of consumers but only move on to additional tactics on specific victims.

Procedure is the implementation of techniques/sub-techniques that have been used by threat actors, e.g. for supply chain attack:

> XCSSET adds malicious code to a host's Xcode projects by enumerating CocoaPods `target_integrator.rb` files under the `/Library/Ruby/Gems` folder or enumerates all `.xcodeproj` folders under a given directory. XCSSET then downloads a script and Mach-O file into the Xcode project folder.

Tactics have an ID number of the form `TAxxxx`, techniques have an ID number of the form `Txxxx`, and sub-techniques (which may or may not exist for a given technique) have an ID number of the form `Txxxx.xxx`. In the Mitre Att&ck data model, tactics and techniques/sub-techniques are treated as objects. Procedures do not have a separate ID number; instead, they are represented as optional attributes of a technique or sub-technique object.

All the Tactics

From Enterprise tactics[6] we can read that the tactics are described as:

1. Reconnaissance: *The adversary is trying to gather information they can use to plan future operations.*
2. Resource Development: *The adversary is trying to establish resources they can use to support operations.*
3. Initial Access: *The adversary is trying to get into your network.*
4. Execution: *The adversary is trying to run malicious code.*
5. Persistence: *The adversary is trying to maintain their foothold.*
6. Privilege Escalation: *The adversary is trying to gain higher-level permissions.*

[6] attack.mitre.org/tactics/enterprise

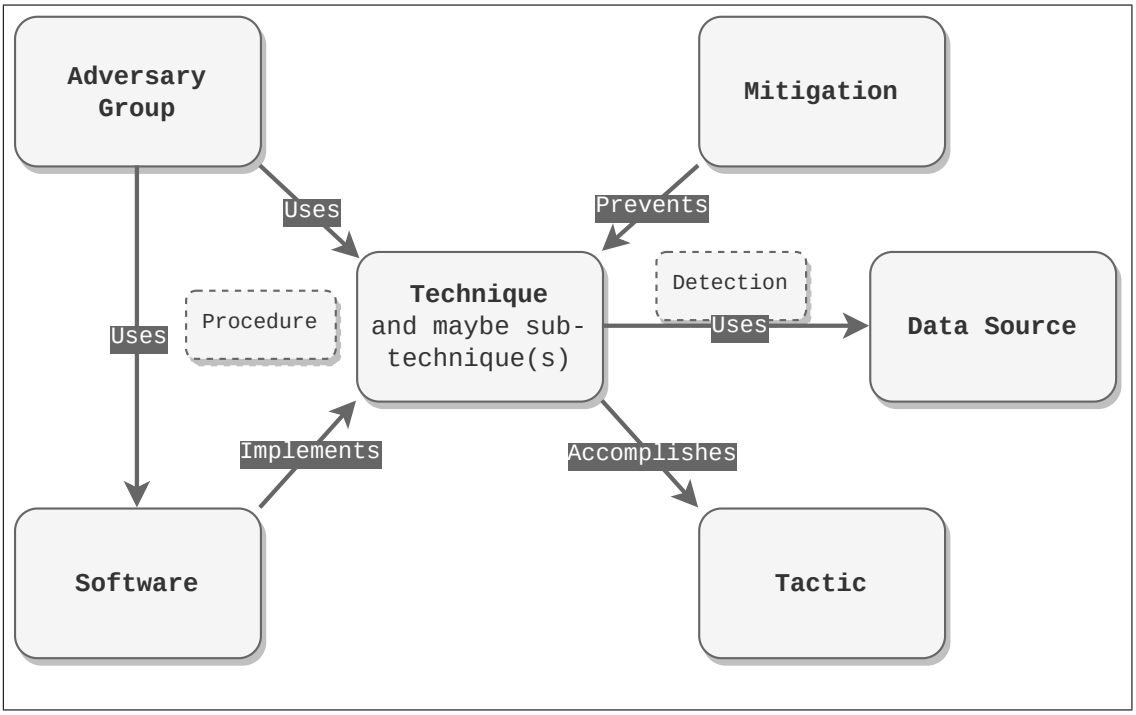


Figure 10.4: Mitre Att&ck Data Model.

7. Defense Evasion: *The adversary is trying to avoid being detected.*
8. Credential Access: *The adversary is trying to steal account names and passwords.*
9. Discovery: *The adversary is trying to figure out your environment.*
10. Lateral Movement: *The adversary is trying to move through your environment.*
11. Collection: *The adversary is trying to gather data of interest to their goal.*
12. Command and Control: *The adversary is trying to communicate with compromised systems to control them.*
13. Exfiltration: *The adversary is trying to steal data.*
14. Impact: *The adversary is trying to manipulate, interrupt, or destroy your systems and data.*

Att&ck Model

Mitre Att&ck Data Model in figure 10.4. The figure is a slightly modified version of

Figure 3 in Strom et al. [67]. In addition to tactics, techniques/sub-techniques, and procedures (TTPs), the model includes four additional types of objects (note again that the example texts are quotations copied directly from the Mitre Att&ck knowledge base[7]):

Adversary group is a threat actor known by a common name in the information security community, as of January 2024 there are 143 adversary groups in Mitre Att&ck, e.g. G0032 Lazarus group[8]:

> Lazarus Group is a North Korean state-sponsored cyber threat group that has been attributed to the Reconnaissance General Bureau. The group has been active since at least 2009 and was reportedly responsible for the November 2014 destructive wiper attack against Sony Pictures Entertainment as part of a campaign named Operation Blockbuster by Novetta. Malware used by Lazarus Group correlates to other reported campaigns, including Operation Flame, Operation 1Mission, Operation Troy, DarkSeoul, and Ten Days of Rain. North Korean group definitions are known to have significant overlap, and some security researchers report all North Korean state-sponsored cyber activity under the name Lazarus Group instead of tracking clusters or subgroups, such as Andariel, APT37, APT38, and Kimsuky.

Software is a tool or malware that is used to conduct behaviour modeled in the TTPs, as of January 2024 there are 760 software tools and malware in Mitre Att&ck, e.g. S0154 Cobalt Strike[9]:

> Cobalt Strike is a commercial, full-featured, remote access tool that bills itself as "adversary simulation software designed to execute targeted attacks and emulate the post-exploitation actions of advanced threat actors". Cobalt Strike's interactive post-exploit capabilities cover the full range of ATT&CK tactics, all executed within a single, integrated system. In addition to its own capabilities, Cobalt Strike leverages the capabilities of other well-known tools such as Metasploit and Mimikatz.

Mitigation is a security concept or technology that can be used to protect against techniques/sub-techniques, as of January 2024 there are 43 enterprise mitigations in Mitre Att&ck, e.g. M1032 Multi-factor Authentication[10]:

> Use two or more pieces of evidence to authenticate to a system; such as username and password in addition to a token from a physical smart card or token generator.

[7] attack.mitre.org
[8] attack.mitre.org/versions/v14/groups/G0032
[9] attack.mitre.org/versions/v14/software/S0154
[10] attack.mitre.org/versions/v14/mitigations/M1032

Data source (including the sub-category "Data Components") is a source of information related to techniques/sub-techniques that can be collected by sensors/logs, as of January 2024 there are 41 data sources in Mitre Att&ck, e.g. DS0028 Logon Session[11]:

> Logon occurring on a system or resource (local, domain, or cloud) to which a user/device is gaining access after successful authentication and authorization. Data Components are 1. *Logon Session: Logon Session Creation*: Initial construction of a new user logon session (ex: Windows EID 4624, /var/log/utmp, or /var/log/wmtp) and 2. *Logon Session: Logon Session Metadata*: Contextual data about a logon session, such as username, logon type, access tokens (security context, user SIDs, logon identifiers, and logon SID), and any activity associated within it.

Similar to TTPs, adversary groups have an ID number of the form `Gxxxx`, software has an ID number of the form `Sxxxx`, mitigations have an ID number of the form `Mxxxx`, and data sources have an ID number of the form `DSxxxx`. To be precise, a *data source* is formally defined as a required attribute of a technique object, so it is not an object in its own right in the Mitre Att&ck data model. However, in our slightly modified version of the model, we treat it as a separate object in order to simplify and visualize the relationship between a technique and its associated data source.

Att&ck Example

Mitre Att&ck Model Example in figure 10.5. The Mitre Att&ck model can represent attack behavior in considerable detail. For example, we can model a situation in which the adversary group *APT32* uses the *pass-the-ticket* sub-technique to achieve the tactic *lateral movement*. This may be carried out by using the tool Mimikatz, with a specific procedure involving the Mimikatz modules `LSADUMP::DCSync` and `KERBEROS::PTT`. We may be able to reduce the risk of this behavior by applying mitigations such as M1026 "Privileged Account Management", and we may be able to detect it through Event ID 4769 or other events related to the data sources Active Directory, logon sessions, and user accounts. In other words, the model helps us see how these concepts relate to one another, which makes it easier to understand threats and to reason about how to protect our infrastructures.

Mitre CAPEC

When you browse the Mitre Att&ck matrix and view techniques, you will sometimes see that a technique links to a CAPEC ID number. For example, for the sub-technique

[11] attack.mitre.org/versions/v14/datasources/DS0028

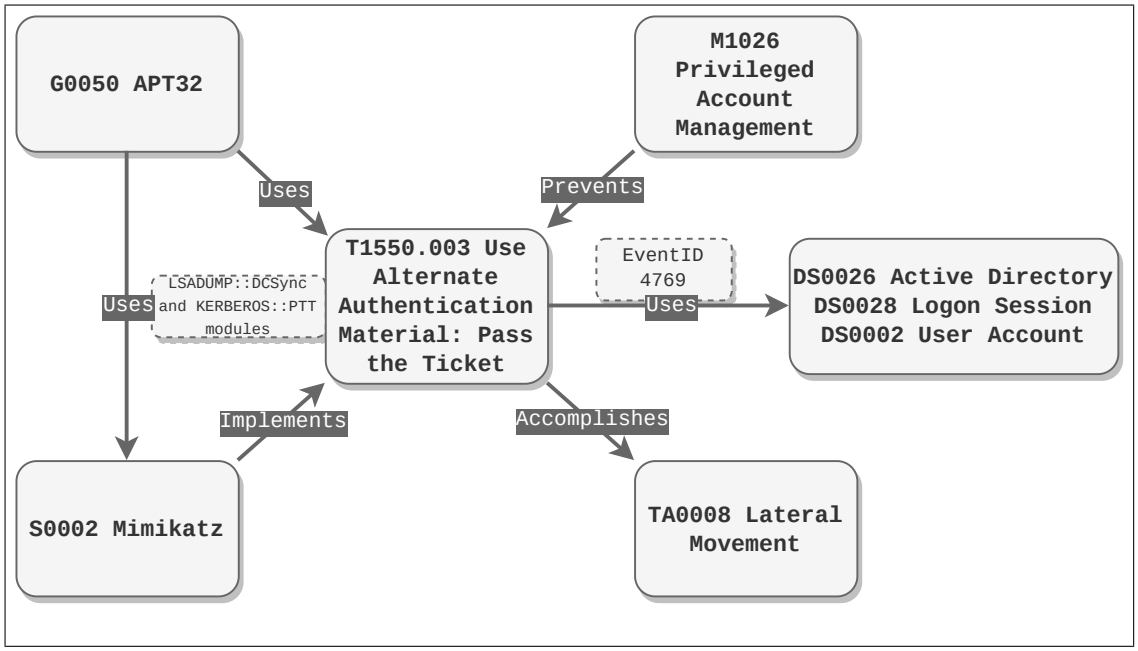


Figure 10.5: Mitre Att&ck Model Example.

"Boot or Logon Autostart Execution: Winlogon Helper DLL", there is a link to CAPEC-579, which is "Replace Winlogon Helper DLL". CAPEC is an acronym for *Common Attack Pattern Enumeration and Classification*. Mitre describes the differences between CAPEC and Att&ck as follows[12]:

CAPEC "CAPEC is focused on application security and describes the common attributes and techniques employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. (e.g., SQL Injection, XSS, Session Fixation, Clickjacking)"

Att&ck "ATT&CK is focused on network defense and describes the operational phases in an adversary's lifecycle, pre- and post-exploit (e.g., Persistence, Lateral Movement, Exfiltration), and details the specific tactics, techniques, and procedures (TTPs) that advanced persistent threats (APT) use to execute their objectives while targeting, compromising, and operating inside a network."

Att&ck is primarily intended for practitioners who work with infrastructure and security, whereas CAPEC is aimed more at developers working with application security. In some cases, a CAPEC attack pattern is reflected in an Att&ck technique or subtechnique, so the two are related. However, they serve different purposes and are designed for different audiences.

[12] capec.mitre.org/about/attack_comparison.html

Critique

Mitre Att&ck is widely used and highly regarded in industry. One of its greatest strengths is that it provides a common language: by referring to technique IDs, we can discuss the same attack techniques in a precise and consistent way. We can also think of it as a menu from which attackers choose and combine different elements to build an attack. It is therefore very useful for learning the broader picture of the many kinds of attack behavior that we need to be aware of. At the same time, it helps us recognize that attack behavior is often difficult to distinguish from normal behavior (e.g. checking your IP address). Mitre Att&ck encourages us to focus on techniques rather than on signatures or exact procedures, which is important in the security industry: we need to detect higher-level behavior, not just low-level detailed signatures, because attackers can often bypass signature-based detection such as standard antivirus software.

Mitre Att&ck is not a checklist that can be used to claim, for example, that you have achieved 90% protection simply because you have run a test suite covering everything in the matrix. Threat actors are often more advanced and adaptive than what can be represented by the combinations you are able to compose yourself in the Att&ck matrix. It is a good starting point, but it is important to remember that attackers may do more than what is explicitly listed in the matrix. Furthermore, even if we attempt to detect the behavior described by the techniques, this is often very difficult to achieve in practice. In other words, Mitre Att&ck is excellent for learning, for gaining an overview, and for establishing a common language, but we must also be aware of the limits of the model (as with all models, there are always limitations).

10.3.3 Using Att&ck Navigator

Difficulty Levels in figure 10.6. Mitre Att&ck includes a very useful tool called the Attack Navigator[13], which we can use to browse the matrix and explore the data model. At the same time, the tool can be somewhat overwhelming when you first encounter it. Where should we begin in order to learn some of this? Fortunately, Travis Smith, a principal security researcher at Tripwire, has carried out the very useful exercise of categorizing the techniques by difficulty level. The colors in the matrix represent (quotation from TravisFSmith/mitre_attack[14]):

1. *Blue* These are techniques which are not really exploitable, rather they use other techniques to be viable.
2. *Green* These are the easiest techniques to exploit, there is no need for POC malware, scripts, or other tools.
3. *Yellow* These techniques usually need some sort of tool, such as Metasploit.

[13] mitre-attack.github.io/attack-navigator
[14] github.com/TravisFSmith/mitre_attack

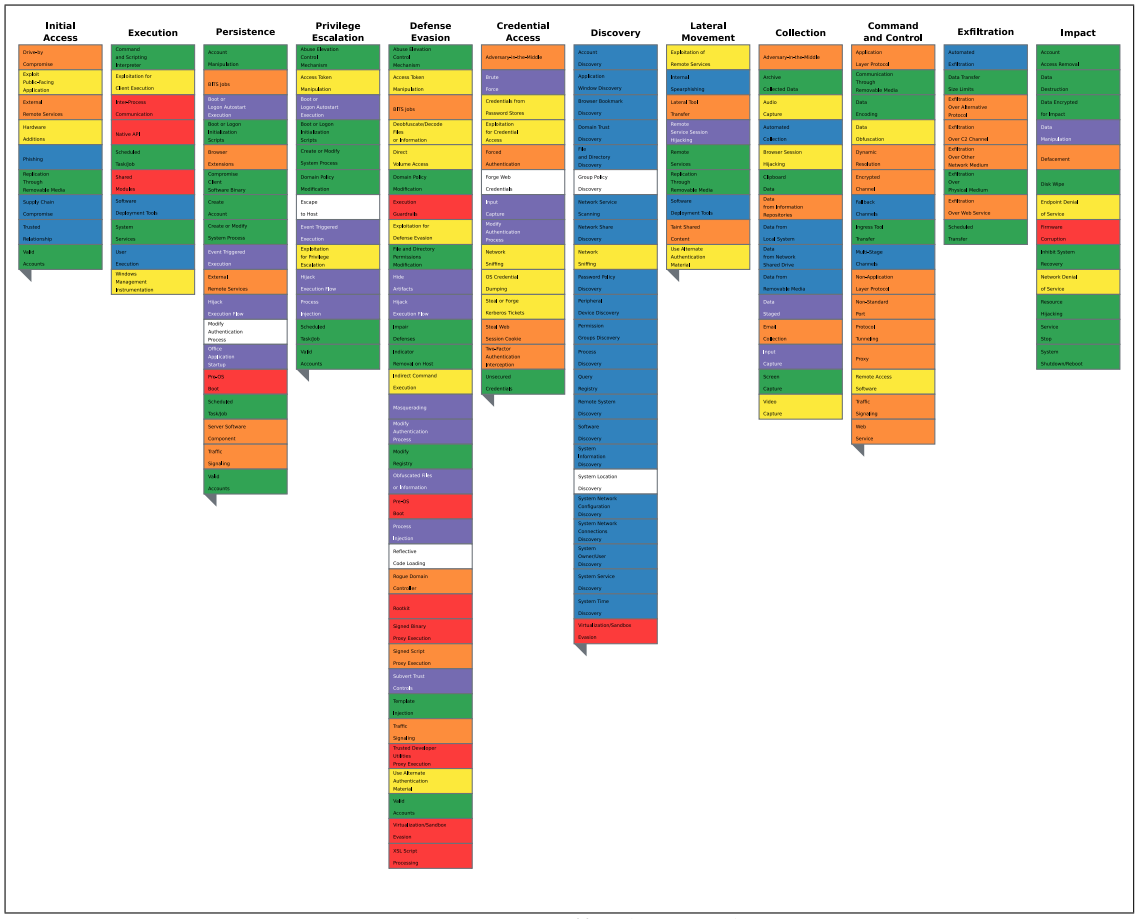


Figure 10.6: Difficulty Levels.

4. *Orange* These techniques require some level of infrastructure to setup. Once setup, some are easy and some are more advanced.
5. *Red* These are the most advanced techniques which require an in-depth understanding of the OS or custom DLL/EXE files for exploitation.
6. *Purple* These are high level techniques which include sub-techniques of varying levels.
7. (White are not categorized as they have appeared after Travis did his work)

10.4 Testing and Analysis

10.4.1 Atomic Red Team

Atomic Red Team in figure 10.7. Atomic Red Team[15] is a collection of tests that can be

```
PS C:\Users\Administrator> Invoke-AtomicTest T1222.001 -TestNumbers 5 -CheckPrereqs
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Using Logger:  Default-ExecutionLogger
All logging commands found
CheckPrereq's for: T1222.001-5 Grant Full Access to folder for Everyone - Ryuk Ransomware Style
Prerequisites met: T1222.001-5 Grant Full Access to folder for Everyone - Ryuk Ransomware Style
PS C:\Users\Administrator> Invoke-AtomicTest T1222.001 -TestNumbers 5 -ShowDetailsBrief
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Using Logger:  Default-ExecutionLogger
All logging commands found
T1222.001-5 Grant Full Access to folder for Everyone - Ryuk Ransomware Style
PS C:\Users\Administrator> Invoke-AtomicTest T1222.001 -TestNumbers 5
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Using Logger:  Default-ExecutionLogger
All logging commands found
Executing test: T1222.001-5 Grant Full Access to folder for Everyone - Ryuk Ransomware Style
Successfully processed 24 files; Failed processing 0 files
Done executing test: T1222.001-5 Grant Full Access to folder for Everyone - Ryuk Ransomware Style
PS C:\Users\Administrator> Invoke-AtomicTest T1222.001 -TestNumbers 5 -Cleanup
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Using Logger:  Default-ExecutionLogger
All logging commands found
Executing cleanup for test: T1222.001-5 Grant Full Access to folder for Everyone - Ryuk Ransomware Style
Done executing cleanup for test: T1222.001-5 Grant Full Access to folder for Everyone - Ryuk Ransomware Style
```

Figure 10.7: Atomic Red Team.

used to emulate the attack behaviors described by the techniques and sub-techniques in the Mitre Att&ck matrix. Note that these tests are not "exploits" in themselves, but they do make changes to the system, so they must be used with care. The main benefit of, and reason for, being aware of this collection is that it is designed based on Mitre Att&ck, which means that the tests map directly to individual techniques and sub-techniques. Atomic Red Team has two components: a collection of tests and an execution framework. In our case, we will of course use the PowerShell framework, which allows us to execute the tests with the cmdlet `Invoke-AtomicTest`. The tests are organized by technique and sub-technique, and each test has a unique test number. For example, the test for the sub-technique "Boot or Logon AutoStart: Registry Run Keys" is T1547.001, and the test number for the specific test we will run is 3. By using the test numbers, we can execute specific tests and also clean up after them when we are done.

10.4.2 BloodHound

BloodHound in figure 10.8. BloodHound [60] is a popular analysis tool for Active Directory environments. To use BloodHound for analysis, you first run SharpHound, which collects data from Active Directory and generates JSON files that are compressed into a ZIP archive. BloodHound can then load this ZIP archive and draw a graph of the Active Directory environment using *nodes* and *edges*. Nodes can represent users, groups, computers, domains, GPOs, and OUs (and, when used in an Azure environment, addi-

[15] atomicredteam.io

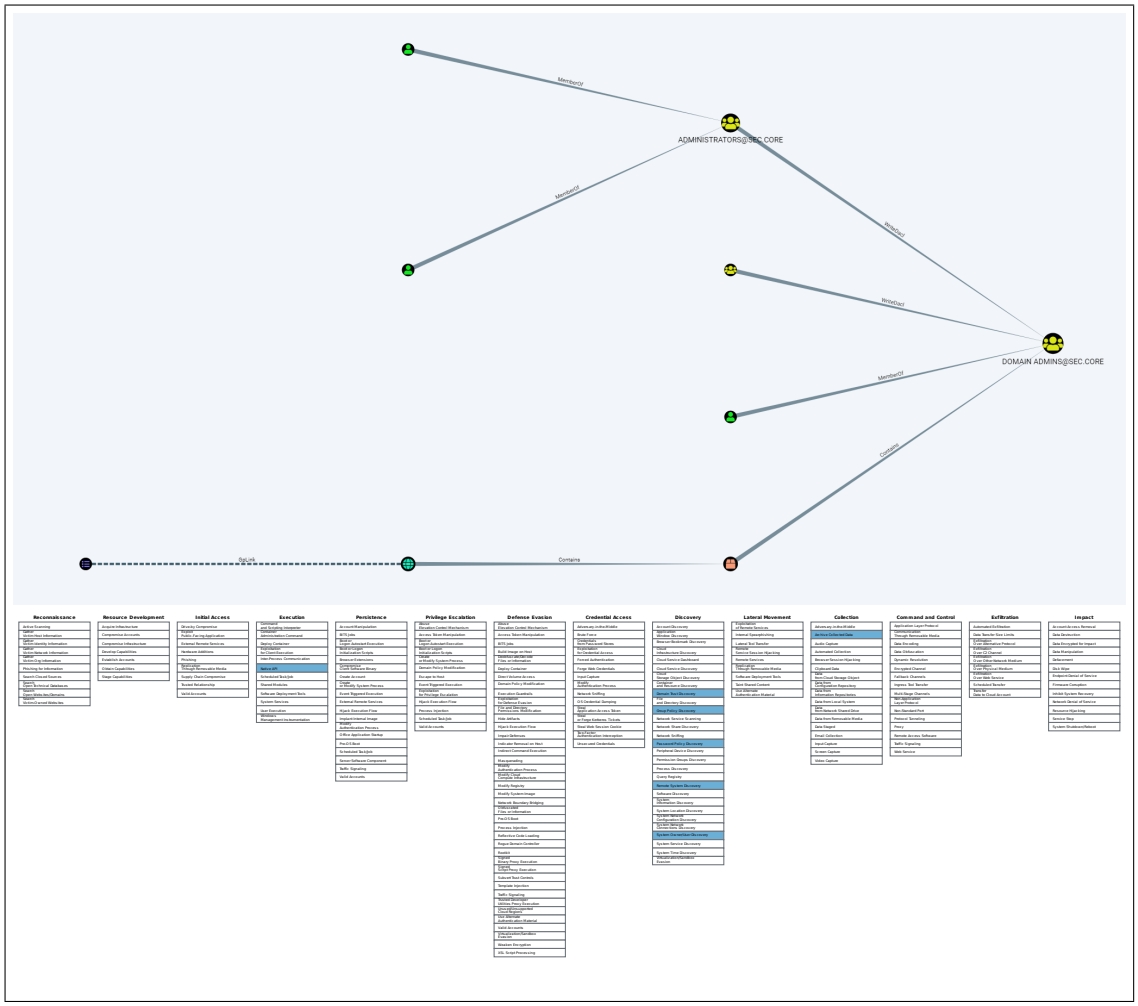


Figure 10.8: BloodHound.

tional node types are supported, such as AzTenant). The set of possible edges is much larger than the set of possible nodes. Examples of edges include MemberOf, GPLink, Contains, WriteDACL, Owns, CanRDP, and CanPSRemote. All of these are described in the BloodHound documentation[16].

10.4.3 Other Tools

There is a wide range of tools available for security analysis, especially in the context of penetration testing. Two tools that the author knows are used by leading security companies are Burp Suite and Nessus. These are, however, only two examples. Most professionals who work with security analysis have their own toolbox, which may in-

[16] bloodhound.readthedocs.io/en/latest/data-analysis/edges.html

clude Nessus and/or Burp, but is often broader than that. To see what else may be included in such toolboxes, it can be useful to search the internet for cheat sheets, for example Pentest-Cheat-Sheets[17].

[17] github.com/Kitsun3Sec/Pentest-Cheat-Sheets

10.5 Lab tutorials

1. **Atomic Red Teams.** Log into DC1 as domain administrator. This lab follows the documentation in the Wiki[18]. Let's see if we can test for technique T1047 "Windows Management Instrumentation" which belongs to tactic TA0002 Execution.

```
Uninstall-WindowsFeature Windows-Defender
Restart-Computer -Force

IEX (IWR 'https://raw.githubusercontent.com/redcanaryco/
          invoke-atomicredteam/master/install-atomicredteam.ps1'
          -UseBasicParsing);
Install-AtomicRedTeam -getAtomics

# Import the module (maybe this is not needed)
Import-Module `
  "C:\AtomicRedTeam\invoke-atomicredteam\Invoke-AtomicRedTeam.psd1" `
  -Force

# List all available tests with the Att&ck ID number
Invoke-AtomicTest All -ShowDetailsBrief

# Show info for all tests in technique T1047
# READ THE OUTPUT FROM THIS COMMAND.
Invoke-AtomicTest T1047 -ShowDetailsBrief

# Check prereqs for T1047
Invoke-AtomicTest T1047 -CheckPrereqs

# Eight tests seems ok, last two needs prereqs
Invoke-AtomicTest T1047 -GetPrereqs

# Run tests for local and remote execution
Invoke-AtomicTest T1047 -TestNumbers 5,6

# Run all ten tests for WMI
Invoke-AtomicTest T1047

# MAKE SURE YOU READ AND STUDY THE OUTPUT GENERATED

# Cleanup
Invoke-AtomicTest T1047 -Cleanup
```

[18] github.com/redcanaryco/invoke-atomicredteam/wiki

2. **BloodHound.** Let's do some reconnaissance. This lab assumes you start with your infrastructure with a fresh domain install, meaning the way it looks after chapter six (AD installed and all hosts joined to the domain, but no OUs or users/groups created yet).

   (a) Login as local Admin on SRV1 and start an elevated PowerShell ("PowerShell as administrator")

   ```
   # Install neo4j (includes java), git and wget
   choco install -y neo4j-community git wget 7zip

   # Close and reopen PowerShell as administrator,
   # download BloodHound-repo
   cd $home
   git clone https://github.com/BloodHoundAD/BloodHound.git
   ```

   Visit localhost:7474 in a browser, login: username/password is `neo4j/neo4j`, you will be asked to change password, remember to write it down.

   (b) Let's start BloodHound and see that we can log in with the same username and password.

   ```
   cd $home
   wget  https://github.com/BloodHoundAD/BloodHound/releases/
         download/4.1.0/BloodHound-win32-x64.zip
   7z x .\BloodHound-win32-x64.zip
   .\BloodHound-win32-x64\BloodHound.exe
   ```

   (c) To gather data for BloohHound we need some tools that Windows will think is malware, but we know what we are doing (I hope) so let's remove Windows Defender and gather data

   ```
   Uninstall-WindowsFeature Windows-Defender
   Restart-Computer -Force
   # login as Domain Administrator
   cd \Users\Admin\BloodHound\Collectors
   .\SharpHound.exe
   ```

   This should have been possible to do without being a domain user with something like (the author have not tried to troubleshoot why this does not work)

   ```
   #.\SharpHound.exe
   # --ldapusername "SEC\mysil"
   # --ldappassword "Pa$$w0rdMy"
   # --disablekerberossigning
   # --Domain sec.core
   # --Domaincontroller dc1.sec.core
   ```

(d) What does our domain look like? Log out and log back in as local Admin, then do

```
cd $home
.\BloodHound-win32-x64\BloodHound.exe
# Log in
# "Upload data" (4th button from the top in right side)
# choose the zip file created by SharpHound
```

Run the query ”Find Shortest Paths to Domain Admins” and you should see something like the screenshot from BloodHound in the chapter text.

(e) Log in as domain administrator and run the following commands (change values if your users are different from these):

```
Add-ADGroupMember -Identity 'Enterprise admins' -Members felix
```

Run SharpHound again and redo the previous item with uploading data and ”Find Shortest Paths to Domain Admins”. Notice how BloodHound shows you that `felix` is now a path to domain administrator.

10.6 Review questions and problems

1. Describe the relationship between the *Cyber Kill Chain* and *Mitre Att&ck*.
2. What does the acronym *TTP* stand for in Mitre Att&ck, and what is the difference between a tactic, a technique, and a procedure?
3. What is meant by an *adversary* in the context of Mitre Att&ck?
4. Give two strengths and two limitations of Mitre Att&ck as described in the chapter.
5. Name and describe at least five of the Mitre Att&ck Tactics.
6. Visit the Attack Navigator[19], create a "New empty layer", choose "Enterprise". Use the colored version of the Mitre Att&ck Matrix in the chapter, choose at least one blue and one green technique, right click on them and find examples of exact procedures of how they have been used.

[19] mitre-attack.github.io/attack-navigator

11 Security: Defenses

11.1 NSM Grunnprinsipper

NSM Grunnprinsipper for IKT-sikkerhet 2.1 in figure 11.1. Defending an ICT system involves designing, implementing, and maintaining a secure configuration ("hardening" the system, as well as configuring logging and additional security software). In this sense, we return to Chapter 2.3, "Ivareta en sikker konfigurasjon", in NSM's "Grunnprinsipper for IKT-sikkerhet 2.1" [1].

11.2 Mitre D3fend

Mitre D3fend [25] is an ongoing research effort to develop a countermeasure knowledge base that describes tactics and techniques used in practice to defend against attacks. D3fend can be used to establish links to the TTPs (Tactics, Techniques, and Procedures) defined in Mitre Att&ck.

11.2.1 All the Tactics

From "D3FEND: A knowledge graph of cybersecurity countermeasures[1] we can read that the tactics are described as:

[1] d3fend.mitre.org

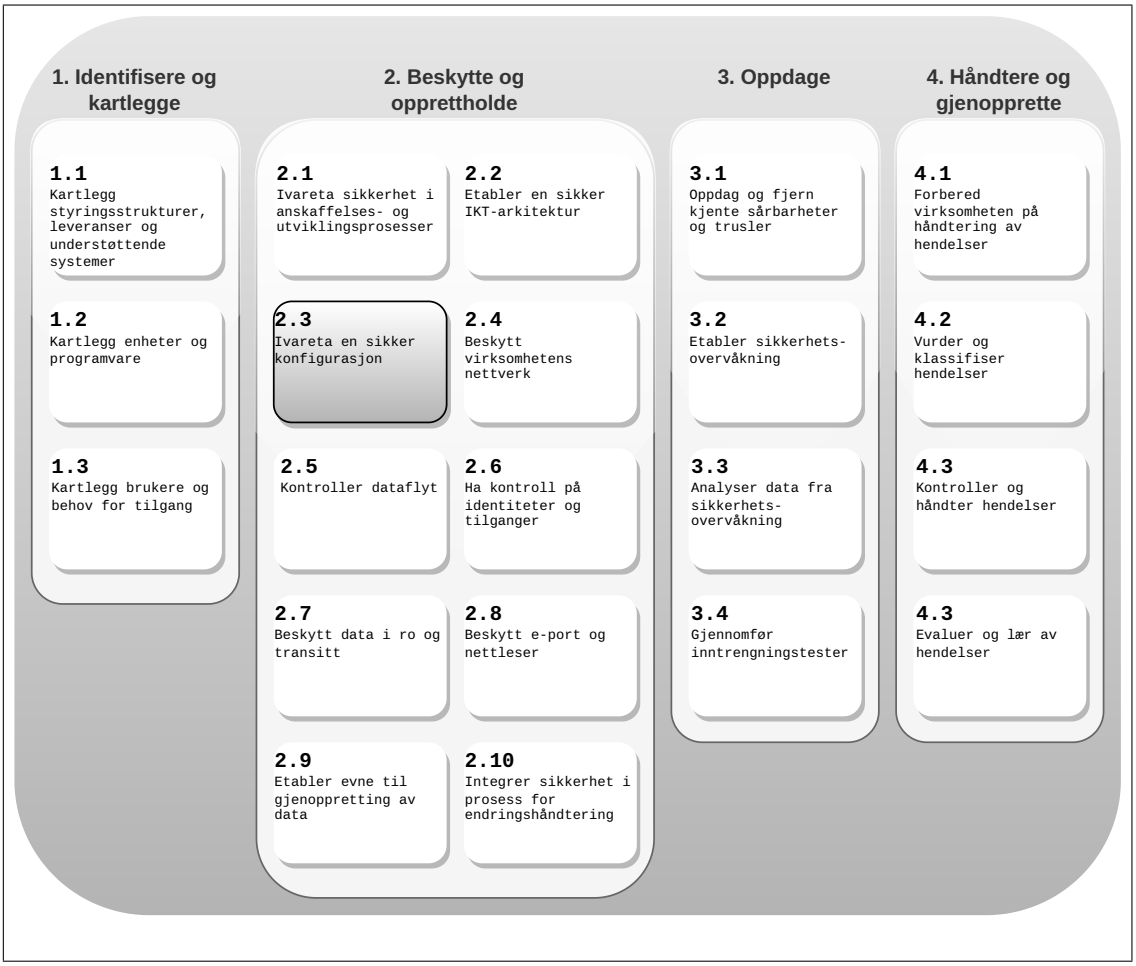


Figure 11.1: NSM Grunnprinsipper for IKT-sikkerhet 2.1.

1. Model: *The model tactic is used to apply security engineering, vulnerability, threat, and risk analyses to digital systems. This is accomplished by creating and maintaining a common understanding of the systems being defended, the operations on those systems, actors using the systems, and the relationships and interactions between these elements.*

2. Harden: *The harden tactic is used to increase the opportunity cost of computer network exploitation. Hardening differs from Detection in that it generally is conducted before a system is online and operational.*

3. Detect: *The detect tactic is used to identify adversary access to or unauthorized activity on computer networks.*

4. Isolate: *The isolate tactic creates logical or physical barriers in a system which reduces opportunities for adversaries to create further accesses.*

5. Deceive: *The deceive tactic is used to advertise, entice, and allow potential attackers ac-*

cess to an observed or controlled environment.

6. Evict: *The eviction tactic is used to remove an adversary from a computer network.*
7. Restore: *The restore tactic is used to return the system to a better state.* ("build back better" which we will also read about on the next chapter)

Maybe you can remember MH-DIDER as an acronym? (Model, Harden, Detect, Isolate, Deceive, Evict, Restore).

11.3 Hardening

Achieving secure Windows clients and servers is not a matter of magic. It means recognizing that we can never be 100% protected, but that we must do our best to reduce risk and be prepared to respond when incidents occur. As already stated, the fundamentals are:

- Backup and restore
- Identity management and access control
- Configuration management
- Software package management
- Logging and monitoring
- Fast (automatic) redeployment/installation/provisioning

The process of configuring a server or client to be as secure as possible is called *hardening* ("herding" in Norwegian). In relation to our list of fundamentals, hardening is primarily concerned with configuration management, although it also relates closely to access control, software package management, and logging. A good starting point is to work with the Microsoft Security Baselines, which we covered in Chapter 7. In addition, it is important to consult official documentation and recommendations, such as Best Practices for Securing Active Directory[2].

11.3.1 Defender

Defender in figure 11.2. As part of the hardening process, we should configure anti-malware software such as Microsoft Defender. Defender can detect malware and may

[2] docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/best-practices-for-securing-active-directory

```
PS C:\Users\Administrator> Get-Command -Module ConfigDefender

CommandType     Name
-----------     ----
Function        Add-MpPreference
Function        Get-MpComputerStatus
Function        Get-MpPreference
Function        Get-MpThreat
Function        Get-MpThreatCatalog
Function        Get-MpThreatDetection
Function        Remove-MpPreference
Function        Remove-MpThreat
Function        Set-MpPreference
Function        Start-MpRollback
Function        Start-MpScan
Function        Start-MpWDOScan
Function        Update-MpSignature
```

Figure 11.2: Defender.

prevent us from installing software that it suspects could be used for malicious purposes (as we will see in the lab exercises). We can also interact with Defender through PowerShell:

```
$Preferences = Get-MpPreference
Preferences.ScanScheduleDay
```

If this says `0` it means "every day"[3], meaning the host will scan for malware every day (as opposed to only one of the seven days of the week).

[3] docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/dn455323(v=vs.85)

11.4 Lab tutorials

1. No lab tutorials this week.

11.5 Review questions and problems

1. Name and describe all the seven of the tactics in Mitre D3fend.

12 Infrastructure Orchestration

12.1 NSM Grunnprinsipper

NSM Grunnprinsipper for IKT-sikkerhet 2.1 in figure 12.1. The last chapter in NSM's "Grunnprinsipper for IKT-sikkerhet 2.1" [1] focuses on how to "Håndtere og gjenopprette", in other words, how to manage an incident after it has occurred and how to restore affected services. At this stage, the organization has already experienced a security breach or operational failure, and the primary objective is to return to a stable and trustworthy operational state as quickly and safely as possible.

In such situations, it is often necessary to decommission compromised systems entirely and recreate them from scratch in a clean and known-good state. Servers that have been hacked can no longer be trusted, even if the visible indicators of compromise have been removed. As a result, deleting affected servers and rebuilding them is frequently safer than attempting to repair them in place. This process of systematic teardown and reconstruction is the core topic addressed in this chapter.

However, restoration should not be limited to simply recreating servers in their previous form. Instead, we should aim to "build back better." As Morris [54] argues, infrastructure should be *antifragile*, meaning that each incident strengthens the system rather than weakening it. In an antifragile infrastructure, failures and attacks become sources of learning, leading to improved configurations, stronger controls, and more resilient architectural choices.

This implies that incidents must be treated as learning opportunities. Before rebuilding servers, we should analyze what went wrong, identify weaknesses in configuration or process, and deliberately improve the security posture of the new environment. The

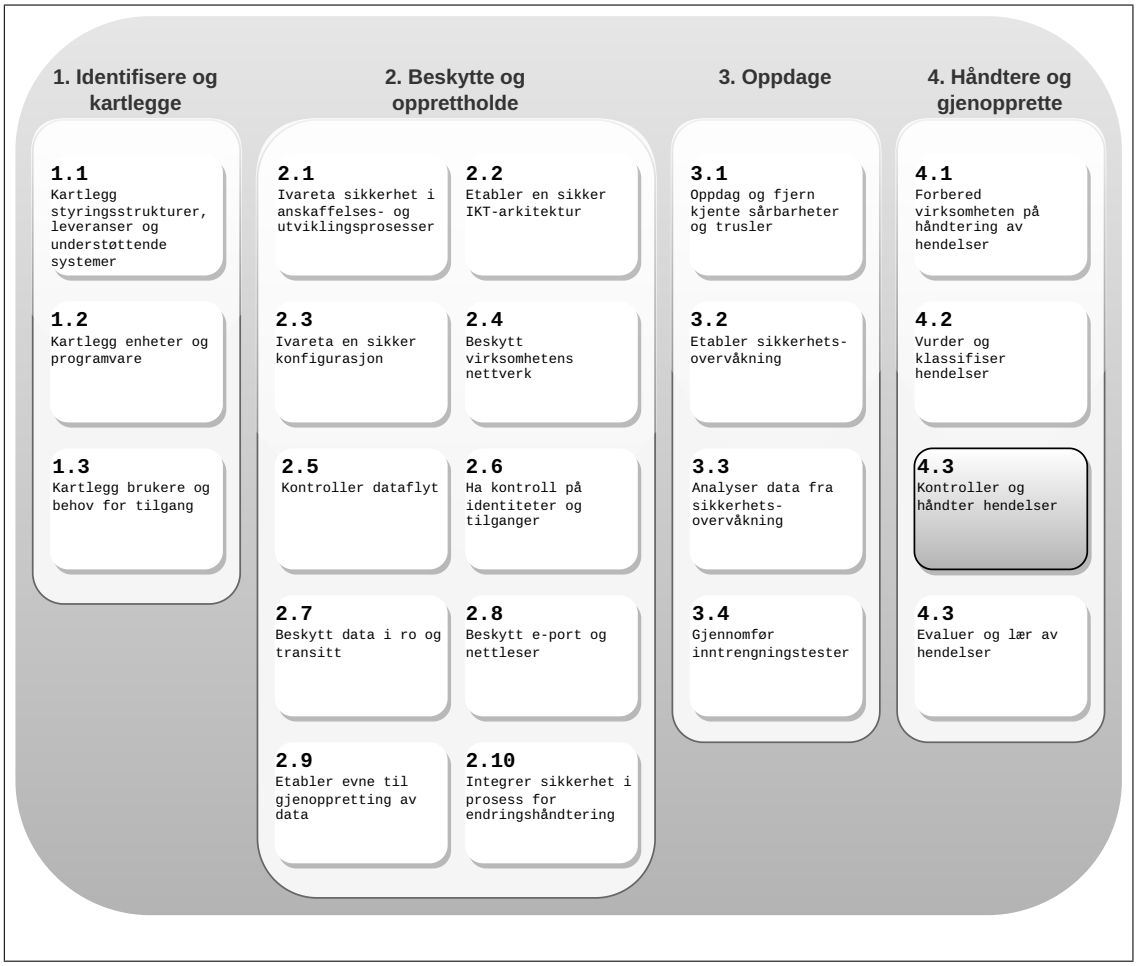


Figure 12.1: NSM Grunnprinsipper for IKT-sikkerhet 2.1.

rebuilt infrastructure should therefore be *more secure and more robust* than the one that was compromised.

The importance of structured incident handling and recovery is also directly addressed in NSM's chapter 4.3, "Kontroller og håndter hendelser." In particular, section 4.3.4 emphasizes the need for controlled recovery processes, which reinforces the idea that restoration is not merely a technical task, but a disciplined and security-driven activity:

> Iverksett gjenopprettingsplan i løpet av, eller i etterkant av hendelsen. Tiltakene vil variere avhengig av type hendelse, men kan inkludere: a) Reaktivere redundante ressurser som ble tapt eller skadet under hendelsesforløpet. b) Reinstallere maskin- og programvare på rammede komponenter. c) Gjenopprette konfigurasjonsinnstillinger, med eventuelle tilpasninger. d) Gjenopprette tjenester som ble stoppet under hendelsesforløpet. IKT-systemene bør bygges opp til en bedre forfatning enn de var i før hendelsen inntraff

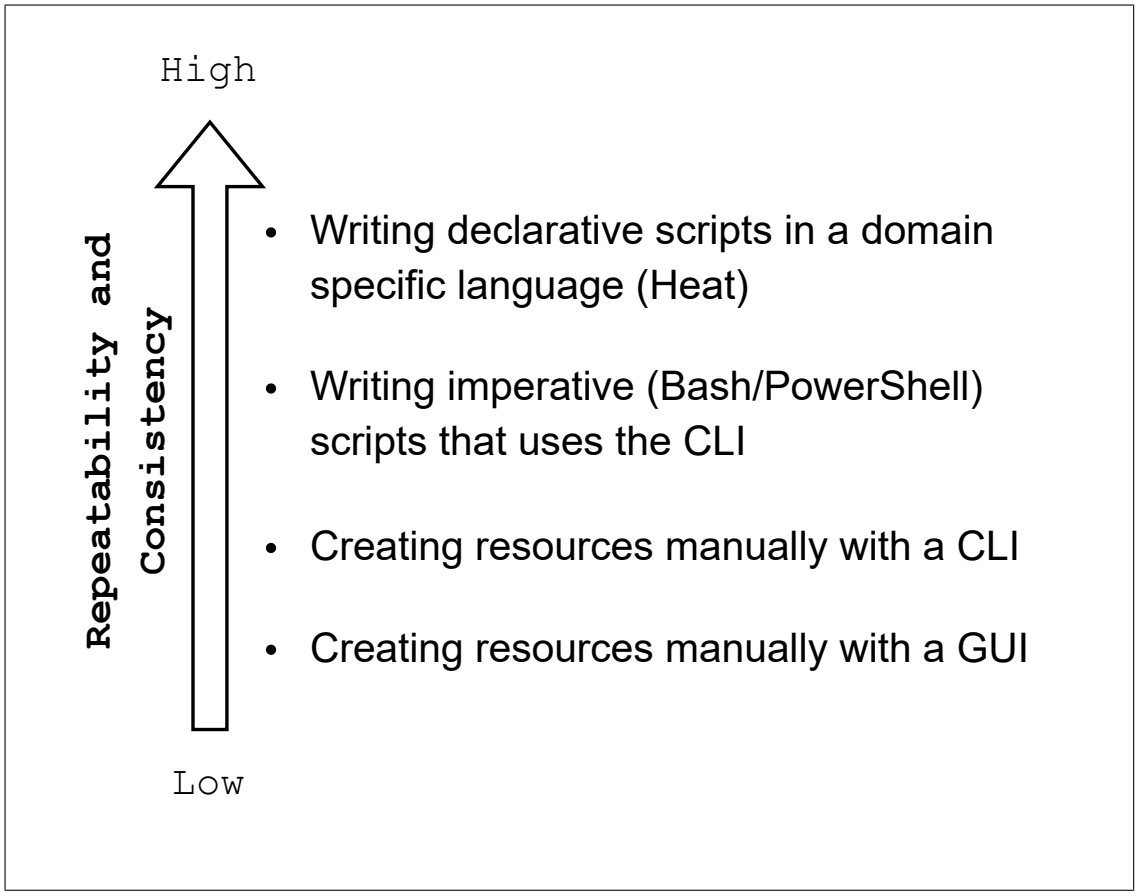


Figure 12.2: Evolution.

("build back better").

12.2 Evolution

Evolution in figure 12.2. The simplest way to use a public or private cloud is to log in to a GUI (such as OpenStack Horizon, which we have used throughout this course) and create resources (key pairs, networks, routers, servers, etc.) manually by pointing and clicking, while occasionally providing additional input values. Although such manual tasks are straightforward, they are time-consuming and prone to error. Always remember: *humans make errors; computers do not (unless we program them to)*. For this reason, we should move to a command-line interface (CLI), since command-line operations can be documented more easily, and a command-line interface also allows for scripting.

Since we all love scripting in Bash and PowerShell, the next natural step would be to place all the OpenStack commands into a script, together with additional variables, loops, and conditions. After doing this for a while, however, you will see that this ap-

proach also creates its own set of problems: what happens if a command fails? You need to handle every situation in which something can go wrong. You also need separate functionality for creating and deleting resources. If you try to write such a script, you will quickly realize that there must be a better way. And there is. While regular programming and scripting are what we call *imperative programming*, we should instead turn to what we call *declarative programming*. Imperative programming focuses on how to do something, while declarative programming focuses on the end state that we want to achieve.

When professionals use public or private clouds to create infrastructures, they use a declarative domain-specific language and write ”code” that describes the infrastructure they want to create. We call it a *domain-specific language* because we are referring to a ”programming language” that is designed for a specific purpose, as opposed to a general-purpose programming language such as C or C++.

Infrastructure orchestration is a subtopic of *Infrastructure as Code (IaC)*. In Infrastructure as Code, we should always keep the following two principles in mind [54]:

Repeatability Recreating (deleting and creating) the same infrastructure, or creating variations of the same basic infrastructure (”reusability”), should be fast and easy.

Consistency Whenever we create an infrastructure, the same things should happen: the end result (the ”end state”) should always be the same, and there should be no surprises.

Since this is a course in ”Secure Core Services”, these principles are particularly important to us because we need the ability to recreate services quickly when incidents occur. If we are attacked, we may need to take down our servers and recreate them rapidly.

12.3 Orchestration Tools

Orchestration Tools in figure 12.3. Since we use OpenStack, we will use OpenStack’s declarative, domain-specific orchestration language/tool, which is called Heat. Each of the major public cloud providers—Google, Azure, and Amazon—has its own similar language/tool. Heat was originally developed as a ”clone” of Amazon CloudFormation. All of these languages/tools are quite similar and aim to accomplish the same kinds of tasks, so once you know one of them, you can usually switch to another without too much difficulty. Terraform is somewhat different from the others because it is designed to be cross-platform. This means that if you want to write infrastructure code for deployment on, for example, both Azure and Amazon, Terraform is probably a good choice. Terraform has its own language, called HCL, while most other tools use a language based on YAML or JSON[1]:

[1]There is also a format called TOML, but it is not widely used in the orchestration world.

- OpenStack: Heat (templates in yaml)
- Amazon: CloudFormation (templates in yaml/json)
- Azure: Azure Resource Manager, (ARM templates in json)
- Google Compute Engine: Cloud Deployment Manager (templates in yaml)
- Hashicorp: Terraform (HashiCorp Configuration Language (HCL))
- ...

YAML and *JSON*

Figure 12.3: Orchestration Tools.

YAML is short for "YAML Ain't Markup Language". It is a human-friendly (that is, "easy-to-read") language for storing data. The most important property to be aware of in YAML is that it is based on *indentation*. The main page yaml.org is itself valid YAML. If you want to look up the exact details of YAML, you can consult the specification[2], but it is probably more useful to browse a cheat sheet such as the YAML cheat sheet[3] (where you can also see how YAML maps to JSON). This is a YAML example:

```
dc1:
  type: OS::Nova::Server
  properties:
    name: dc1
    image: 'Windows Server 2025 Standard [Evaluation]'
```

JSON is short for "JavaScript Object Notation" and is similar to YAML, but it is more focused on data interchange than on readability. While YAML is based on indentation, JSON represents everything using nested blocks enclosed in braces ({ and }), similar to what you know from C and C++ programming. If you want to look up the exact details of JSON, you can consult the reference[4], but it is probably more useful to browse a cheat sheet, such as the JSON cheat sheet[5]. This is the example above written in JSON:

```
{
  "dc1": {
    "type": "OS::Nova::Server",
    "properties": {
```

[2] yaml.org/spec/1.2.2
[3] quickref.me/yaml
[4] www.ecma-international.org/publications-and-standards/standards/ecma-404
[5] quickref.me/json

```
heat_template_version: ...

description: >
  HOT template to create ...

parameter_groups:
  ...

parameters:
  ...

resources:
  ...

outputs:
  ...
```

Figure 12.4: Heat Template Syntax.

```
            "name": "dc1",
            "image": "Windows Server 2025 Standard [Evaluation]"
          }
        }
      }
```

In general, we can say that YAML is typically used when readability is important, while JSON is more commonly used when speed or performance is a priority. System administrators tend to prefer YAML, whereas programmers often prefer JSON.

12.4 OpenStack Heat Basics

Heat Template Syntax in figure 12.4. A Heat template [58] starts with a version number, which is a date that corresponds to the release date of an OpenStack version (which occurs every six months). The version number can optionally be followed by a description, which typically uses the folded multiline YAML construction that starts with the character > (”folded” means that any newline that follows is replaced with a space). The `parameter_groups` section is optional and not widely used, but it allows parameters to be grouped, which can be useful for applications that interface with Heat. The *parameters* section is where you define all the input parameters for your template, for example which server flavor your stack should use, or which SSH key pair should be used. An important feature of these parameters is that you can *set default values*, thereby creating flexible templates that are still easy to use. With default values, the template

- Resources dependencies
- Conditions
- Iteration
- Boot scripts
- Nested stacks

Figure 12.5: Advanced Topics.

can be used without explicitly providing values for all parameters, which makes it easy to get started with the template.

The most important section of the template is the *resources* part. This is what you can think of as "the actual code" of the template. In this part, you describe all the resources that make up your stack (remember: "stack" means the same as "infrastructure" in this context). A resource [59] is anything you can create in OpenStack, for example `OS::Nova::Server` or `OS::Neutron::Router`.

Finally, the *outputs* section is used to define information that should be available after a stack has been created, for example the IP addresses of servers, URLs for web applications, or other values that have been created as part of the stack.

Feel free to study the template we used in the beginning of the course[6].

12.5 OpenStack Heat Advanced

Advanced Topics in figure 12.5. A Heat template is what we call a *configuration definition file* [54]. When we start working with such templates, we soon discover that we need more functionality. It is important to realize that there is a clear difference between domain-specific languages such as Heat and general-purpose programming languages. A Heat template is a configuration definition file (we may also refer to it as an infrastructure definition file); it is not program source code like a C source file. However, *it is acceptable to refer to configuration definition files as "code", because we should follow the same process that programmers use when writing or modifying them: using tools to catch errors and check quality, and storing our "code" in Git repositories*. Heat is not a general-purpose programming language, but it does have some features that make it resemble a programming language, as well as some useful capabilities that solve typical problems we encounter:

Resource dependencies Study the use of `depends_on` in the file `iac_top.yaml` at github.com/githubgossin/IaC-heat/blob/master/iac_top.yaml

[6] git.ntnu.no/erikhje/sky/blob/main/single_windows_server.yaml

Conditions Heat supports defining conditions[7] based on the input parameters provided to the stack. These conditions can then be used to dynamically create resources or make other changes to the stack.

Iteration Study the use of `OS::Heat::ResourceGroup`, `count`, and `server_name` in the file `iac_rest.yaml` at github.com/githubgossin/IaC-heat/blob/master/iac_rest.yaml. Heat also supports the function `repeat`[8] to enable iteration over lists.

Boot scripts Study the `user_data` section in the file `cl_dc_srv_basic.yaml` at git.ntnu.no/erikhje/sky/blob/main/cl_dc_srv_basic.yaml, and also how the boot script can be placed in an external file, as in `managed_windows_server.yaml` at github.com/githubgossin/IaC-heat/blob/master/lib/managed_windows_server.yaml

Nested stacks Study the file structure in the Git repository `IaC-heat` at github.com/githubgossin/IaC-heat

[7] docs.openstack.org/heat/latest/template_guide/hot_spec.html (see the conditions section)
[8] docs.openstack.org/heat/latest/template_guide/hot_spec.html (see the repeat section)

12.6 Lab tutorials

1. **OpenStack CLI**. Install the OpenStack command line client (do this on your laptop or any other host you prefer to use, it is preinstalled on login.stud.ntnu.no if you prefer to use that server):

   - Instructions for Windows at
     www.ntnu.no/wiki/display/skyhigh/Openstack+CLI+on+Windows
   - Instructions for Linux using login.stud.ntnu.no or other Linux host at
     www.ntnu.no/wiki/display/skyhigh/Using+the+commandline+clients
     (if installing on other Linux host do
     `sudo apt install python3-openstackclient python3-heatclient`). You only have to read and do the first part of this page, stop when you see the headline "Creating an initial network topology"

   Also read about authentication alternatives[9] (remember that we used application credentials[10] when we set up the backup system with `restic`). Note that if you use Application credentials created from Horizon (like we did with backups) you have to check the box "Unrestricted (dangerous)". When you have completed your setup test that it works with the command:
   `openstack flavor list`

2. **Create and delete a stack using CLI**. Download the Heat template Single Windows Client from
   git.ntnu.no/erikhje/sky/blob/main/single_windows_client.yaml,
   create the stack from the template, view that it has been created in Horizon (the OpenStack web-interface), finally delete the stack.

   ```
   openstack stack create -t single_windows_client.yaml \
     --parameter key_name=dcsg1005 mysilstack
   # view in GUI and you can also verify with
   openstack stack list
   # finally delete the stack
   openstack stack delete mysilstack
   ```

3. **Template from developers and environment file**. Create a new network with two Ubuntu instances by using the Heat template `servers_in_new_neutron_net.yaml` from github.com/openstack/heat-templates/blob/master/hot
   Remember from chapter 8.6 to always examine code that you reuse. This template is from the repo of the OpenStack Heat developers, so this is probably good code, and reusing good code is a best practice we should enforce (as long as we trust the source the code is coming from).

[9] www.ntnu.no/wiki/display/skyhigh/Authentication+alternatives
[10] www.ntnu.no/wiki/display/skyhigh/Application+credentials

```
openstack stack create -t servers_in_new_neutron_net.yaml \
 -e heat_demo_env.yaml heat_demo
```

Create an environment file `heat_demo_env.yaml` that looks like this (the environment file is for enforcing a good computer science principle: *separating code and data*):

```
parameters:
  key_name: KEY_NAME
  image: Ubuntu Server 22.04 LTS (Jammy Jellyfish) amd64
  flavor: gx1.1c1r
  public_net: ntnu-internal
  private_net_name: net1
  private_net_cidr: 192.168.125.0/24
  private_net_gateway: 192.168.125.1
  private_net_pool_start: 192.168.125.200
  private_net_pool_end: 192.168.125.250
```

If you get an error immediately when trying to create a stack you probably have a syntax error (e.g. your YAML file does not have correct indentation or similar). If syntax is OK, but the stack fails to create e.g. `CREATE_FAILED` message or similar, try something like
`openstack stack event list heat_demo --nested-depth 3`

12.7 Review questions and problems

1. Why is it often safer to rebuild a compromised server from scratch instead of trying to repair it in place?
2. What is meant by the idea that infrastructure should be "antifragile" after an incident?
3. Why are manual cloud operations performed through a GUI usually not sufficient for professional infrastructure management?
4. What is a domain-specific language, and why is Heat considered one?
5. What do the principles of repeatability and consistency mean in Infrastructure as Code?
6. Why are repeatability and consistency especially important in a course about secure core services?
7. What is the most important structural difference between YAML and JSON?
8. Briefly explain the purpose of the main sections in a Heat template: `parameters`, `resources`, and `outputs`.
9. Why can a Heat template reasonably be referred to as "code" even though it is not a general-purpose program?
10. What is the purpose of advanced Heat features such as resource dependencies, conditions, iteration, boot scripts, and nested stacks?
11. Study the template we used in the beginning of the course at git.ntnu.no/erikhje/sky/blob/main/single_windows_server.yaml.
    (a) How does a Heat resource retrieve/make use of a value passed as a parameter to the template?
    (b) How does a Heat resource reference another Heat resource?
12. Download the template we used in the beginning of the course from git.ntnu.no/erikhje/sky/blob/main/single_windows_server.yaml.
    Study how a boot script is used to change hostname in `cl_dc_srv_basic.yaml` at git.ntnu.no/erikhje/sky/blob/main/cl_dc_srv_basic.yaml,
    and modify the template you have downloaded in such a way that it will install PowerShell Core and SysInternals with a boot script.
13. After you have completed this chapters lab tutorial, download the template `servers_in_new_neutron_net.yaml` from github.com/openstack/heat-templates/blob/master/hot.

Modify the template in such a way that the two servers (the ”instances”) in the template can be different (e.g. an Ubuntu and a Windows instance). You can do this by changing the parameter list and the corresponding references to the parameters in the resources. Also add a security group to both servers, use the security group you find in the template we used in the beginning of the course at git.ntnu.no/erikhje/sky/blob/main/single_windows_server.yaml.
Verify that the template works after you have modified it by creating a stack. Tip: remember from the lab tutorial that you can get information about what is failing in your stack with

```
openstack stack event list STACK_NAME --nested-depth 3
```

Bibliography

[1] Nasjonal Sikkerhetsmyndighet (NSM). *Grunnprinsipper for IKT-sikkerhet 2.1*. URL: https://nsm.no/regelverk-og-hjelp/rad-og-anbefalinger/grunnprinsipper-for-ikt-sikkerhet (visited on 10/28/2024).

[2] Cloud Native Computing Foundation (CNCF) Technical Oversight Committee (TOC). *CNCF Cloud Native Definition v1.0*. URL: https://github.com/cncf/toc/blob/main/DEFINITION.md (visited on 06/11/2018).

[3] Abutalib Aghayev et al. "File Systems Unfit as Distributed Storage Back Ends: Lessons from 10 Years of Ceph Evolution". In: *login Usenix Mag.* 45.1 (2020). URL: https://www.usenix.org/publications/login/mar20/aghayev.

[4] ajgo@chromium.org. *Issue 1254631*. URL: https://bugs.chromium.org/p/chromium/issues/detail?id=1254631 (visited on 12/13/2022).

[5] Baan Alsinawi. *Key Takeaways from the NotPetya Malware Infection*. URL: https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2018/key-takeaways-from-the-notpetya-malware-infection (visited on 09/26/2018).

[6] Pieter Arntz. *Why you shouldn't automate your VirusTotal uploads*. URL: https://www.malwarebytes.com/blog/news/2022/04/why-you-shouldnt-automate-your-virustotal-uploads (visited on 04/18/2022).

[7] restic authors. *Restic Documentation*. URL: https://restic.readthedocs.io/en/latest (visited on 01/03/2022).

[8] Nasreddine Bencherchali. *Windows System Processes - An Overview For Blue Teams*. URL: https://nasbench.medium.com/windows-system-processes-an-overview-for-blue-teams-42fa7a617920 (visited on 10/24/2020).

[9] Timm Böttger et al. "An Empirical Study of the Cost of DNS-over-HTTPs". In: *ACM Internet Measurement Conference (to appear)*. 2019. URL: http://eecs.qmul.ac.uk/~boettget/assets/doh-imc19.pdf (visited on 08/01/2019).

[10] Scott Chacon and Ben Straub. *Pro Git*. 2nd. USA: Apress, 2014. ISBN: 1484200772. URL: https://git-scm.com/book/en/v2.

[11] Chocolatey. *AUTOMATE PACKAGE INTERNALIZER*. URL: https://docs.chocolatey.org/en-us/guides/organizations/automate-package-internalization (visited on 02/04/2022).

[12] Inc. Chocolatey Software. *Chocolatey Community Package Repository*. URL: https://docs.chocolatey.org/en-us/information/security#chocolatey-community-package-repository (visited on 12/13/2022).

[13] A. Costello. *Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)*. RFC 3492. RFC Editor, Mar. 2003, pp. 1–34. URL: https://www.rfc-editor.org/rfc/rfc3492.txt.

[14] Jim Curry. *Introducing OpenStack*. URL: https://web.archive.org/web/20171026111206/https://www.openstack.org/blog/2010/07/introducing-openstack/ (visited on 07/19/2010).

[15] National Vulnerability Database. *CVE-2021-44228 Detail*. URL: https://nvd.nist.gov/vuln/detail/CVE-2021-44228 (visited on 01/12/2022).

[16] Wayne Davison. *rsync*. URL: https://rsync.samba.org (visited on 12/15/2022).

[17] Patrick Debois. *In depth research and trends analyzed from 50+ different concepts as code*. URL: https://www.jedi.be/blog/2022/02/23/trends-and-inventory-of-50-as-code-concepts (visited on 12/12/2022).

[18] Sara Dickinson. *DNS Privacy Project*. URL: https://dnsprivacy.org (visited on 08/01/2019).

[19] Ruian Duan et al. "Towards Measuring Supply Chain Attacks on Package Managers for Interpreted Languages". In: *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021. URL: https://www.ndss-symposium.org/ndss-paper/towards-measuring-supply-chain-attacks-on-package-managers-for-interpreted-languages/.

[20] Veeam R&D Forums. *Why isn't "Backup Operator" role used ("day 2" question)*. URL: https://forums.veeam.com/veeam-agent-for-windows-f33/why-isn-t-backup-operator-role-used-day-2-question-t75916.html (visited on 08/10/2021).

[21] Æ. Frisch. *Essential System Administration: Tools and Techniques for Linux and Unix Administration*. O'Reilly Media, 2002. ISBN: 9780596550493. URL: https://books.google.no/books?id=uRW8V9QOL7YC.

[22] CNCF Security Technical Advisory Group. *Software Supply Chain Best Practices*. URL: https://github.com/cncf/tag-security/tree/main/supply-chain-security/supply-chain-security-paper (visited on 02/17/2023).

[23] Martin Gundersen and Ståle Grut. *Betalte hackerne: – Vi var presset opp i et hjørne*. URL: https://nrkbeta.no/2022/01/17/betalte-hackerne-vi-var-presset-opp-i-et-hjorne/ (visited on 01/17/2022).

[24] Santosh Janardhan. *More details about the October 4 outage*. URL: https://engineering.fb.com/2021/10/05/networking-traffic/outage-details (visited on 10/05/2021).

[25] Peter E. Kaloroumakis and Michael J. Smith. *Toward a Knowledge Graph of Cybersecurity Countermeasures*. Tech. rep. The MITRE Corporation, 2021.

[26] Sean Michael Kerner. *OpenStack Now Powers 75 Public Clouds Worldwide*. URL: https://www.eweek.com/cloud/openstack-now-powers-75-public-clouds-worldwide/ (visited on 11/13/2018).

[27] Gene Kim, Kevin Behr, and George Spafford. *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win*. 1st. IT Revolution Press, 2013. ISBN: 0988262592.

[28] P. Krogh. *The DAM Book: Digital Asset Management for Photographers*. O'Reilly Media, 2009. ISBN: 9781449343712. URL: https://books.google.no/books?id=RX4KAQAAQBAJ.

[29] Tom Limoncelli, Christina J Hogan, and Strata R Chalup. *The Practice of System and Network Administration*. 2nd. Pearson Education, 2007.

[30] Subhro Majumder. *Active Directory: Design Considerations and Best Practices*. URL: https://social.technet.microsoft.com/wiki/contents/articles/52587.active-directory-design-considerations-and-best-practices.aspx (visited on 07/18/2021).

[31] Microsoft. *About Mutual Authentication Using Kerberos*. URL: https://learn.microsoft.com/en-us/windows/win32/ad/about-mutual-authentication-using-kerberos (visited on 08/23/2019).

[32] Microsoft. *Active Directory FSMO roles in Windows*. URL: https://docs.microsoft.com/en-us/troubleshoot/windows-server/identity/fsmo-roles (visited on 01/12/2021).

[33] Microsoft. *Active Directory Security Groups*. URL: https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups (visited on 12/03/2021).

[34] Microsoft. *All Classes*. URL: https://docs.microsoft.com/en-us/windows/win32/adschema/classes-all (visited on 08/23/2019).

[35] Microsoft. *Attributes (AD DS)*. URL: https://docs.microsoft.com/en-us/windows/win32/ad/attributes (visited on 08/17/2020).

[36] Microsoft. *Changes to Service Host grouping in Windows 10*. URL: https://docs.microsoft.com/en-us/windows/application-management/svchost-service-refactoring (visited on 08/27/2021).

[37] Microsoft. *Create a virtual machine with a static private IP address using the Azure portal*. URL: https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/virtual-networks-static-private-ip-arm-pportal (visited on 10/01/2021).

[38] Microsoft. *Deploy Windows Server Update Services*. URL: https://learn.microsoft.com/en-us/windows-server/administration/windows-server-update-services/deploy/deploy-windows-server-update-services (visited on 07/29/2021).

[39] Microsoft. *Frequently asked questions about the GUID Partitioning Table disk architecture*. URL: https://learn.microsoft.com/en-us/troubleshoot/windows-server/backup-and-storage/guid-partitioning-table-disk-faq (visited on 12/15/2022).

[40] Microsoft. *Group Policy Preferences*. URL: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn581922(v=ws.11) (visited on 08/31/2016).

[41] Microsoft. *Local Accounts*. URL: https://learn.microsoft.com/en-us/windows/security/identity-protection/access-control/local-accounts (visited on 07/12/2022).

[42] Microsoft. *Password policy recommendations for Microsoft 365 passwords*. URL: https://learn.microsoft.com/en-us/microsoft-365/admin/misc/password-policy-recommendations (visited on 02/17/2023).

[43] Microsoft. *Ports used in Configuration Manager*. URL: https://docs.microsoft.com/en-us/mem/configmgr/core/plan-design/hierarchy/ports (visited on 01/10/2022).

[44] Microsoft. *Roles, Role Services, and Features*. URL: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc754923(v=ws.11) (visited on 11/17/2009).

[45] Microsoft. *Security Considerations for Writers*. URL: https://learn.microsoft.com/en-us/windows/win32/vss/security-considerations-for-writers (visited on 07/01/2022).

[46] Microsoft. *Service Principal Names*. URL: https://learn.microsoft.com/en-us/windows/win32/ad/service-principal-names (visited on 10/12/2021).

[47] Microsoft. *Sysinternals*. URL: https://docs.microsoft.com/en-us/sysinternals (visited on 12/16/2021).

[48] Microsoft. *Sysmon v14.14*. URL: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon (visited on 01/25/2023).

[49] Microsoft. *UNC paths*. URL: https://docs.microsoft.com/en-us/dotnet/standard/io/file-path-formats#unc-paths (visited on 09/15/2021).

[50] Microsoft. *Volume Shadow Copy Service*. URL: https://learn.microsoft.com/en-us/windows-server/storage/file-server/volume-shadow-copy-service (visited on 07/12/2022).

[51] Microsoft. *Well-known SIDs*. URL: https://docs.microsoft.com/en-us/windows/win32/secauthz/well-known-sids (visited on 07/01/2021).

[52] Microsoft. *Windows Commands*. URL: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands (visited on 12/13/2022).

[53] MITRE. *Steal or Forge Kerberos Tickets*. URL: https://attack.mitre.org/techniques/T1558 (visited on 01/16/2022).

[54] Kief Morris. *Infrastructure as Code: Dynamic Systems for the Cloud Age*. 2nd. O'Reilly Media, 2020.

[55] J. Moskowitz. *Group Policy: Fundamentals, Security, and the Managed Desktop*. Online access with DDA: Askews. Wiley, 2015. ISBN: 9781119035589. URL: https://books.google.no/books?id=-6KLBgAAQBAJ.

[56] B.C. Neuman and T. Ts'o. "Kerberos: an authentication service for computer networks". In: *IEEE Communications Magazine* 32.9 (1994), pp. 33–38. DOI: 10.1109/35.312841.

[57] NTNU. *File Backup*. URL: https://i.ntnu.no/wiki/-/wiki/English/File+Backup (visited on 12/15/2022).

[58] OpenStack. *Heat Orchestration Template (HOT) specification*. URL: https://docs.openstack.org/heat/latest/template_guide/hot_spec.html (visited on 04/18/2022).

[59] OpenStack. *OpenStack Resource Types*. URL: https://docs.openstack.org/heat/latest/template_guide/openstack.html (visited on 04/18/2022).

[60] Specter Ops. *BloodHound: Six Degrees of Domain Admin*. URL: https://bloodhoundenterprise.io (visited on 03/17/2022).

[61] OWASP. *Password Storage Cheat Sheet*. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (visited on 12/31/2022).

[62] OpenStack project. *Firewalls and default ports*. URL: https://docs.openstack.org/install-guide/firewalls-default-ports.html (visited on 12/14/2021).

[63] RexEgg. *The Elements of Good Regex Style*. URL: https://www.rexegg.com/regex-style.html (visited on 03/06/2023).

[64] Justin Samuel and Justin Cappos. "Package Managers Still Vulnerable: How to Protect Your Systems". In: *login:* (Feb. 2009), pp. 7–15.

[65] Snyk. *https://snyk.io/what-is-snyk*. URL: https://snyk.io (visited on 02/23/2023).

[66] Murugiah Souppaya and Karen Scarfone. *Guide to Enterprise Patch Management Planning: Preventive Maintenance for Technology*. en. 2022-04-06 04:04:00 2022. DOI: https://doi.org/10.6028/NIST.SP.800-40r4. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934311.

[67] Blake E Strom et al. *MITRE ATT&CK: Design and Philosophy*. Tech. rep. The MITRE Corporation, 2020.

[68] Sunny. *10 Years of OpenStack – Tim Bell at CERN*. URL: https://www.openstack.org/blog/10-years-of-openstack-tim-bell-at-cern/ (visited on 07/16/2020).

[69] SwiftOnSecurity. *sysmon-config — A Sysmon configuration file for everybody to fork*. URL: https://github.com/SwiftOnSecurity/sysmon-config (visited on 10/17/2021).

[70] Steve Traugott and Joel Huddleston. "Bootstrapping an Infrastructure". In: *12th Systems Administration Conference (LISA 98)*. Boston, MA: USENIX Association, Dec. 1998. URL: https://www.usenix.org/conference/lisa-98/bootstrapping-infrastructure.

[71] Nikolai Tschacher. *Typosquatting programming language package managers*. URL: https://incolumitas.com/2016/06/08/typosquatting-package-managers (visited on 06/08/2016).

[72] Rory Ward and Betsy Beyer. "BeyondCorp: A New Approach to Enterprise Security". In: *;login:* Vol. 39, No. 6 (2014), pp. 6–11.

[73] Wikipedia contributors. *Kill chain — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Kill_chain&oldid=1069871686. [Online; accessed 11-March-2022]. 2022.

[74] Wikipedia contributors. *Tape drive — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Tape_drive&oldid=1125947646. [Online; accessed 12-January-2023]. 2022.

[75] Wikipedia contributors. *VirusTotal — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=VirusTotal&oldid=1127343525. [Online; accessed 22-February-2023]. 2022.

[76] Wikipedia contributors. *Windows Registry — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Windows_Registry&oldid=1060508454. [Online; accessed 22-December-2021]. 2021.

[77] Pavel Yosifovich et al. *Windows Internals, Part 1: System Architecture, Processes, Threads, Memory Management, and More (7th Edition)*. 7th. USA: Microsoft Press, 2017. ISBN: 0735684189.