HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

STUDYTHESIS

Readiness for Tailored Attacks and Lateral Movement Detection

Authors:

Claudio MATTES
claudio.mattes@hsr.ch

Lukas KELLENBERGER
lukas.kellenberger@hsr.ch

Supervisor:

Cyrill BRUNSCHWILER
Hochschule für Technik Rapperswil
cyrill.brunschwiler@hsr.ch

DEPARTEMENT COMPUTER SCIENCES
HSR UNIVERSITY OF APPLIED SCIENCES RAPPERSWIL
CH-8640 RAPPERSWIL, SWITZERLAND

December 3, 2018

Abstract

Management Summary

Initial Situation

The amount of cyber-attacks, where malicious code is used, which not only settles on the infected system, but also infects other systems in the network, has increased massively recently. The outcome is often the complete infiltration of the organization. In the analysis of such an event, information and time are key factors to success. Consequently, readiness for such an event is a decisive factor.

The Japan Computer Emergency Response Team Coordination Center has analysed the procedure and the used tools of such attacks. In their most recent publication on this topic, they give hints which events indicate a possible contamination. The aim of this student project is to use this published information and write a tool that helps to identify the readiness of a system.

Procedure

The project was initially limited to Windows machines running on the operating system Windows 10 or Windows Server 16. The project was divided into four phases, one week iteration, five weeks for the elaboration of the project, six weeks for construction and two weeks for the final phase, the transition. During the elaboration phase we did some research on the topic and we tested different tools which cloud be interesting for our project. At the end of the elaboration we had decided to realise the project using PowerShell. In the following six weeks we wrote the "System Readiness Inspector - SRI", a PowerShell script.

The SRI reads information about the system on which it is running and evaluates which attack categories can or cannot be detected with these settings. This information obtained is then visualized into a PDF document and output by the script.

Results

The SRI runs successfully and outputs important system settings about the readiness. Illustrated in a PDF, the analyst can see at a glance which of his audit settings are missing or incorrect. The script also evaluates which attacks might be missed due to incorrectly configured settings. SRI helps an analyst to check a system for its readiness and saves him the tedious task of collecting and evaluating the data.

Outlook

SRI is still at an early stage of its development. The further development of the visualization is conceivable. The extension to an entire fleet will also be an approach that will certainly be pursued

further. Although SRI is a useful helper when it comes to get an quick overview about the audit settings and the readiness in general.

Contents

Part I

Technical Report

1 Introduction and Overview

As described in the introduction of the task definition, the key for a successful analysis in case of an advanced persistence threat (APT) or lateral movement in a network, it is fundamental to have solid event logging of all systems participating in the network.

2 Test environment

This chapter of the report describes the setup of the testing environment in which not only the tools during the research were tested, but also was used to test the System Readiness Inspector itself.

A virtual network was set up on the Microsoft Azure Cloud as a test environment. The test network was set up in the cloud so that the development team can access the network regardless of its location. The test network consists of a Windows server and two Windows clients. Active Directory service was configured on the server to manage the client computer and to have the possibilities to create group policies. Group policies are used in almost every corporate environment to build rule sets for configurations. These configurations are a core element to check the readiness of a system. The following operating systems were installed in this test network:

Server:

- Windows Server 2016

Clients:

- Windows 10 Pro, Version 1709

The network is structured as followed:

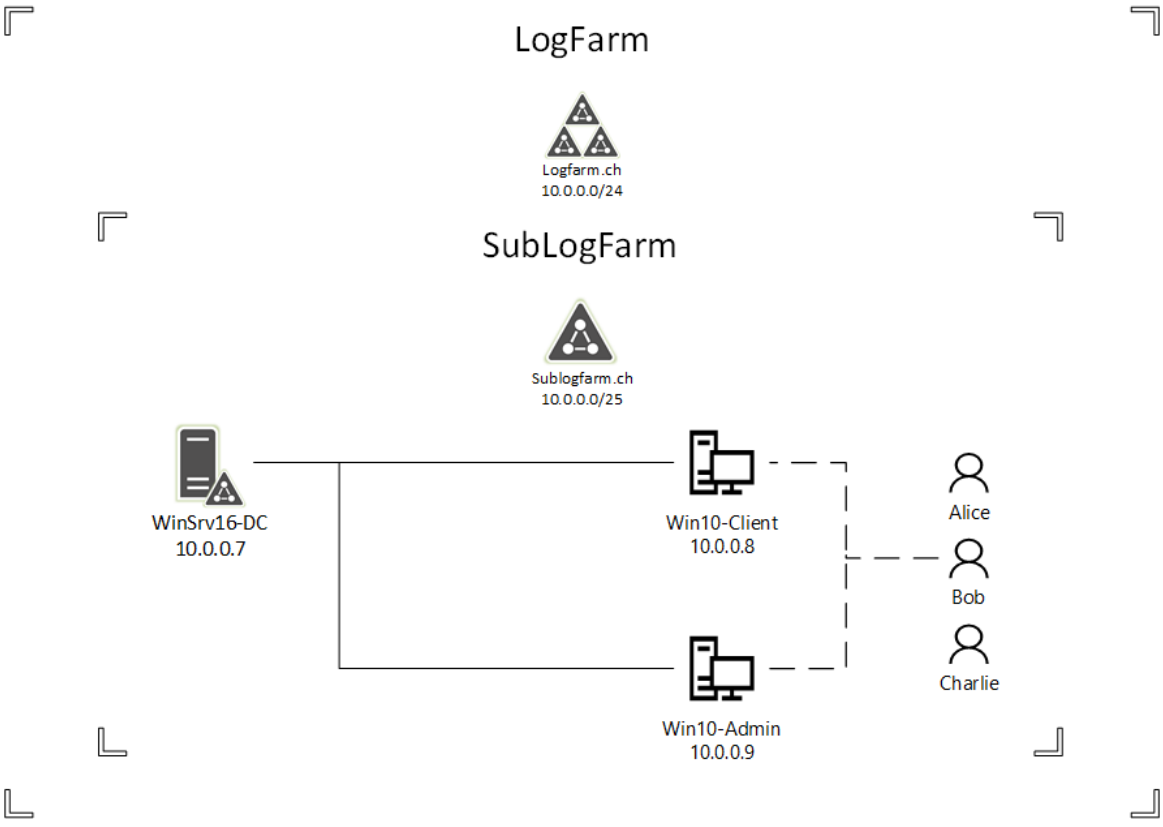

Figure 2.1: Test Environment

2.1 User

Three users were configured for the logfarm-network:

Name	Privileges
alice	Domain administrator
bob	User
charlie	User

Table 2.1: Test Environment User

2.2 Difficulties

Various difficulties occurred which are presented in this subsection.

Connect to the virtuel machines via Remote Desktop Protocol (RDP)
After setting up the virtual machines on Azure, the developers tried to connect to the devices via the Remote Desktop Protocol but failed. First, the developers suspected the issue was the incoming port rules, so the machines were reinstalled. However, this did not fix the issue. It became apparent that the problem were not the virtual machines (VM), but with the network used to connect to the Microsoft Azure Cloud. Some firewall rules blocked the RDP-connection. In order to avoid this, the developers used a Virtual Private Network (VPN) connection in which these rules did not apply.

Firewall setting for Internet Control Message Protocol (ICMP)
After the virtual network had been set up, the developers tested the connections in the virtual network. The configured Domain Name System (DNS) ran without any problem and could translate all hostnames. Testing the network using Pings showed that almost all clients were receiving pings, but the ping-requests by one client remained unanswered. It turned out that, for some inexplicable reason, the incoming ICMP-firewall-settings were different on this client. After adjusting the setting, the ping-requests were answered positively.

RDP connection for Bob and Charlie
Due to the fact that the user `alice` owns domain administrator privileges, this user was able to connect over RDP without an error. Bob and Charlie on the other hand did not have this permissions. The developers had to create a group for them, the RDP-Group. This group was then allowed to login over RDP on the clients Win10-Client and Win10-Admin.

3 Analysis

This chapter describes the first step of this project, the research of published technical reports and tools which are considered interesting for this project.

3.1 BloodHound / SharpHound

3.1.1 Description

BloodHound describes itself on its wiki page on GitHub as follows:

> *"BloodHound is a single page Javascript web application, built on top of Linkurious, compiled with Electron, with a Neo4j database fed by a PowerShell/C# ingestor. BloodHound uses graph theory to reveal the hidden and often unintended relationships within an Active Directory environment. Attacks can use BloodHound to easily identify highly complex attack paths that would otherwise be impossible to quickly identify. Defenders can use BloodHound to identify and eliminate those same attack paths. Both blue and red teams can use BloodHound to easily gain a deeper understanding of privilege relationships in an Active Directory environment." [1]*

3.1.2 Difficulties

BloodHound was tested in the test environment which is described later in this chapter. Both the C# and Python ingestors were successfully installed and tested. The only problem which occurred was that the Python-ingestor does not yet run on the latest Python release. One must have a Python 2.7.x version installed to run the scripts successfully.

3.1.3 Conclusion

The most interesting aspect of BloodHound for our project is the way it retrieves its data. Due to the decision that the application, in a first step, only reads the data of the local computer and not the whole domain, BloodHound will only be important in a later part of the project. Their so called ingestor will be used to retrieve the data of a whole network instead of only a local computer.

3.2 Windows Event Logging Forensic Logging Enhancement Services

3.2.1 Description

Windows Event Logging Forensic Logging Enhancement Services (WEFFLES) is a Threat Hunting/Incident Response Console with Windows Event Forwarding and PowerBI, coded and published by Microsoft-Security-Employee Jessica Payne. It is build to help setting up the Windows Event Forwarding, so that all the collected logs of a system are stored on one centralised server, and afterwards to analyse the collected data. Jessica Payne wrote an installation instruction on the Microsoft TechNet blog https://blogs.technet.microsoft.com/jepayne/2017/12/08/weffles/. Once the data is collected the generated weffels.csv file can simply be imported into Excel and start filtering the logs to gain the needed. Jessica Payne recommends to use PowerBI, a business analytics tool designed by Microsoft. In her published blog she also gives a short introduction on what to look out for, which event ids are important and other useful tips and tricks for detecting suspicious activities in the network.

3.2.2 Conclusion

WEFFELS will not be the product on which this project is based, but could become an important point of reference. The installation guide and other WEFFELS-related documents collected by Jessica

Payne provide a lot of information for reading and understanding logs, which will be very helpful for this project. Also an interesting aspect of WEFFLES and the Jessica Payne article is how she visualised the logs, using Microsoft PowerBI.

3.3 Microsoft Security Compliance Toolkit

3.3.1 Description

The Microsoft Security Compliance Toolkit (SCT) [2] allows security administrators to analyse their configured enterprise Group Policy Objects (GPO) in comparison to the Microsoft-recommended GPO baselines. The toolkit comes with several baseline GPO's for different versions of Microsoft Windows Client and Servers:

- Windows 10 security baselines
  - Windows 10 Version 1803 (April 2018 Update), 1709 (Fall Creators Update), 1703 (Creators Update), 1607 (Anniversary Update), 1511 (November Update), 1507
- Windows Server security baselines
  - Windows Server 2016
  - Windows Server 2012 R2
- Microsoft Office security baseline
  - Office 2016

3.3.2 Difficulties

The toolkit is very simple and could be understood and used without any difficulties. The handling is very intuitive and does not require much training. Please note, however, that the toolkit cannot be used with Windows 10 Home, since active directory support is not provided with this version.

3.3.3 Conclusion

This toolkit can be used for a very baseline GPO in enterprise environment. With the delivered baselines it is easy to compare the configured GPO and to see the readiness of the enterprise GPO. The toolkit enables the comparison of different local GPO's installed on different Clients or Servers to check their consistency. In addition, the provided baselines can be used for building new GPO's. Furthermore, Microsoft delivers with the SCT a Local Group Policy Object Utility (LGPO.exe) to:

- Import and apply policy settings
- Export local policy to a GPO backup
- Parse a registry.pol file to "LGPO text" format
- Build a registry.pol file from "LGPO text"

This toolkit is very interesting, but cannot be used to build on it. The reason for this is that the source code of the complete toolkit is not available. However, it can be used as additional help for checking the readiness of a system and comparing the local policies against templates or other local policies.

3.4 LogonTracer

3.4.1 Description

Japan Computer Emergency Response Team Coordination Centers (JPCERT/CC) LogonTracer is a tool built to investigate malicious logons on a system based on the research described in section "3.11 JPCERT/CC - Detecting Lateral Movement in APTs". The tool links hostnames or Internet-Protocol (IP) addresses with the *"[...] account name found in logon-related events and displays it as a graph"*. [3] The following event ids are checked with the tool:

- 4624: Successful logon
- 4625: Logon failure
- 4768: Kerberos Authentication
- 4769: Kerberos Service Ticket
- 4776: NTLM Authentication
- 4672: Assign special privileges

The following figure depicts a sample graph of logins from different users in the test environment:

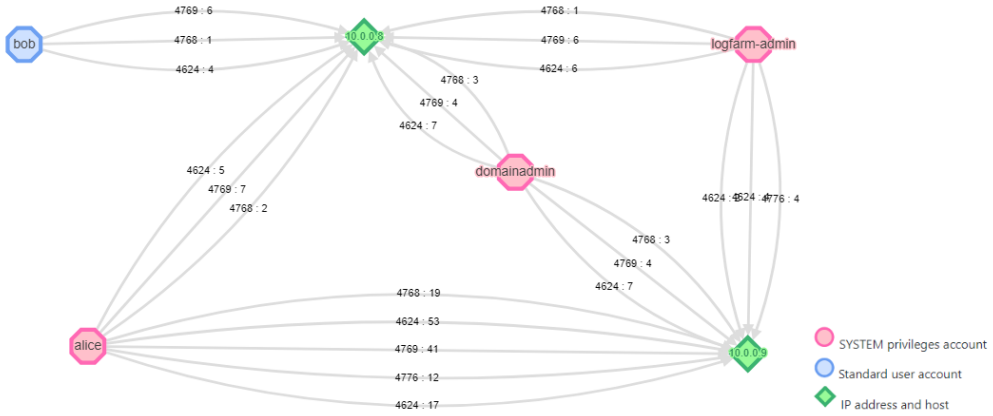

Figure 2.2: LogonTracer: Sample Graph from Test Environment

To use the LogonTracer, only a .evtx-File (Windows Extensible Markup Language (XML) Event Log: export of Windows event logs) is necessary to be uploaded. To get the best result out of LogonTracer an export of the security event log from the domain controller should be used - to get as much information of the network as possible. With the built-in analysis of logins, by using machine learning models and statistical analysis, LogonTracer is able to provide a ranking of the most malicious users which tried to log in. [4]

In addition, LogonTracer provides a timeline for all or selected users to show when each user logged in. The timeline can also be displayed as a graph with the LogonTracer, allowing anomalies to be detected more quickly.

The test environment showed that this graph can quickly become confusing - especially in a larger corporate environment as depicted in figure 2.3 LogonTracer: Confusing Graph from Test Environment. Although only a small environment as described in the section 2 "Test environment" was used, it turned out that various users wanted to log on to the virtual machines. The reason for this is that the test environment was built in the Microsoft Azure Cloud and is accessible via public IP addresses in the cloud.

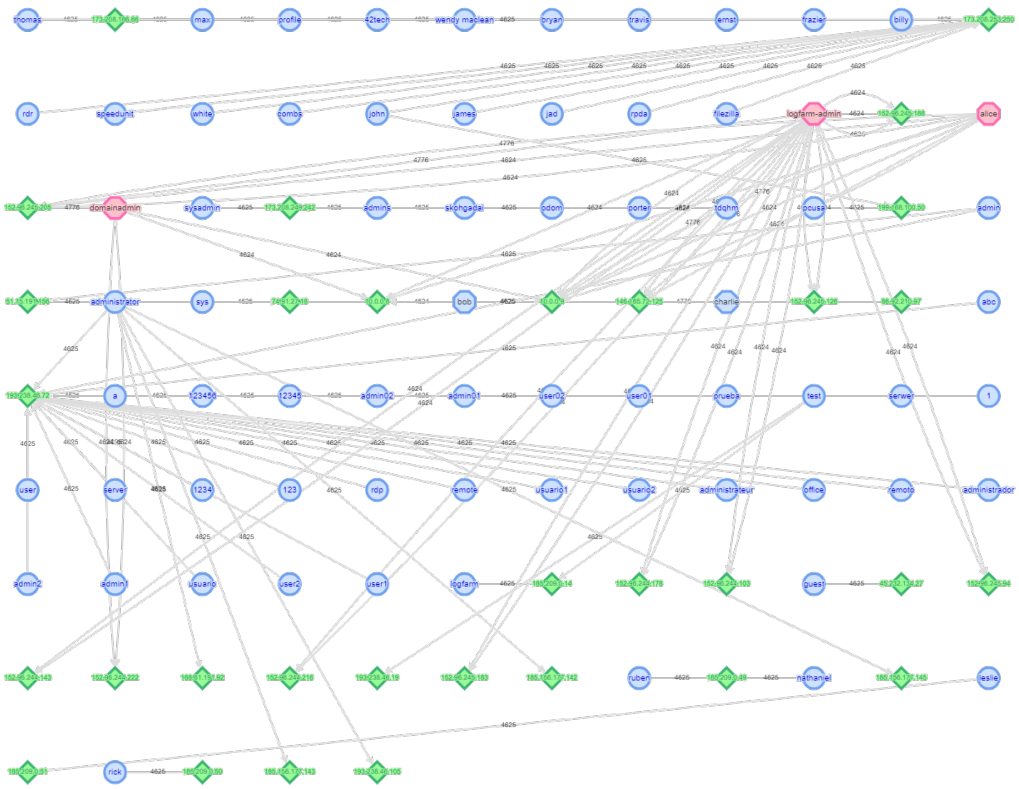

Figure 2.3: LogonTracer: Confusing Graph from Test Environment

Nevertheless, with meaningful filters the search can be restricted and the graph can be used efficiently, as shown in the figure 2.2 LogonTracer: Sample Graph from Test Environment

3.4.2 Difficulties

During the test phase of LogonTracer some difficulties were faced. It is pretty easy to get the docker container, but starting LogonTracer was a bit of a challenge. JPCERT/CC gives the following instructions for starting the docker container:

Listing 2.1: LogonTracer: given docker run command

```
$ docker run --detach \
--publish=7474:7474 --publish=7687:7687 --publish=8080:8080 \
-e LTHOSTNAME=[IP_Address] jpcertcc/docker-logontracer
```

The problem was that the parameter `[IP_Address]` was not described well. If the command `docker ps` was executed it always showed the following `PORTS`:

Listing 2.2: LogonTracer: docker ps (PORTS)

```
PORTS
0.0.0.0:7474->7474/tcp, 0.0.0.0:7687->7687/tcp, 7473/tcp, 0.0.0.0:8080->8080/tcp
```

After some time of investigation and further tests, it turned out that under `PORTS` the ports respectively ip addresses of the container can be bound to the host. But these are not relevant for the LogonTracer, because it provides a web application under the defined parameter `[IP_Address]` and it can eventually be reached via `localhost:8080`. If this parameter was set to `127.0.0.1`, the database containing the imported .evtx file could not be accessed. Thus the graph was never displayed. The parameter `[IP_Address]` set to `localhost` solved this problem.

Listing 2.3: LogonTracer: recommended docker run command

```
$ docker run --detach \
--publish=7474:7474 --publish=7687:7687 --publish=8080:8080 \
-e LTHOSTNAME=localhost jpcertcc/docker-logontracer
```

3.4.3 Conclussion

The LogonTracer is unique in its form and should not be underestimated for the detection of lateral movements. This is because user access to various components available in the network, can be visualised simply and graphically, hence conclusions can be drawn about what has happened.

However, the LogonTracer is not suitable for detection readiness and cannot be used to build on it. Nonetheless, approaches for reading the event log for further work could be used. This tool is also extremely interesting and recommendable for a further detection of lateral movements.

3.5 Microsoft Monitoring Active Directory for Signs of Compromise

3.5.1 Description

This article "Microsoft Monitoring Active Directory for Signs of Compromise" [5] is about configuration of an solid event log monitoring for Microsoft servers. The article gives a quite a good overview about the audit policy in Microsoft systems and what each policy stands for. The article gives information about the most important audit policies and how noisy (if a lot of data is produced by them) they are. This study does not go into the details of the audit policies in detail. Furthermore, the article describes how the policies can be read with powershell.

In this article Microsoft compiles in Appendix L [6] all important event ids which are necessary for a successful detection of APTs and lateral movements.

3.5.2 Conclussion

Due to the fact that audit policies are an important setting for solid event logging, this article and Appendix L will be a central part of the toolkit to be built. As a next step and part of this study these event ids have to be correlated with the event ids found in the JPCERT/CC's study "Detecting Lateral Movement through Tracking Event Logs" [7] to make a clear statement which event ids have to be logged.

3.6 MITRE Adversarial Tactics, Techniques and Common Knowledge (ATT&CK)

3.6.1 Description

MITRE ATT&CK introduces itself on its website as follows:

> *"MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community." [8]*

The portal offers a variety of attacks and their patterns, which are currently known in different operating systems. MITRE ATT&CK describes the attack in short words and then lists possibilities for detection and mitigation. The portal also describes various attack tools, their targets and effects on the system. In addition, the corresponding attacks are always cross-referenced. This is a great advantage for a quick search, especially when time is of the essence.

3.6.2 Conclusion

Although many attacks are described and how they can be detected and fended off, MITRE ATT&CK is not quite suitable for our task. The readiness of a system to detect tailored attacks and lateral movements is only roughly described and would be associated with a time-consuming analysis in order to draw exact conclusions.

3.7 Sysmon

3.7.1 Description

> *System Monitor (Sysmon) is a Windows system service and device driver that, once installed on a system, remains resident across system reboots to monitor and log system activity to the Windows event log. It provides detailed information about process creations, network connections, and changes to file creation time.[9]*

Sysmon logs several events on the system which are partly logged by default too. For example, the event "A new process has been created" with the identifier (ID) 4688 is logged by Sysmon with the ID 1 "Process Creation". The problem is that the default logged event with the ID 4688 logs only the executable file (EXE) name as well as the including path. But bad guys want to stay below the radar, so they might replace the original EXE a with malicious one and rename it like the original. Hence, there is no way to determine with the system based event log entry 4688 if the original EXE was executed. Sysmon eliminates exactly this gap by logging not only the name and path of the EXE but also the hash value of the EXE. Ergo Sysmon brings a big advantage to detect if a malicious EXE was executed or not. For that a reference hash value of the executed EXE is required to compare the hash values on its correctness. [10]

3.7.2 Conclusion

As mentioned in the description Sysmon is a important tool to be enabled for solid detection of attacks. So Sysmon has to be detected if it is running or not to prepare an environment for a good readiness. In order to do not create duplicated events, the events similar logged by default and Sysmon must be examined by their differences. Very likely Sysmon is the better choice.

3.8 Sysmon Tools

3.8.1 Description

Sysmon Tools [11] contains some useful functions to make better use of Sysmon. Among other things there are different views for the representation of the single entries which were recorded by Sysmon. A Process View is provided which can be used to examine a process in more detail. Related processes are taken into account and represented in a simple data-flow-like view, sorted by chronological order. With the Map View you can include geo-locate IP addresses during the import phase and Map View tries to geo-map the network destinations with ipstack [12]. The All Events View represents a full search by Sysmon and can be filtered and grouped accordingly. Furthermore, Sysmon Tools offers a Sysmon Shell, which can be used to create a customized XML configuration for Sysmon using a graphical user interface (GUI). Templates are also provided for further building.

3.8.2 Conclusion

This tool can also be a great help for detecting attacks and, with the Sysmon Shell, a robust configuration for Sysmon can be created. However, Sysmon Tool will have no basis for this project.

3.9 sysmon-modular

3.9.1 Description

With sysmon-modular [13] a clean configuration of the Windows system service System Monitor (Sysmon), an xml-file which is loaded by Sysmon, is provided. Noisy process creations, which are made by legitimate programs, are suppressed as far as possible by Sysmon. The tool offers the possibility and it is expressly recommended by the developer to adapt the configuration to the respective organisation. Furthermore, sysmon-modular implements various attacks in MITRE ATT&CK for detection with Sysmon. It offers the possibility to detect the attacks shown in the figure 2.4 with Sysmon.

Figure 2.4: Detectable attacks with sysmon-modular

3.9.2 Conclusion

Sysmon-modular offers a very good basic configuration for Sysmon based on the platform MITRE ATT&CK which is widely used in the security scene. Unfortunately, sysmon-modular was discovered when decisions were made to develop a tool based on the study "Detecting Lateral Movement through Tracking Event Logs" by JPCERT/CC. The readiness of a system with the basis of MITRE ATT&CK patterns would probably have had an even greater impact. However, Sysmon-modular will most likely not be included in the tool during this study, unless there are still enough time reserves for such an

integration. This tool would better fit the goal to realise a "Readiness Optimizer" as initially mentioned in the task definition.

3.10 CryptoAPI 2.0

3.10.1 Description

The Microsoft feature CryptoAPI 2.0 (CAPI2) Diagnostics provides the ability to collect detailed information about certificate chain validation, certificate store operations and signature verification. CAPI2 in doubt extremely important for any Public Key Infrastructure (PKI) to perform several security based tasks, such as

- Build and verify certificate chains
- Manage per-user and per-computer certificate stores
- Encrypt/decrypt, encode/decode and sign/verify messages

Hence, CAPI2 enables an organisation to secure its communications and business transactions. Identification of users, devices or organisation as well as signed e-mail, code signing and secure web browsing gets possible with todays standards of hash-functions and encryption due to CAPI2. PKI problems are not always easy to troubleshoot and therefor it is necessary to have a good diagnostic capabilities in such cases.

> *CAPI2 Diagnostics in Windows Vista*[1] *provides logging of detailed information about certificate validation, network retrievals, revocation, and other low-level API results and errors. [...] utilizes the event logging and Event Viewer to provide better logging and troubleshooting capabilities for PKI applications based on the CAPI2 API set. [14]*

3.10.2 Conclusion

To detect whether a system is ready for a good detection of lateral movements and APTs CAPI2 is a core component to be logged in every system and CAPI2 Diagnostics must be enabled on a system. Hence, it is required to detect if CAPI2 is enabled on the system. On the other hand, CAPI2 Diagnostics produces a lot of events and therefor the log size should be chosen wisely. For this reason the recommendation of 4 Megabyte (MB) from Microsoft shall be applied. [14]

[1] Windows Vista and above

3.11 JPCERT/CC - Detecting Lateral Movement in APTs

3.11.1 Description

This document [15] is from a presentation by Shingo Abe, a JPCERT/CC employee. In it he describes how to find system intruders more effectively using Windows Event Logs. The collected data is used to to detect inconsistencies more effectively, such as when an administrator logs on to another machine or when an administrator logs on suspiciously often.

3.11.2 Conclusion

This presentation contains interesting information which could be built into the project at a later point. The information this document contains is more suitable for monitoring purposes than for checking the readiness of a system.

3.12 JPCERT/CC - Detecting Lateral Movement through Tracking Event Logs

3.12.1 Description

This is a document [7] JPCERT/CC has published in the year 2017. It describes how, in their experience, attackers proceed with lateral movement. In a very detailed 81-page report they describe the procedure step-by-step, the tools used and what is most interesting for the project, the logs generated while doing so.

3.12.2 Conclusion

This report will have the biggest impact on this project, it shows which logs have to be read in any case. In addition, JPCERT/CC describes in this report which configurations are necessary for solid logging. The appendix not only describes the individual event log IDs, but also the audit policy that can be used to achieve them. For this reason, the checklist to be used will mainly be based on this report. With the provided information we see the greatest potential to develop a suitable tool for the accomplishment of the task in the given time. The given information of the configuration settings in JPCERT/CCs study appendix must be correlated with the "Advanced security auditing Frequently Asked Questions (FAQ)" [16] in order to define the right auditing settings so that the right events are captured.

4 Design

4.1 Decision for a new Tool

At the beginning it was not clear how the tool should be built exactly and what the functionality and scope should be based on. After a detailed analysis of different tools, reports and studies, it was possible to better estimate how an efficient detection of the readiness of a system can be implemented. It would have been desirable to be able to build on an existing tool, but as shown in a five-week analysis, there is no such tool. For this reason it was decided to develop a tool based on JPCERT/CCs study. The configurations in the Advanced Audit Settings of the GPOs are to be checked accordingly and in a second step the event logs are to be searched for the EventIDs.

4.2 Mandatory Event Logs

The following tables lists the event logs which are mandatory and must be logged based on the study "JPCERT/CC - Detecting Lateral Movement through Tracking Event Logs":

System	
EventID	**Description**
8222[2]	Shadow copy has been created
20001[2]	Driver Management concluded the process to install driver

Table 2.2: Mandatory System Event Logs

Applications & Service > Microsoft > Windows > TaskScheduler > Operational	
EventID	**Description**
102[2]	Task completed
106[2]	A task has been registered
129[2]	A task process has been created
200[2]	The operation that has been started
201[2]	The operation has been completed

Table 2.3: Mandatory TaskScheduler Event Logs

Applications & Service > Microsoft > Windows > Windows Remote Management > Operational	
EventID	**Description**
6[2]	Creating WSMan Session
169[2]	User authentication authenticated successfully

Table 2.4: Mandatory Windows Remote Management Event Logs

Applications & Service > Microsoft > Windows > TerminalServices-LocalSessionManager > Operational	
EventID	**Description**
21[2]	Remote Desktop Services: Session logon succeeded
24[2]	Remote Desktop Services: Session has been disconnected

Table 2.5: Mandatory TerminalServices-LocalSessionManager Event Logs

[2]Recorded by default Windows settings

Applications & Service > Microsoft > Windows > Sysmon > Operational	
EventID	**Description**
1[3]	Process create
2[3]	A process changed a file creation time
5[3]	Process terminated
8[3]	CreateRemoteThread
9[3]	RawAccessRead: detects when the process is using "\\.\"

Table 2.6: Mandatory Sysmon Event Logs

Applications & Service > Microsoft > Windows > TaskScheduler > Operational	
EventID	**Description**
102[2]	Task completed
106[2]	Task registered
129[2]	Created Task Process
200[2]	Action started
201[2]	Action completed

Table 2.7: Mandatory TaskScheduler Event Logs

Applications & Service > Microsoft > Windows > WinRM > Operational	
EventID	**Description**
6[2]	Creating WSMan Session
169[2]	User authentication: authenticated successfully

Table 2.8: Mandatory Windows Remote Management Event Logs

Applications & Service > Microsoft > Windows > TerminalServices > LocalSessionManager > Operational	
EventID	**Description**
21[2]	Remote Desktop Services: Session logon succeeded
24[2]	Remote Desktop Services: Session has been disconnected

Table 2.9: Mandatory Windows Local Session Manager Event Logs

[2]Recorded by default Windows settings
[3]Recorded by default Sysmon settings

Security	
EventID	**Description**
104[2]	The System log file was cleared
4624	An account was successfully logged on
4634	An account was logged off
4648	A logon was attempted using explicit credentials
4656	A handle to an object was requested
4658	The handle to an object was closed
4660	An object was deleted
4661	A handle to an object was requested
4663	An attempt was made to access an object
4672	Special privileges assigned to new logon
4673	A privileged service was called
4688	A new process has been created
4689	A process has exited
4690	An attempt was made to duplicate a handle to an object
4720	A user account was created
4726	A user account was deleted
4728	A member was added to a security enabled global group
4729	A member was removed from a security enabled global group
4768	A Kerberos authentication ticket (TGT) was requested
4769	A Kerberos service ticket was requested
4946	A change has been made to Windows Firewall exception list. A rule was added
5140	A network share object was accessed
5142	A network share object was added
5144	A network share object was deleted
5145	A network share object was accessed
5154	WFP has permitted an application or service to listen on a port for incoming connections
5156	WFP has allowed a connection
7036[2]	The service state has changed
7045[2]	A service was installed in the system

Table 2.10: Mandatory Security Event Logs

[2]Recorded by default Windows settings

4.3 Correlation: Advanced Audit Policy Setting and Event Log IDs

In this section, the "Advanced Audit Policies" required to trigger the corresponding event logs are shown in tables. Based on these tables, the "Advanced Audit Policies" are checked for correctness with the tool. There are several combinations of settings which can be configured:

Not Configured:
Nothing selected

No Auditing:
"Configure the following audit events:"

Success (S):
"Success"

Failure (F):
"Failure"

Success and Failure (S, F):
"Success" and "Failure"

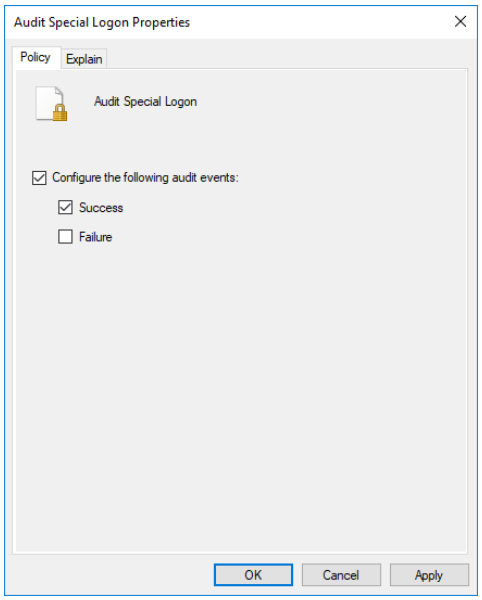

Figure 2.5: Advanced Audit Policy - Logon/Logoff - Audit Special Logon

Account Logon	
Subcategory	**EventIDs**
Audit Kerberos Authentication Service	4768(S, F)
Audit Kerberos Service Ticket Operations	4769(S, F)

Table 2.11: Advanced Audit Policy Setting Account Logon

Account Management	
Subcategory	**EventIDs**
Audit User Account Management	4720(S), 4726(S), 4738(S), 4724(S), 4722(S)
Audit Security Group Management	4728(S, F), 4729(S, F), 4737 (S, F)

Table 2.12: Advanced Audit Policy Setting Account Management

Detailed Tracking	
Subcategory	**EventIDs**
Audit Process Creation	4688(S)
Audit Process Termination	4689(S)

Table 2.13: Advanced Audit Policy Setting Logon/Logoff

Logon/Logoff	
Subcategory	**EventIDs**
Audit Logon	4624(S), 4648(S)
Audit Logoff	4634(S)
Audit Special Logon	4672(S)

Table 2.14: Advanced Audit Policy Setting Logon/Logoff

Object Access	
Subcategory	**EventIDs**
Audit Detailed File Share	5145(S, F)
Audit File Share	5140(S, F), 5142(S), 5144(S)
Audit File System	4656(S, F), 4658(S), 4660(S), 4663(S), 4670(S)
Audit Filtering Platform Connection	5154(S), 5156(S), 5447(S, F)
Audit Handle Manipulation	4658(S), 4690(S)
Audit Kernel Object	4656(S, F), 4658(S), 4660(S), 4663(S)
Audit Other Object Access Events	4698(S, F)
Audit Registry	4656(S, F), 4658(S), 4660(S), 4663(S)
Audit SAM	4661(S, F)

Table 2.15: Advanced Audit Policy Setting Object Access

Policy Change	
Subcategory	**EventIDs**
Audit MPSSVC Rule-Level Policy Change	4946(S)

Table 2.16: Advanced Audit Policy Setting Policy Change

Privilege Use	
Subcategory	**EventIDs**
Audit Non Sensitive Privilege Use	4673(S, F)
Audit Sensitive Privilege Use	4673(S, F)

Table 2.17: Advanced Audit Policy Setting Privilege Use

4.4 Domain Analysis

The following section describes the problem domain which is faced during this project. Despite the decision to not programm an object orientated solution, there are several things to be aware of and to think through carefully. For this reason building a domain model is a simple and suitable suitable technique to use for. The following figure 2.6 shows the domain model and will be explained in some details afterwards.

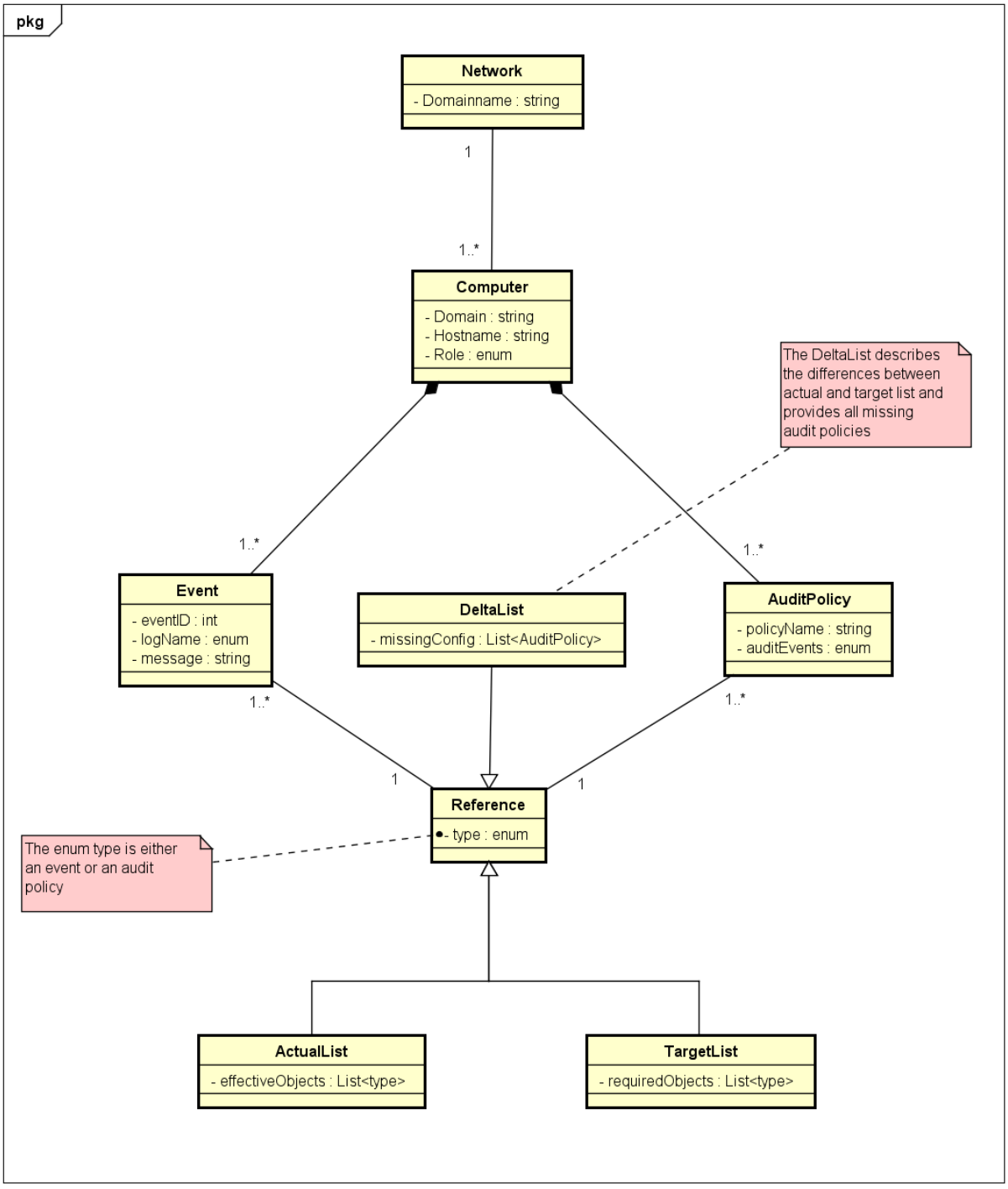

Figure 2.6: Domain Model

4.4.1 Network

The class network depicts the organizations wide network which is used to connect all clients and servers together. In this project the main goal is to locally detect the readiness of the system and not to extend the detection for a system-wide infrastructure. For further development on this project and a system-wide extension, the network is already considered in this domain model.

4.4.2 Computer

A computer illustrates either a client like a Windows 10 machine or a server in particular a domain controller running on a Windows Server 2016. In principle, however, every Windows computer is represented. A computer is a core component in our project, because the detection is done on a single client or server.

4.4.3 Event

An event represents a single event log entry in simplified form.

4.4.4 AuditPolicy

AuditPolicy displays the individual settings of the audit policies of the group policy, which can be found via `gpedit.msc` under "Computer Configuration > Windows Settings". However, only the settings under "Security Settings > Advanced Audit Policy Configuration" are considered and not the settings under "Security Settings > Local Policies > Audit Policy". The reason for this is that Microsoft recommends that only one of the two policies is used:

> *[...] do not use both the basic audit policy settings under Local Policies\Audit Policy and the advanced settings under Security Settings\Advanced Audit Policy Configuration. Using both basic and advanced audit policy settings can cause unexpected results in audit reporting.* [16]

A single audit policy setting represents one or more event IDs logged by this configuration.

4.4.5 Reference

ActualList The ActualList represents the current state of the system. It reflects the event log IDs that have occurred and the audit policies that have been set.

TargetList The TargetList represents either the list of event logs or configured audit policies which must be present for a solid detection of attacks.

DeltaList Based on the required lists (audit policies, event logs) and the current state of the computer, the DeltaList shows which settings are missing in the audit policies.

5 System Architecture

In this section the following main question is answered:

"What would a system architecture look like to fulfill the described problem domain?"

This includes the coverage of use cases, non-functional requirements, technologies used and how the tool will be designed.

5.1 Use Cases (UC)

A visual representation of the use cases with a use case diagram was deliberately omitted, because there is only one actor involved - the security advisor. The actor is not specifically mentioned in the use cases every time, because it is always the same. During the elaboration phase it was decided in consultation with the client that the project would be limited to a Readiness Analyser only.

5.1.1 UC01 - Read Resultant Set of Policies

Description
The specified audit policies are read and saved in a temporary file.

Precondition
The system is running and the tool must possess administrator permissions.

Main Success Scenario

1. Read the specified audit policies from the system
2. Save the needed information from the audit policies in a temporary file for analysis purposes.

5.1.2 UC02 - Analyse Audit Policies

Description
The list which was created in UC01 is compared to a "perfect settings"-list. Missing or wrong settings are going to be exported into a separate file.

Precondition
UC01 is fulfilled: the temporary file is available.

Main Success Scenario

1. The temporary files can be read
2. Creates a list of incorrect settings

5.1.3 UC03 - Find Event Logs

Description
Event logs are search by ID and marked in an external file as found or missing.

Precondition
The system is running and must have valid event logs. The tool must possess administrator permissions.

Main Success Scenario

1. Search for the specified event logs from the local system
2. Save the result from the search in a temporary file for analysis purposes.

5.1.4 UC04 - Analyse Found Event Logs

Description
The implemented logic analyses, by defined event ids, which events occurred or are missing and creates a list of events that did not occurred or are not logged yet.

Precondition
UC03 is fulfilled: the temporary file is available.

Main Success Scenario

1. The temporary file can be read
2. The list with the defined event ids is available
3. Create a list of events which occurred and which are missing

5.1.5 UC05 - Display missing or wrong system configuration

Description
Based on the list created in UC02 and UC04 the user gets an overview of missing configurations (the result) which would improve the readiness of the system for a good attack detection.

Precondition
The lists from UC02 and UC04 are available.

Main Success Scenario

1. Displays a visual output of missing or wrong system configurations

5.1.6 UC06 - Save Result to specific path

Description
The actor has the possibility to save the overview from UC05 to a file in a specific path defined by the actor himself. This file contains the result from UC05 in a descriptive way.

Precondition
UC05 is fulfilled: the result, respectively the overview is available

Main Success Scenario

1. A file is saved to a specific path with the result from UC05
2. The path can be defined by the actor

5.1.7 UC07 - Main Script

Description
The actor is able to use the implemented functionalities in an easy way. Therefor the actor requires the script to be used with simple arguments to run the script in its different given modes. More specific the actor should be faced with the possibility to run the script online (check the current system) and offline (check any system with provided exports). In addition, the actor is able to call a help function of the script to get more information about the script itself and how to use it.

Precondition
All functions and process flows have to be implemented and defined.

Main Success Scenario

1. The actor can call all functionalities just through the main script with appropriate arguments
2. The actor can call a helper function to get information how the script is supposed to use

5.1.8 UC08 - Get Domain Information

Description
The actor has the possibility to gather information about single or all domain group policies. This information should be processed and analyzed in the same way as the local gathered data.

Precondition
Access to `SYSVOL` is possible.

Main Success Scenario

1. The actor gets a result about the readiness of domain group policies which are of interest.

5.2 Non Functional Requirements

NFR-No.	Description
NRF01	After using the Toolkit the system must remain in the status quo. More specifically the system shall not deliberately alter any existing entry in the event logs and registry. However, the tool may produce new event logs.
NFR02	The user shall not notice significant performance degradation from the system when using the Toolkit.
NFR03	The Toolkit must be portable with no installation procedure before use.
NFR04	The minimal target version of the system for the Toolkit to run must be Microsoft Windows 10 Professional or Microsoft Server 2016.
NFR05	The Toolkit runs in one go, but can also be executed in single steps with the possibility to skip single steps (pause/abort in case of performance problems)

Table 2.18: Non Functional Requirements

5.3 Technologies

5.3.1 Chosen Technologies

PowerShell & Visual Studio Code
The decision as to which technology to use, was made in favour of PowerShell. The reason why PowerShell was used, was that it is close to the Microsoft Operating System and that it has a large and detailed documentation at its disposal.

The scripts are written in Visual Studio Code with the extension packet "PowerShell". Visual Studio code is preferred to PowerShell Integrated Scripting Environment (PowerShell ISE) because it only requires working in one Integrated Development Environment (IDE) for implementation and documentation.

LaTeX& Visual Studio Code
The documentation is written with LaTeX in Visual Studio Code wit the LaTeX Workshop extension. The main reason for LaTex was that the developers are already familiar with it. Furthermore, LaTeX offers a very simple way for referencing sources. On the other hand, we made the experience that with LaTeX the formatting is more reliable than for example when Microsoft Word is used.

Azure Cloud
The test environment is set up, as described in section 2 "Test environment", in the azure cloud. One server and two clients form a virtual network, this enables developers to access it from anywhere to any given time. A disadvantage is the changing public IP-addresses to access the VMs. In the end, the advantages outweigh the disadvantages.

GitHub
GitHub is used as a version control tool for source code and documentation. GitHub has been elected because of its good reputation and the experience the developers already gained with.

Continuous Integration
Continuous Integration (CI) for Powershell is unfortunately not very widespread as has been shown after some time of research. Fortunately the article "Converting a PowerShell Project to use Azure DevOps Pipelines" [18] by Daniel Scott-Raynsford was found, which describes in detail how a CI environment can be set up in Microsoft Azure DevOps. Due to the fact that Azure DevOps offers a very simple and clear handling, as well as supports all common operating systems (Linux, Windows and MacOS), it was decided to set up the CI environment in Azure DevOps. The structure and the important findings are described in the Continuous Integration manual.

5.3.2 Rejected Technologies

Python
The decision to use PowerShell and maybe C# for a GUI instead of Python was made because the developers do not have much experience with Python. Also PowerShell is closer to the Microsoft operating system. With Python there is no guarantee that the libraries which would be used are as powerful to solve the requirements.

5.4 Sequence Diagram

This section describes the process of the toolkit and explain the individual steps in detail. As mentioned in the Use Cases, the actor of this toolkit will be a security advisor, who will execute the toolkit.

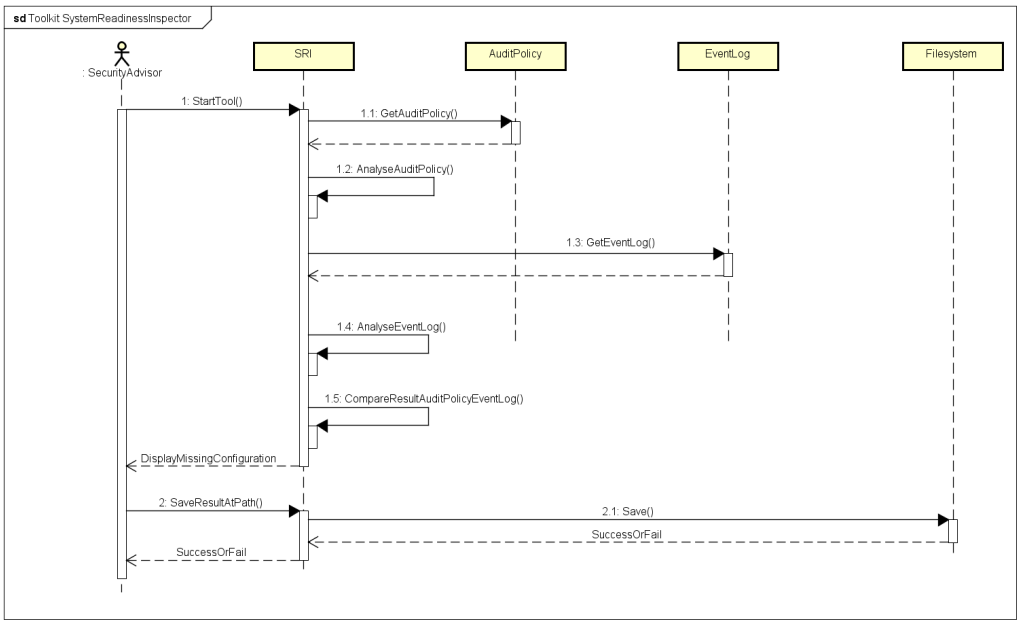

Figure 2.7: Sequence Diagram SystemReadinessInspector - SRI

5.4.1 GetAuditPolicy()

This task is responsible to get all Audit Policies, which are relevant for logging the right events according to JPCERT/CCs study. To gather all information about the Audit Policies and the current state of its configuration the Resultant Set of Policies (RSoP) must be read. [19] RSoP is a Microsoft snap-in to create a detailed report about the applied policy settings.

5.4.2 AnalyseAditPolicy()

In this task the gathered information from the task GetAuditPolicy(), which is represented as a XML-File, is going to be analysed against the recommendation from JPCERT/CCs study (see 3.12 JPCERT/CC - Detecting Lateral Movement through Tracking Event Logs). the result of this analysis will be stored in a XML-based format in a temporary file.

5.4.3 GetEventLog()

This task is responsible for getting the event logs from the system. Therefor the command `Get-EventLogs` [20] retrieves all logs from 'System', 'Application' and 'Security'. This logs are, to be analysed later, saved as a 'CSV' file to the current path were the PowerShell is running.

5.4.4 AnalyseEvent()

In this task the created command-separated values file (CSV) from GetEventLog() is used to analyse the collected logs. They are compared to a list provided by JPCERT/CC to find out how ready the system would be if an enemy attack had taken place. The result of this comparison will be stored as a 'XML' file in order to visualise it.

6 Implementation

6.1 Module: GetAndAnalyseAuditPolicies

The basic idea was to implement the use case "UC01 - Read Resultant Set of Policies" separately from the use case "UC02 - Analyse Audit Policies". However, during the implementation it quickly became clear that these two use cases could be merged and did not have to be implemented separately. Therefore, both use cases were integrated into one script. The following describes how the two use cases were implemented.

6.1.1 Result

Both Use Cases were implemented together in one script. The script follows the following schedule:

- Reading and caching of the RSoP
- Verification that all defined audit policies are in place
- Verification that all defined audit policies are correctly configured
- Check if "Audit: Force audit policy subcategory settings (Windows Vista or later) to override audit policy category settings" is enabled in registry to prevent conflicts between security settings
- Check if Sysmon is installed and running as a service
- Check whether CAPI2 is enabled

Each result of the individual steps is collected in hashtables and merged together to be exported to a XML file. Finally, the environment and files that are no longer needed are deleted, so that only the result XML is available for further processing. A result could possibly look like the following listing:

Listing 2.4: Example Result Audit Policy Analysis

```
<?xml version="1.0" encoding="utf-8"?>
<AuditPolicies>
   <AuditNonSensitivePrivilegeUse>NotConfigured</AuditNonSensitivePrivilegeUse>
   <AuditOtherObjectAccessEvents>NotConfigured</AuditOtherObjectAccessEvents>
   <AuditUserAccountManagement>NotConfigured</AuditUserAccountManagement>
   <AuditKernelObject>NotConfigured</AuditKernelObject>
   <AuditSAM>NotConfigured</AuditSAM>
   <AuditKerberosAuthenticationService>NotConfigured</AuditKerberosAuthenticationService>
   <AuditHandleManipulation>NotConfigured</AuditHandleManipulation>
   <AuditRegistry>NotConfigured</AuditRegistry>
   <AuditProcessTermination>NotConfigured</AuditProcessTermination>
   <AuditFileSystem>NotConfigured</AuditFileSystem>
   <AuditMPSSVCRule-LevelPolicyChange>NotConfigured</AuditMPSSVCRule-LevelPolicyChange>
   <AuditSpecialLogon>NotConfigured</AuditSpecialLogon>
   <AuditFileShare>NotConfigured</AuditFileShare>
   <AuditLogoff>NotConfigured</AuditLogoff>
   <AuditDetailedFileShare>NotConfigured</AuditDetailedFileShare>
   <AuditSensitivePrivilegeUse>NotConfigured</AuditSensitivePrivilegeUse>
   <AuditLogon>NotConfigured</AuditLogon>
   <AuditFilteringPlatformConnection>NotConfigured</AuditFilteringPlatformConnection>
   <AuditProcessCreation>NotConfigured</AuditProcessCreation>
   <ForceAuditPolicySubcategory>Enabled</ForceAuditPolicySubcategory>
   <Sysmon>InstalledAndRunning</Sysmon>
   <CAPI2LogSize>4194304</CAPI2LogSize>
   <CAPI2>EnabledGoodLogSize</CAPI2>
</AuditPolicies>
```

6.1.2 Approach

Read Resultant Set of Policies

Research was done to read the corresponding audit policy configurations from the system. At the beginning, the approach was to read the required configurations using the command `auditpol`. [21] This command can be used to read out and manipulate the currently valid information on the audit policies. However, the manipulation of the audit policies is not necessary within the tool and can be ignored. The command provides exactly the information needed to fulfill this use case:

Listing 2.5: auditpol

```
PS C:\Windows\system32> auditpol /get /category:Logon/Logoff
System audit policy
Category/Subcategory                    Setting
Logon/Logoff
   Logon                                Success and Failure
   Logoff                               Success and Failure
   Account Lockout                      No Auditing
   IPsec Main Mode                      No Auditing
   IPsec Quick Mode                     No Auditing
   IPsec Extended Mode                  No Auditing
   Special Logon                        Success and Failure
   Other Logon/Logoff Events            No Auditing
   Network Policy Server                No Auditing
   User / Device Claims                 No Auditing
   Group Membership                     No Auditing
```

Unfortunately, this output is not very ideal for a suitable further processing and analysis of the current configuration. The return value of the command is an ordinary array filled with corresponding strings and therefore the complete array should have been checked for correct content by string comparisons. Furthermore, the command `auditpol` does not offer the possibility of remote configuration with regard to an extension of the tool to a whole fleet of computers. For this reason, the idea of building the tool on the basis of this command was rejected.

Further research has shown that Microsoft provides a RSoP [19] for reading audit policies. This can also be accessed via a PowerShell command. Microsoft offers the command `Get-GPResultantSetOfPolicy` [22] for this purpose. This command can be used to generate an XML-based report of the currently valid GPOs. Since traversing an XML-based file via PowerShell proves to be very simple, this variant is preferable to the `auditpol` command. After a short test, it quickly became clear that the generated XML provides all necessary information for the further analysis. Unfortunately, the `Get-GPResultantSetOfPolicy` command is not available by default on all systems. However, this command is used and the missing Module: "GroupPolicy", which is used to activate the command, will be prerequisite for the script. [23] [24]

Analyse Audit Policies

The current configuration of the system's audit policies is then to be evaluated from the temporarily cached file. The basis for this provides the section "4.3 Correlation: Advanced Audit Policy Setting and Event Log IDs" based on "3.12 JPCERT/CC - Detecting Lateral Movement through Tracking Event Logs".

6.1.3 Implementation

This section describes the implementation of `GetAndCheckAuditPolicies` in detail. For this purpose, the following is referred to in the section 6.1.1 Result described schedule.

The first step is to read the RSoP from the local system with the command `Get-GPResultantSetOfPolicy`. The XML that is retrieved is then temporarily cached in the execution path of the script and read in again for further processing.

Listing 2.6: Get-GPResultantSetOfPolicy

```
try {
   Get-GPResultantSetOfPolicy -ReportType Xml -Path $pathRSOPXML | Out-Null
}
catch {
   Write-Host "Necessary Module: ''GroupPolicy'' is not provided
               within this system" -ForegroundColor Red
   return
}

if ([System.IO.File]::Exists($pathRSOPXML)) {
   [xml]$rsopResult = Get-Content $pathRSOPXML
}
```

The generated XML (RSoP) is an extraction of the GPOs and contains only the configurations set from them. Afterwards the analisis begins and the entries are searched in the XML file, in which the required configurations for the "Advanced Audit Policies" are stored (see figure 2.8).

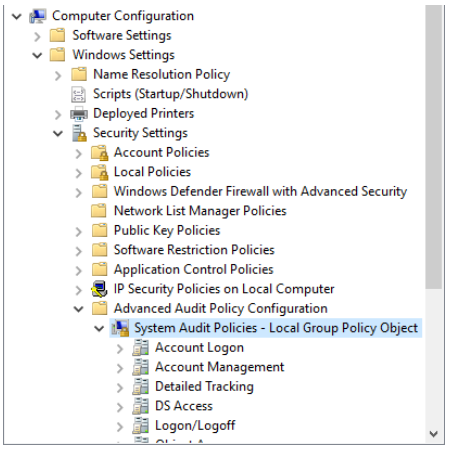

Figure 2.8: GPO - Advanced Audit Policies

The first step of the analysis is to search for missing configurations. The system iterates over the queried `AuditSettings` and searches for missing configurations. Any missing setting will be written in a hashtable `result` for further processing.

Listing 2.7: GetAndCheckAuditPolicies: Search missing configurations

```
$auditSettings = $rsopResult.Rsop.ComputerResults.ExtensionData.Extension.AuditSetting

foreach ($auditSettingSubcategoryName in $targetAuditSettings) {
   if ($auditSettings.keys -notcontains $auditSettingSubcategoryName) {
      $result.Add(($auditSettingSubcategoryName -replace (" ")), "NotConfigured")
   }
}
```

After checking for missing configurations the set settings are checked for correctness according to JPCERT/CC - Detecting Lateral Movement through Tracking Event Logs.

Listing 2.8: GetAndCheckAuditPolicies: Get configured audit settings from RSoP

```
foreach ($auditSetting in $auditSettings.GetEnumerator()) {
   if ($targetAuditSettings -notcontains $auditsetting.name) {
      continue
   }
   if ($auditSetting.value -and $auditSetting.name) {
      try {
         $auditSettingValue = $auditSetting.value
      }
      catch {
         $auditSettingValue = 0
      }
      $auditSubcategoryName = $auditSetting.name
      switch ($auditSettingValue) {
         NoAuditing {
            $auditSettingValueString = "NoAuditing"
            continue
         }
         Success {
            $auditSettingValueString = "Success"
            continue
         }
         Failure {
            $auditSettingValueString = "Failure"
            continue
         }
         SuccessAndFailure {
            $auditSettingValueString = "SuccessAndFailure"
            continue
         }
         Default { continue }
      }
      $result.Add(($auditSubcategoryName -replace (" ")), $auditSettingValueString)
   }
}
```

When checking the audit policies is done the next step is to verify if the setting "Audit: Force audit policy subcategory settings (Windows Vista or later) to override audit policy category settings" is enabled as considered in section "4.4.4 AuditPolicy". This had to be solved via the registry, because this information is not available in the RSoP.

Listing 2.9: GetRegistryValue

```
Function GetRegistryValue($path, $name) {
   return Get-ItemProperty -Path $path -Name $name -ErrorAction Stop
}
```

The registry entry is captured in a separate function to provide a the possibility for tests. This function is called with the following parameter [25] to get the searched registry entry:

```
$path = "HKLM:\System\CurrentControlSet\Control\Lsa"
$name = "SCENoApplyLegacyAuditPolicy"
```

Listing 2.10: Function IsForceAuditPolicyEnabeled

```
Function IsForceAuditPolicyEnabeled ($auditPolicySubcategoryKey) {
   $result = @{}

   if ($auditPolicySubcategoryKey) {
      if ($auditPolicySubcategoryKey.SCENoApplyLegacyAuditPolicy -eq 1) {
         $result.Add("ForceAuditPolicySubcategory", "Enabled")
         return $result
      }
      else {
         $result.Add("ForceAuditPolicySubcategory", "Disabled")
         return $result
      }
   }
   else {
      $result.Add("ForceAuditPolicySubcategory", "NotDefined")
      return $result
   }
}
```

The next step is to check if Sysmon as a service is installed (also not contained in the RSoP) and if so is it running or not. Since a service can be renamed to hide it from the bad guys, the `Get-Service` command cannot make a 100% statement about whether the service is actually installed. For this reason the description of the service is queried, which does not change with renaming. [10]

Listing 2.11: Function IsSysmonInstalled

```
Function IsSysmonInstalled($service) {
   Write-Host "Check Sysmon"
   $service = Get-CimInstance win32_service -Filter "Description = 'System Monitor
       service'"
   $result = @{}

   if ($service) {
      if ($service.State -ne "Running") {
         $result.Add("Sysmon", "InstalledNotRunning")
         return $result
      }
      else {
         $result.Add("Sysmon", "InstalledAndRunning")
         return $result
      }
   }
   else {
      $result.Add("Sysmon", "NotInstalled")
      return $result
   }
}
```

As a last step the script is checking whether CAPI2 is enabled and has the right minimum log size of 4MB. The decision for 4MB is mentioned in the section "3.10 CryptoAPI 2.0". Unfortunately, this information is also not available via the RSoP. Therefore the command `wevtutil` is used to query CAPI2 in the event log. The reason for this is that CAPI2 can only be enabled via the Event Viewer. [14] In order to enable testing here as well, a Get function for the event log entry has been created.

Listing 2.12: Function GetCAPI2

```
Function GetCAPI2 {
   return wevtutil gl Microsoft-Windows-CAPI2/Operational /f:xml
}
```

The log size is stored in the Windows system as mebibyte (MiB). This is the reason for defining the initial log size to `4194304`. The following conversion from 4 MB to mebibyte should make it clear:

$$4 \text{ MB} = 4 \cdot 1024 \cdot 1024 \text{ Bytes} = 4194304 \text{ Bytes}$$

Listing 2.13: Function IsCAPI2Enabled

```
Function IsCAPI2Enabled([xml] $capi2, [uint32] $requiredLogSize) {
   $capi2Enabled = $capi2.channel.enabled
   $currentLogSize = $capi2.channel.logging.maxsize -as [uint32]
   $result = @{}

   if ($requiredLogSize -lt 4194304) {
      $requiredLogSize = 4194304
   }

   if ($capi2Enabled -eq "true" -and $currentLogSize -ge $requiredLogSize) {
      $result.Add("CAPI2", "EnabledGoodLogSize")
      $result.Add("CAPI2LogSize", "$currentLogSize")
   }
   elseif ($capi2Enabled -eq "true" -and $currentLogSize -lt $requiredLogSize) {
      $result.Add("CAPI2", "EnabledBadLogSize")
      $result.Add("CAPI2LogSize", "$currentLogSize")
   }
   else {
      $result.Add("CAPI2", "Disabled")
   }
   return $result
}
```

At the end of the script all temporary files are removed.

To fulfill UC08 - Get Domain Information (see section 5.1.8) two functions where created to gather the advanced audit settings from group policies. One for gathering information about a specific group policy and the other for all group policies. To achieve this, the information is gathered from the System Volume (SYSVOL) where all group policies rely in a Active Directory Network.

Listing 2.14: Function GetDomainAuditPolicies

```
Function GetDomainAuditPolicies ($policyName) {
   $domain = Get-WmiObject Win32_ComputerSystem -ComputerName "localhost"
             | Select-Object -ExpandProperty Domain

   # Check if system is in $domain
   # Check if $policyName exists

   $policyId = Get-GPO -Name $policyName | Select-Object -ExpandProperty id
   $policyCSVPath = "\\$domain\SYSVOL\$domain\Policies\{$policyId}
                     \MACHINE\Microsoft\Windows NT\Audit"

   if (Test-Path $policyCSVPath) {
      $policyCSV = $policyCSVPath + "\audit.csv"
   }
   else {
      Write-Host "For this Group Policy exist no definition"
      return
   }

   if ([System.IO.File]::Exists($policyCSV)) {
      $auditSettings = @{}
      $policy = Import-Csv $policyCSV -Encoding UTF8
      foreach ($element in $policy) {
         $auditSettings.Add($element.Subcategory, $element."Setting Value")
      }
      return $auditSettings
   }
   else {
      Write-Host "For this Group Policy exist no auditing definition"
      return
   }
}
```

6.2 Module: GetAndCompareLogs

This section describes the implementation of the "UC03 - Find Event Logs" as well as "UC04 - Analyse Found Event Logs". Both use cases were fulfilled in the PowerShell script "GetAndCompareLogs". Here is a description how the use cases were implemented.

6.2.1 Result

The script "GetAndCompareLogs", where both use cases were implement, runs as follows:

- Reading and caching the Event Logs "System" & "Security"
- Filter cached Logs by EventID, group EventIDs that occur more than once. Found EventIDs are exported as "CSV"
- Checking and caching whether a list of EventIDs from "Application and Service" Logs can be read out
- Export result set of found EventIDs as "CSV"
- Import list of found Event Logs and compare it with the predefined checklist
- Result of the comparison is written into an "XML" file
- Import and compare found Application and Service Logs with predefined checklist
- Result of the comparison is written into the same "XML" as before

The now no longer needed CSV files are deleted. The XML with the result set is now available for any further processing. A result could possibly look like the following listing:

Listing 2.15: Example Result Audit Policy Analysis

```
<?xml version="1.0"?>
<Logs>
   <EventLogsID>
      <6>present</6>
      <21>missing</21>
      <24>missing</24>
      <102>missing</102>
      <104>missing</104>
      <106>missing</106>
      <201>missing</201>
      <4624>present</4624>
      <4634>present</4634>
      <4648>present</4648>
      <4656>present</4656>
      ...
   </EventLogsID>
   <AppAndServID>
      <106>present</106>
      <200>present</200>
      <129>present</129>
      <201>present</201>
      <102>present</102>
      <6>missing</6>
      <169>missing</169>
      <21>present</21>
      <24>present</24>
   </AppAndServID>
</Logs>
```

6.2.2 Approach

Get Event Logs
After research was done on how to read out the Event Logs "System" and "Security" the desicon was made to use to PowerShell command`Get-EventLog` [20]. This command allows to read out the whole EventLog by the LogName or also to search after a specific EventID. The first approach was to search for each EventID individually. The EventIDs to search for were taken from the JPCERT/CC Appendix B in the "Detecting Lateral Movement through Tracking Event Logs" report. [7]. The script run successfully, but the runtime was not practicable. It took over 5 minutes to search for all EventIDs in an Event Log of the size of about 37 000 Logs, or in other words 300 Kilobyte (KB). The developers than started to calculate the worst case scenario, in this case the worst case scenario is that none of the searched EventIDs is found in the EventLog. There are n EventIDs in the checklist and m entries in the EventLogs, if no EventID is found, every entry is called m times. That results in O(n*m). The developers decided to cache the Event Logs, reducing the runtime to O(m). The cached Logs are then grouped into EventIDs and export into a "CSV" file.

To read out the "Application and Service" Logs we can not use Get-EventLog. The first approach used the `Get-WinEvent` [26] command. The logic stayed the same, read out all events, group and export them into a 'CSV' file. Unfortunately the Get-WinEvent is very slow, it took over 10 minutes to read out just under 6000 logs. The developers found an other, much quicker command called `wevtutil` [27]. Unfortunately it is not quite simple to read out all Logs, for that reason each EventID will be searched if it appeared. Unlike Get-EventLog, this is not a problem because the command is faster, the EventIDs are more likely to occur and the amount of Logs is smaller. On the testing environment with a machine with 4 Gigabyte (GB) Ram and an Intel Xeon E5 with 2 cores it took about 10 seconds to check for 9 EventIDs in 15 000 Log entries. If and EventID was found it was added to an ArrayList, after all IDs are checked the file is exported as a 'CSV'.

Analyse Found Event Logs
To analyse the occurred EventIDs the two generated "CSV" files are imported into the PowerShell script. The respective checklists, which are based on the JPCERT/CC - Detecting Lateral Movement through Tracking Event Logs, are embedded in the script. Each id from the checklist is checked if it is present in the respective CSV file. Is this the case, the id is added to the XML-file and marked as present. Did the id no occur in the it will be added and marked as missing. The file looks like the example in "Result" shown.

6.2.3 Implementation

This section describes the implementation of `GetAndCompareLogs` in detail. For this purpose, the following is referred to in the section "6.2.1 Result" described schedule.

The first step is to read out the "System" and "Security" Logs. To achieve this goal the command `Get-EventLog` is used in the first part of the function `GetEventLogsAndExport`.

Note The code has been adapted for better readability and easier understanding

Listing 2.16: Function GetEventLogsAndExport Part 1

```
1  $logNames = @("System", "Security")
2  $eventLogs = New-Object System.Collections.ArrayList
3
4  Function GetEventLogsAndExport{
5     foreach($log in $logNames){
6     $eventLogs += Get-EventLog -LogName $log
7     }
8  ...
```

The second part of the function filters the EventIDs from the chaced logs. Subsequently, multiple EventIDs are grouped together.

Listing 2.17: Function GetEventLogsAndExport Part 2

```
1  $currentFolder = (Resolve-Path .\).Path
2  $exportEventLogsIntoCSV=$currentFolder + "\myeventlogs.csv"
3
4  $eventLogs| Select EventID -Unique |Export-CSV $exportEventLogsIntoCSV -NoTypeInfo
       -Encoding UTF8
5  }
```

After the export the function `GetApplicationAndServiceLogs` is called. As before, the function is divided into two parts, first how to get the data. The same procedure is used three times, for the "TaskScheduler", "WindowsRemoteManagement" and "LocalSessionManager". Due to the fact that the code is very similar it is only shown once. To search for the EventIDs `wevtutil` is used.

Listing 2.18: Function GetApplicationAndServiceLogs Part 1

```
1  $appAndServLogs = New-Object System.Collections.ArrayList
2  $idsForTaskScheduler = ("106", "200", "129", "201", ...
3
4  $appAndServLogs += "EventID"
5
6  Function GetApplicationAndServiceLogs{
7
8  foreach($id in $idsForTaskScheduler){
9  if(wevtutil qe Microsoft-Windows-TaskScheduler/Operational
       /q:"*[System[(EventID="$id" )]]" /uni:false /f:text){
10    $appAndServLogs += $id
11     }
12 }
13 ...
```

After all three Logs were checked and all found EventIDs were added, the information is exported into a "CSV"-file.

Listing 2.19: Function GetApplicationAndServiceLogs Part 2

```
$exportApplicationAndServiceLogsIntoCSV = $currentFolder +
    "\myapplicationandservicelogs.csv"

$appAndServLogs | Out-File -FilePath $exportApplicationAndServiceLogsIntoCSV
}
```

The next point on the list is importing the found "EventLogs" and "Service And Application" Logs. Due to the similarity we only show one code.

Listing 2.20: Function ImportCompareExport

```

$eventLogIdsToCheck = (6, 21, 24, 102, 104, 106, 129, ...

# Create XML "resultOfEventLogs.xml"

$importEventLogs = $exportEventLogsIntoCSV
$myEventLogs = Import-Csv $importEventLogs -Encoding UTF8

Function ImportCompareExport{
   foreach($id in $eventLogIdsToCheck){
     if($myEventLogs | where {$_.EventID -eq $id}){
      # Write to XML with value "present"
         }
         else{
            # Write to XML with value "missing"
         }
   }
}
# Close XML
```

The same happens with the "App and Service" Logs in the `GetApplicationAndServiceLogs` function. At the end all temporary files are deleted.

6.3 Module: Visualize

In this script the "UC05 - Display missing or wrong system configuration" is implemented, here the description how it was done.

6.3.1 Result

The script "UC05 - Display missing or wrong system configuration" runs as follows:

- Create Protable Document Format (PDF) at given folder and "open" it
- Import audit policies and compare them to a given checklist, result is written and visualized in a table
- Check which attack tool categories can be detected with the current audit guidelines and which cannot
- Import the found EventLogs and check if the important EventIDs, according to JPCERT/CC, are found
- "Close" PDF-document

The resulting PDF looks something like this:

AuditPolicies

Aduit Name	Target	Actual	Prio
AuditNonSensitivePrivilege	SuccessAndFailure	SuccessAndFailure	High
AuditProcessTermination	Success	SuccessAndFailure	Medium
AuditSAM	SuccessAndFailure	NotConfigured	Low
...	...	...	

With this policies it is possible to detect X out of 14 attack categories
The following attack categories cannot be detected with certainty:
...

WindowsLogs

EventID6	present
EventID104	missing
...	...

6.3.2 Approach

At first, the developers toyed with the idea to use "PowerBI" [28], like Jessica Payne uses it in "WEFFELS". But after a short tryout they decided that the tool was too overpowered for their purpose. Also did they not like that the user would have to install a third-party tool to analyse his data. The Dynamic Link Library (DLL) "iTextSharp", originally a C# library, allows to generate a PDF directly from the code, which can also be used in PowerShell. This variant is not very versatile and it is difficult to create an appealing design, but it is enough for now.

6.3.3 Implementation

This section describes the implementation of `Display missing or wrong system configuration` in detail. For this purpose, the following is referred to in the section "6.3.1 Result" described schedule.

The iTextSharp.dll and the functions from `PowerShell-PDF` [29] were imported. The first step is to create a PDF-document and "open" it. For this purpose the function `OpenPDF` was created:

Listing 2.21: Function OpenPDF

```
function OpenPDF{
   $pdf = New-Object iTextSharp.text.Document
   New-PDF -Document $pdf -File #export path
   $pdf.Open()
  }
```

The function `WriteAuditPolicies` then compares the found audit policies and display the ones who are incorrectly. It will call two other functions, `CreateAddCellWithColor` and `CreateAddCell`.

Listing 2.22: Functions WriteAuditPolicies & CreateAddCellWithColor & CreateAddCell

```
function WriteAuditPolicies{
   $auditChecklist = @{AuditLogon = @("Success", "Medium"; ...)}
   $incorrectAudits = @() # will be returned for later use
   [xml] auditXml = Get-Content $auditPath
   $myAudits = $auditXml.AuditPolicies.ChildNodes
   foreach ($audit in $myAudits) {
      $localName = $audit.LocalName
      CreateAddCell $localName # Display auditname into cell

      $checkaudit = $auditChecklist[$localName]
      $checkauditvalue = $checkaudit[0] # Correct setting
      $checkauditprio = $checkaudit[1] # Priority of audit

      if ($audit.InnerXml -eq $checkauditvalue) { # Checks if audit values are equal
         CreateAddCell $checkauditvalue # Displays correct audit value
         CreateAddCellWithColor $audit.InnerXml 0 255 0
         # Displays actual audit value into cell, color green
      }
      elseif ($audit.InnerXml.startswith("Succ") #checks if audit is ''overpowered''
      -and $checkauditvalue -eq "Success") {
         CreateAddCell $checkauditvalue # Displays correct audit value
         CreateAddCellWithColor $audit.InnerXml 0 106 0
         # Displays actual audit value into cell, color darkgreen
      }
      else { #audit is wrong
         CreateAddCell $checkauditvalue # Displays correct audit value
         CreateAddCellWithColor $audit.InnerXml 255 0 0
         #Displays actual audit value into cell, color red
         $incorrectAudits += $audit.LocalName
      }
      CreateAddCell $checkauditprio # Displays audit priority into cell
   }
   return $incorrectAudits
}
```

```
function CreateAddCellWithColor($content, $R, $G, $B) {
   # Create iTextSharp.text.Paragraph and add content
   # Create iTextSharp.text.pdf.PdfPCell with paragraph and set backgroundcolor $R $G $B
   # Add Cell to Table
   }

function CreateAddCell($content) {
      # Create iTextSharp.text.Paragraph and add content
      # Create iTextSharp.text.pdf.PdfPCell with paragraph
      # Add Cell to Table
}
```

Now that the "Import audit policies and compare them to a given checklist,result is written and visualized in a table" is done, it is possible to check which attack tool categories can be detected with the current audit settings. For that purpose the function `ToolsCanBeDetected` was created. This function relies on the return of the $incorrectAudits.

Listing 2.23: Function ToolsCanBeDetected

```
   function ToolsCanBeDetected($incorrectAudits){
      [xml] $auditsByCategorie = Get-Content "$PSScriptRoot\AuditByCategorie.xml"
      $notDetectableCategories = @()
      $causingAudit = @()

      $Categories = $auditsByCategorie.Category.ChildNodes
      foreach ($Category in $Categories) {
         [int]$checknr = 0
         foreach ($incorrectAudit in $incorrectAudits) {
            if ($Category.ChildNodes.InnerXml -contains $incorrectAudit) {
               $checknr += 1
               $causingAudit += $incorrectAudit
            }
         }

         if ($checknr -gt 0) {
            $notDetectableCategories += $Category.LocalName + "(" + $causingAudit + ")"
         }

      # Output of the not detectable categories and the causing audits
   }
```

The next step is to display the found EventLogs and if they are missing or present. Therefore two tables, one for the WindowsLogs and one for the Application And Service Logs, are created. Because this two tables are created the same way, only one case is shown. Therefore the function `WriteEventLogs` was created.

Listing 2.24: Function ToolsCanBeDetected

```
   function WriteEventLogs {
      [xml] $eventxml = Get-Content # importPath
      # Add Title
      $eventsWindows = $eventxml.Logs.EventLogsID.ChildNodes
      $result = @()
      foreach ($event in $eventsWindows) {
         $result += $event.LocalName
         $result += $event.InnerXml
      }
      #Add result to table
   }
```

As a final task, all these function have to be called in the right order, and the opened PDF has to be closed. For this case the simple function `VisualizeAll` was created:

Listing 2.25: Function VisualizeAll

```
1   function WriteEventLogs {
2       $pdf = OpenPDF $exportFolder
3       $incorrectAudits = WriteAuditPolicies $exportFolder
4       ToolCanBeDetected $incorrectAudits
5       WriteEventLogs $exportFolder
6       $pdf.Close()
7   }
```

6.4 Main Script: SRI

The aim of the main script is to supply the user with various procedures to evaluate the readiness of audit policies and/or event logs. However, the user is not meant to use single functions provided by the modules because most function provide just a metaset of data in order to make a statement of the readiness. For this reason, the main script is able to be called with various parameter for the carious procedures. Moreover, the user gets a help functionality (via PowerShells common `Get-Help`) to provide an overview of parameter combinations.

6.4.1 Result

The various procedures/modes provided by the main script , with eventual additional parameter, are

- ***Online, Offline, AllGroupPolicies and GroupPolicy***

```
PS C:\>./sri.ps1 [-Online] [-OnlineExportPath <String>] [-CAPI2LogSize <Int32>]

PS C:\>./sri.ps1 [-Offline] [[-AuditPolicies]] [[-EventLogs]] [-ImportPath] <String>
                 [[-ExportPath] <String>] [-CAPI2LogSize <Int32>]

PS C:\>./sri.ps1 [-GroupPolicy] [-GroupPolicyName] <String>

PS C:\>./sri.ps1 [-AllGroupPolicies]
```

Note: *Mandatory parameter are underlined.*

-Online

The current system which is calling the script will be checked on its readiness.

Parameter	
No parameter	The result PDF will be saved to the current path
-OnlineExportPath	The result PDF will be saved to this path
-CAPI2LogSize	Definition of the CAPI2 log size suitable for the environment. By default this value is set to 4MB as recommended from Microsoft [14]

-Offline

Some system will be checked on its readiness - by default audit policies and event log are analysed. Export files of this system are required.

Parameter	
-ImportPath	Defines where the required files rsop.xml[a], windowslogs.csv[b], appandservlogs.csv[c] remain for analysis. The result PDF will be saved to the current path
-AuditPolicies	Checks only the audit policies. The result PDF will be saved to the current path **-ImportPath** requires rsop.xml
-EventLogs	Checks only the event logs The result PDF will be saved to the current path **-ImportPath** requires windowslogs.csv and appandservlogs.csv
-ExportPath	The result PDF will be saved to this path
-CAPI2LogSize	Definition of the CAPI2 log size suitable for the environment. By default this value is set to 4MB as recommended from Microsoft [14]

[a] XML-Export of Resultant Set of Policy [19]
[b] Export of
[c] Export of

-GroupPolicy

Audit policies from a specific group policy are analysed.

Parameter	
-GroupPolicyName	The name of the group policy to be analysed

-AllGroupPolicies

All audit policies from every group policy in the current domain are analysed.
The result PDF will be saved to the current path

6.4.2 Approach

Users should not have to call the individual functions from the PowerShell modules. For this reason, the idea was to provide a main script which defines several modes to call with appropriate parameter. Each mode has a predefined procedure of function calls which will create a result PDF. In addition, the script should be delivered with a integrated help functionality to supply a on-demand overview of all possible script modes and its parameter.

6.4.3 Implementation

To get a better understanding how each mode proceeds, this section describes the source code in form of activity diagrams. The activity diagrams are an overview and do contain the core of each mode.

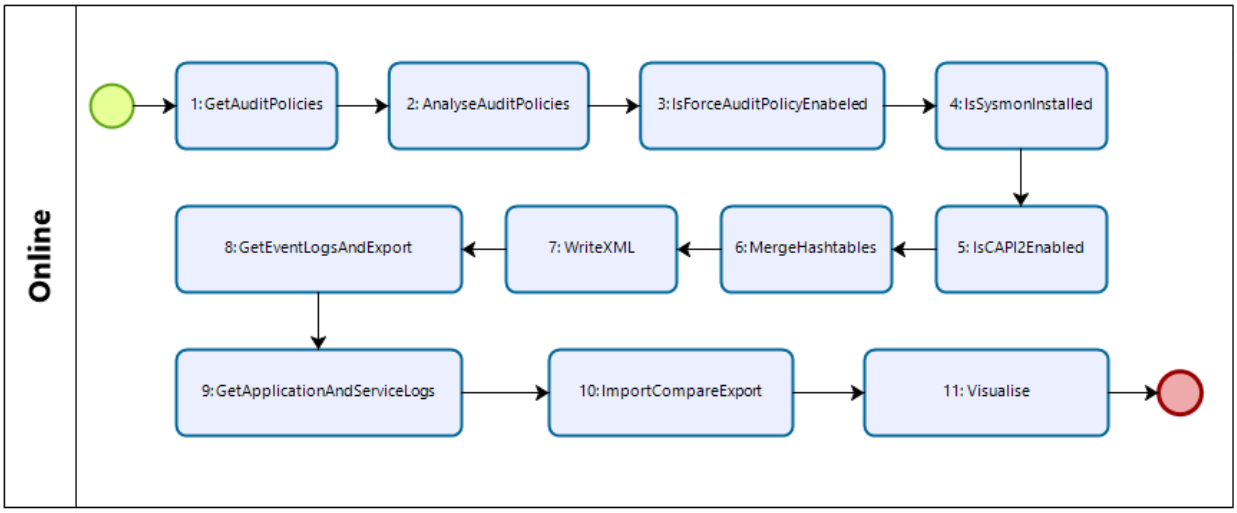

Figure 2.9: Mode: Online

1. Get the audit policies from RSoP
2. Analyse the audit policies
3. Get the registry value of the audit setting "Audit: Force audit policy subcategory settings (Windows Vista or later) to override audit policy category settings" and check if it is enabled
4. Check if sysmon is installed and running
5. Check if CAPI2 is enabled and has a minimum log size of 4MB
6. Merge all returned hashtables from step 2-5 to one hashtable
7. Write the "result_audit_policy.xml" for further processing to PDF
8. Get all events from "System" and "Security" event logs and write it uniquely to a temporary CSV
9. Get all events from "Application and Service" event log and write it uniquely to a temporary CSV
10. Compare the gathered events with the target lists of events (see "4.2 Mandatory Event Logs")
11. Create the result PDF

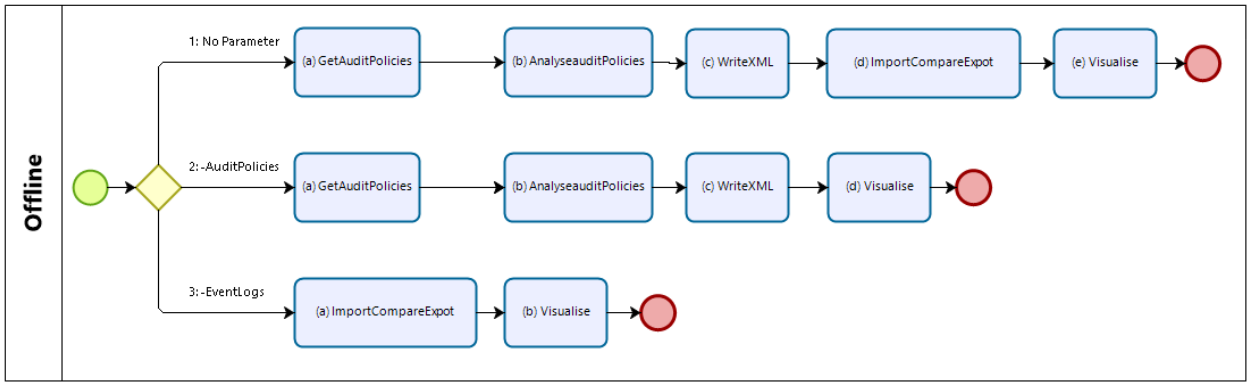

Figure 2.10: Mode: Offline

1. The offline mode without parameter will check the audit policies and event logs with the supplied export files[4]
   (a) Get the audit policies from the supplied rsop.xml
   (b) Analyse the audit policies
   (c) Write the "result_audit_policy.xml" for further processing to PDF
   (d) Compare the events form the supplied windowslogs.csv and appandservlogs.csv with the target lists of events
   (e) Create the result PDF
2. The offline mode with the parameter `-AuditPolicies` will check the audit policies with the supplied export file
   (a) Get the audit policies from the supplied rsop.xml
   (b) Analyse the audit policies
   (c) Write the "result_audit_policy.xml" for further processing to PDF
   (d) Create the result PDF
3. The offline mode with the parameter `-EventLogs` will check the event logs with the supplied files
   (a) Compare the events form the supplied windowslogs.csv and appandservlogs.csv with the target lists of events
   (b) Create the result PDF

[4] All export files required must remain at the **-ImportPath**

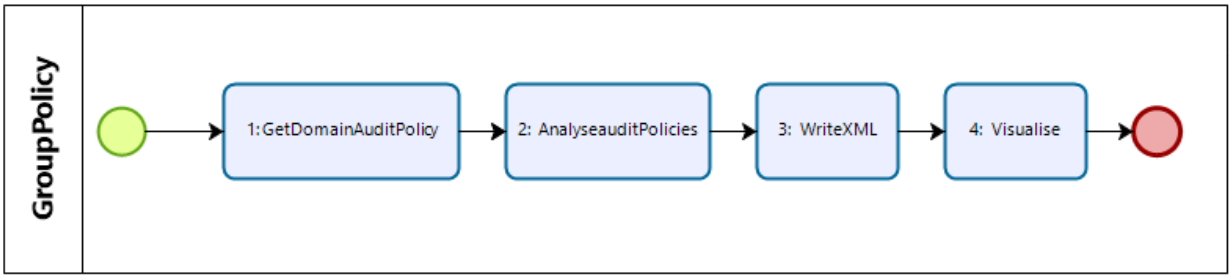

Figure 2.11: Mode: GroupPolicy

1. Get the audit policies from defined group policy which is stored in SYSVOL
2. Analyse the audit policies
3. Write the ”result_audit_policy.xml” for further processing to PDF
4. Create the result PDF and store it in the current path

7 Results

8 Conclusion

Glossary

ARP	Address Resolution Protocol (ARP) - for resolving MAC addresses, with an ARP table for mapping MAC and IP addresses

Listings

List of Figures

List of Tables

Bibliography

[1] harmj0y Andrew Robbins, Rohan Vazarkar. BloodHound - Wiki. https://github.com/BloodHoundAD/BloodHound/wiki, September 2018.

[2] Microsoft. Microsoft Security Complience Toolkit. https://docs.microsoft.com/en-us/windows/security/threat-protection/security-compliance-toolkit-10, June 2018.

[3] JPCERT/CC. LogonTracer. https://github.com/JPCERTCC/LogonTracer, September 2018.

[4] Shusei Tomonaga. Visualise Event Logs to Identify Compromised Accounts - LogonTracer -. https://blog.jpcert.or.jp/2017/11/visualise-event-logs-to-identify-compromised-accounts—logontracer-.html , November 2017.

[5] Microsoft. Monitoring Active Directory for Signs of Compromise | Microsoft Docs. https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/monitoring-active-directory-for-signs-of-compromise, May 2017.

[6] Microsoft. Appendix L: Events. https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/appendix-l–events-to-monitor, July 2018.

[7] JPCERT/CC. Detecting Lateral Movement through Tracking Event Logs. , June 2017.

[8] MITRE ATT&CK. MITRE ATT&CK Website. https://attack.mitre.org/, September 2018.

[9] Mark Russinovich, Thomas Garnier. Sysmon - System Monitor. https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon, July 2018.

[10] Carlos Perez. Operating Offensively Against Sysmon. https://www.darkoperator.com/blog/2018/10/5/operati offensively-against-sysmon, October 2018.

[11] Nader Shalabi. Sysmon Tools. https://github.com/olafhartong/sysmon-modular, October 2018.

[12] ipstack. Locate and identify website visitors by IP address. https://ipstack.com/, October 2018.

[13] Olaf Hartong. sysmon-modular. https://attack.mitre.org/, October 2018.

[14] Microsoft. Troubleshooting PKI Problems on Windows Vista - CAPI2 Diagnostics in Windows Vista. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-vista/cc749296(v=ws.10)#capi2-diagnostics-in-windows-vista, July 2008.

[15] Abe, Shingo. Detecting Lateral Movement in APTs - Analysis Approach on Windows Event Logs Introduction to JPCERT / CC. June 2016.

[16] Microsoft. Advanced security auditing FAQ. https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/advanced-security-auditing-faq, April 2017.

[17] Peter Morin. Extending Your Incident Response Capabilities with Sysmon. https://sector.ca/wp-content/uploads/presentations18/Morin_Sysmon_2019-16-9.pdf, September 2018.

[18] Daniel Scott-Raynsford. Converting a PowerShell Project to use Azure DevOps Pipelines. https://www.powershellmagazine.com/2018/09/20/converting-a-powershell-project-to-use-azure-devops-pipelines/, September 2018.

[19] Microsoft. RSoP - Resultant Set of Policy. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc772175(v=ws.11), February 2017.

[20] Microsoft. Get-EventLog. https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-eventlog?view=powershell-5.1, October 2018.

[21] Microsoft. auditpol. https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/auditpol, September 2017.

[22] Microsoft. Get-GPResultantSetOfPolicy. https://docs.microsoft.com/en-us/powershell/module/grouppolicy/get-gpresultantsetofpolicy?view=win10-ps, October 2018.

[23] Microsoft. GroupPolicy. https://docs.microsoft.com/en-us/powershell/module/grouppolicy/?view=win10-ps, October 2018.

[24] Microsoft. Remote Server Administration Tools for Windows 10. https://www.microsoft.com/en-us/download/details.aspx?id=45520, October 2018.

[25] Microsoft. Audit: Force audit policy subcategory settings (Windows Vista or later) to override audit policy category settings. https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/audit-force-audit-policy-subcategory-settings-to-override, April 2017.

[26] Microsoft. Get-WinEvent. https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-winevent?view=powershell-6, October 2018.

[27] Microsoft. wevtutil. https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/wevtutil, October 2017.

[28] Microsoft. PowerBI. https://powerbi.microsoft.com, November 2018.

[29] Patrick Lambert. PowerShell-PDF. https://github.com/dendory/PowerShell-PDF, March 2015.

Part II

Appendix

Task Definition

Einführung

Es werden vermehrt Cyberangriffe publik, wo Schadcode im Einsatz ist, welcher sich nicht nur auf einem infizierten System niederlässt, sondern weitere Systeme im Netz befällt. Das Ziel oder Resultat ist dabei oft die komplette Infiltrierung einer Organisation. In der Analyse solcher Fälle sind Information und Zeit ein Schlüssel zum Erfolg. Folglich ist die Bereitschaft "Readiness" für ein solches Ereignis ein entscheidender Faktor.

Aufgabe

Ziel dieser Arbeit ist es, ein Tool zu erstellen, welches die Bewertung der eigene Readiness erlaubt aber auch im Analysefall eine Unterstützung bietet. Readiness betrifft viele Aspekte und einfache Dinge wie korrekte Zeitstempel in Logs, deren Vollständigkeit oder die Bereitstellung von Backups. In der konkreten Aufgabenstellung soll die Readiness-Analyse primär für Windows-Infrastrukturen anhand von Logs und spezifischen Events erfolgen. Unter anderem soll auf den neusten Publikationen des japanischen Computer Emergency Response Teams (JPCERT/CC) und der öffentlichen Datenbank der MITRE Corporation, dem Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK^{TM}) Wissenspool, basiert werden. Das JPCERT und MITRE haben dabei die Werkzeuge und das generelle Vorgehen von Angreifern analysiert und geben Hinweise, welche Events auf eine mögliche Verseuchung hinweisen.

Abgrenzung

Es geht nicht darum neue Angriffsvektoren zu finden.

Tätigkeiten

- Projektmanagement und Dokumentation
- Einarbeitung in Incident Handling und Forensik
- Einarbeitung in Angriffstechniken und Werkzeuge
- Einarbeitung in Abwehrtechniken und Härtung von Systemen
- Studium öffentlicher Quellen und verfügbaren Tools
- Umsetzung eines Analyzers gemäss Anforderungen basierend auf etablierten Frameworks

Vorgehen

Im Rahmen der allgemeinen Richtlinien zur Durchführung von Studien- und Bachelorarbeiten gemäss eigenem Projektmanagementplan. Dieser Projektmanagementplan ist als Erstes zu erstellen und enthält insbesondere:

- Die Beschreibung des dem Projektcharakter angepassten Vorgehensmodells.
- Eine erste Aufteilung der Aufgabe in gemeinsam und einzeln zu bearbeitende Teile unter Berücksichtigung der vorgegebenen Teilaspekte. Die genaue Aufteilung muss spätestens nach der Technologiestudie (Elaboration) erfolgen.
- Den Projektplan (Zeitplan) und die Meilensteine.

Anforderungen

Es geht primär darum einen Analyzer zu erstellen um die "Readiness for Tailored Attacks and Lateral Movement Detection" beurteilen zu können. Idealerweise kann dieses Tool von einem IT Administrator ohne spezielle Kenntnisse und grossartige Installationsprozeder ausgeführt werden.

Schematisch aber nicht bindend werden folgende Schritte auszuführen sein

- Definition der Requirements für einen neuen/verbesserten Analyzer
- Design und Analyse basierend auf den Vorgaben
- Vorschläge für die Umsetzung oder Verbesserung eines
  - Readiness Analyzers
  - Readiness Optimizers
  - Compromise Analyzers
- Implementation der Funktionalität und Erstellung eines Benutzerhandbuch
- Erweiterung der Analyzer um neue Erkenntnisse, Werkzeuge und Indicators
- Dokumentation der Software und Skripte

Technologien

- Windows Workstations, Windows Server, Windows Security generell
- Windows Event Logs, Security und Audit Logs
- Windows On-Board Tools, Sysinternals Toolkit
- Active Directory Service (AD) Services
- Group Policy Objects (GPO)
- PowerShell, .NET, Python, Windows Batch

Infrastruktur

Die Arbeiten werden auf den Rechnern der Studenten durchgeführt. Zusätzlich benötigte Software oder Hardware wird bei Bedarf und nach Rücksprache mit Compass Security zur Verfügung gestellt.

Erwartete Resultate

In elektronischer Form

- lauffähiges Toolkit und kompletter Source Code
- komplette Software Dokumentation (Use Cases, Klassenmodell, Sequenzdiagramme usw. in UML)
- komplette Use Cases und Erfolgs-Szenarien resp. Musterlösungen
- alle Dokumente und Protokolle (vorzugsweise in englischer Sprache)

Auf Papier

Gemäss der Anleitung der HSR: \\hsr.ch\root\alg\skripte\Informatik\Fachbereich\Studienarbeit_Informatik Es muss aus den abgegebenen Dokumenten klar hervorgehen, wer für welchen Teil der Arbeit und der Dokumentation verantwortlich war (detaillierte Zeiterfassung).

Termine

Termine gemäss der HSR: \\hsr.ch\root\alg\skripte\Informatik\Fachbereich\Studienarbeit_Informatik\SAI\Termine

Datum	Task
17.09.2018	Beginn der Arbeit, Ausgabe der Aufgabenstellung durch den Betreuer.
18.12.2018	Erfassung des Abstracts im Online-Tool https://abstract.hsr.ch/ Die Studierenden geben den Abstract für die Diplomarbeitsbroschüre zur Kontrolle an ihren Betreuer/Examinator frei. Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract zur Weiterverarbeitung an das Studiengangsekretariat frei Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen auf dem Skripteserver zur Verfügung.
21.12.2018	Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract der Broschüre zur Weiterverarbeitung an das Studiengangsekretariat frei. Hochladen aller verlangten Dokumente auf archiv-i.hsr.ch Abgabe des Berichts an den Betreuer bis 12.00 Uhr

Zeitplan und Meilensteine

Zeitplan und Meilensteine für das Projekt sind von den Studenten selber zu erarbeiten und zusammen mit dem Projektmanagementplan abzuliefern. Die Meilensteine sind bindend. Der erste Meilenstein ist vorgegeben. Mit den Betreuern werden regelmässige Sitzungen zur Fortschrittskontrolle durchgeführt.

Betreuung

Die Arbeiten werden durch Cyrill Brunschwiler betreut. Der Gegenleser ist noch nicht bestimmt.

Kontakt

Cyrill Brunschwiler, Managing Director, Compass Security Schweiz AG
Weststrasse 50, 8003 Zürich, Switzerland
Werkstrasse 20, 8645 Jona, Switzerland

+41 55 214 41 73
cyrill.brunschwiler@compass-security.com
cyrill.brunschwiler@hsr.ch
https://fb.compass-security.com/inbox/hUGXMr2EeZ2V7b

Unterschriften

Jona, 28. September 2018

Cyrill Brunschwiler

Claudio Mattes

Lukas Kellenberger