Signature Host Windows

2IMS40 - Intrusion Detection Laboratory

Group 04

Full Name	**Student ID**
Tudor Dascălu	1546260
Kenna Janssens	1577271
Sander van der Leek	1564226
Chengqi Liu	1954148
Egor Larionov	1562983

Eindhoven, November 21, 2025

Contents

1 | Introduction

In today's highly dependent world where everything heavily relies on technology, there unfortunately exist many malicious actors that would like to use our communication methods in ways that negatively affect other users. Cybercrime is a prevalent topic and new attacks are discovered all the time. It is a never-ending race between the attacker coming up with new innovative ways to compromise a system and the defender tightening their security to prevent any intrusions. One way to do this is through the use of signatures. As such, it is important to be able to develop signatures that can periodically generate alerts of possible intrusion in a system.

A signature is some sort of unique pattern that can be matched against some malicious activity to detect it. Signature-based intrusion detection is very powerful since it is a well-established method and can quickly identify known malicious behaviour. However, this method is ineffective against unknown malicious behaviour which would not get flagged.

The attack investigation is performed on data from a virtualized Windows 11 host where a malware infection has occurred. The data consists of Windows Event Logs [1] and Sysmon logs [2]. These logs contain data about process creation, command executions, registry changes, file creation, etc. Such logs allow the identification of a malware infection by following the typical steps in a malware's life cycle.

This report consists of the following sections. Section 2 explores the malware infection that took place. It investigates the different steps the malware takes and details the indicators of compromise that were identified. Section 3 discusses the signatures that can be used to identify this particular sort of malware infection. This section also evaluates the proposed signatures both empirically and theoretically. Section 4 discusses possible mitigation strategies to prevent such malware infections in the future and how users can protect themselves from malware infections. Section 5 describes how the lab session will be carried out such that our colleagues can also quickly identify the intrusion and develop basic signatures for it. Next, Section 6 goes into detail about how the overall investigation went and what could be improved or done differently next time. Finally, Section 7 summarizes the outcome of the investigation and the developed signatures. This section also brings the bigger picture into perspective.

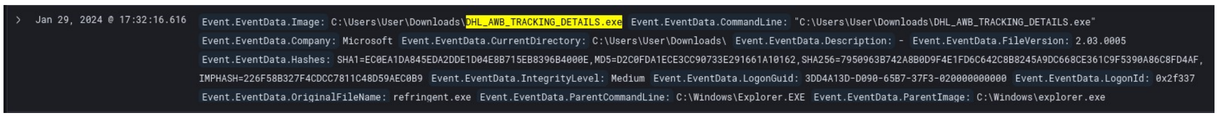
2 | Attack Investigation

The first step taken in the attack investigation was exploring the dataset provided. The dataset consisting of over 3 million Windows Event and Sysmon Logs, was combed through to find suspicious activity using Elastic Search. Since the machine was assumed to have a malware infection, the starting point for searching the malicious activity was looking for some process creations (EventID 1) or for any command run in the Windows terminal that could have started an executable file. Moreover, to increase the chances of finding any suspicious activity, we also looked for file creation events (EventID 11) that appeared after file execution. Quite some trial and error was necessary for trying different queries to reduce the search area, for example, only looking at Sysmon process creation events. Finally, a query was created that filtered out all logs coming from images that include `system32`, `SystemApps`, or `Program Files` in their paths and only selected Sysmon process creation events from `.exe` images (see Figure 2.1).

```
Event.System.EventID:1 and not Event.EventData.Image: system32
and not Event.EventData.Image: SystemApps
and not Event.EventData.Image: Program Files
and Event.EventData.Image : *.exe
```

Figure 2.1: Query used to initially find the attack

Executing this query resulted in 296 logs, which was more convenient to scan through. A quick scan through these logs revealed an entry from a process creation event with a suspicious Image path (see Figure 2.2). Investigating further by checking the SHA1 hash (see Table A.1) on VirusTotal [3], confirmed the suspicion. VirusTotal reports that this file was identified as malicious by 60 out of 73 security vendors. It also reports the malicious file as a trojan and a member of the DarkCloud malware family. The `DHL_AWB_TRACKING_DETAILS.exe` file is also identified as a malware dropper, a file that installs other malware files on the victim's machine.



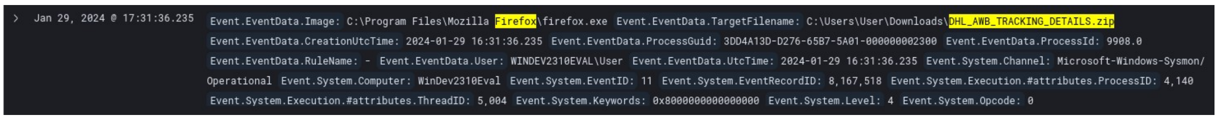
```
> Jan 29, 2024 @ 17:32:16.616  Event.EventData.Image: C:\Users\User\Downloads\DHL_AWB_TRACKING_DETAILS.exe Event.EventData.CommandLine: "C:\Users\User\Downloads\DHL_AWB_TRACKING_DETAILS.exe"
                               Event.EventData.Company: Microsoft Event.EventData.CurrentDirectory: C:\Users\User\Downloads\ Event.EventData.Description: - Event.EventData.FileVersion: 2.03.0005
                               Event.EventData.Hashes: SHA1=EC0EA1DA845EDA2DDE1D04E8B715EB8396B4000E,MD5=D2C0FDA1ECE3CC90733E291661A10162,SHA256=7950963B742A8B0D9F4E1FD6C642C8B8245A9DC668CE361C9F5390A86C8FD4AF,
                               IMPHASH=226F58B327F4CDCC7811C48D59AEC0B9 Event.EventData.IntegrityLevel: Medium Event.EventData.LogonGuid: 3DD4A13D-D090-65B7-37F3-020000000000 Event.EventData.LogonId: 0x2f337
                               Event.EventData.OriginalFileName: refringent.exe Event.EventData.ParentCommandLine: C:\Windows\Explorer.EXE Event.EventData.ParentImage: C:\Windows\explorer.exe
```

Figure 2.2: Suspicious log found by exploring process creation events

The malware of this attack is identified as a stealer malware. The objective of the malware is to collect sensitive data from the machine it has infiltrated. The sensitive data it steals can be user credentials or financial information extracted from web data, like bank account details or credit card information. The stolen data will then be communicated to the Command and Control Centre [4], [5]. The stolen data may consequently be used to launch other attacks, like ransomware attacks, identity theft, or credit card fraud, and the data may be sold to other malicious parties.

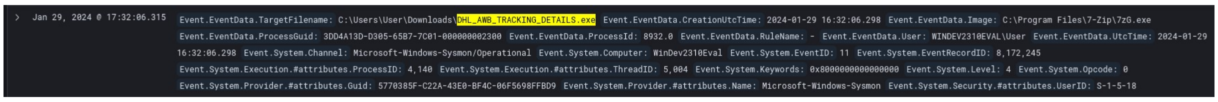
2.1 | Incident Description

To begin with, the malicious file that was found is called `DHL_AWB_TRACKING_DETAILS.exe` and it is the parent image for all the subsequent events that happened on the host machine. By filtering for this file, one could explore the dataset to find how the attack was executed and what happened during the infection. To find the initial step of the attack, a search was performed for the events with the malicious file mentioned above as the target file. By using this query, it was found that the first step in the attack was to download the `DHL_AWB_TRACKING_DETAILS.zip` file from `Firefox` and unzip it with `7zip` into the `Program Files` folder (see Figures 2.3 and 2.4).



```
> Jan 29, 2024 @ 17:31:36.235  Event.EventData.Image: C:\Program Files\Mozilla Firefox\firefox.exe Event.EventData.TargetFilename: C:\Users\User\Downloads\DHL_AWB_TRACKING_DETAILS.zip
                               Event.EventData.CreationUtcTime: 2024-01-29 16:31:36.235 Event.EventData.ProcessGuid: 3DD4A13D-D276-65B7-5A01-000000002300 Event.EventData.ProcessId: 9908.0
                               Event.EventData.RuleName: - Event.EventData.User: WINDEV2310EVAL\User Event.EventData.UtcTime: 2024-01-29 16:31:36.235 Event.System.Channel: Microsoft-Windows-Sysmon/
                               Operational Event.System.Computer: WinDev2310Eval Event.System.EventID: 11 Event.System.EventRecordID: 8,167,518 Event.System.Execution.#attributes.ProcessID: 4,140
                               Event.System.Execution.#attributes.ThreadID: 5,004 Event.System.Keywords: 0x8000000000000000 Event.System.Level: 4 Event.System.Opcode: 0
```

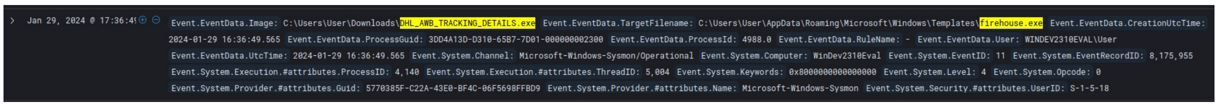
Figure 2.3: Log that shows the download of the malicious file `DHL_AWB_TRACKING_DETAILS.exe` from Firefox



```
> Jan 29, 2024 @ 17:32:06.315  Event.EventData.TargetFilename: C:\Users\User\Downloads\DHL_AWB_TRACKING_DETAILS.exe Event.EventData.CreationUtcTime: 2024-01-29 16:32:06.298 Event.EventData.Image: C:\Program Files\7-Zip\7zG.exe
                               Event.EventData.ProcessGuid: 3DD4A13D-D305-65B7-7C01-000000002300 Event.EventData.ProcessId: 8932.0 Event.EventData.RuleName: - Event.EventData.User: WINDEV2310EVAL\User Event.EventData.UtcTime: 2024-01-29
                               16:32:06.298 Event.System.Channel: Microsoft-Windows-Sysmon/Operational Event.System.Computer: WinDev2310Eval Event.System.EventID: 11 Event.System.EventRecordID: 8,172,245
                               Event.System.Execution.#attributes.ProcessID: 4,140 Event.System.Execution.#attributes.ThreadID: 5,004 Event.System.Keywords: 0x8000000000000000 Event.System.Level: 4 Event.System.Opcode: 0
                               Event.System.Provider.#attributes.Guid: 5770385F-C22A-43E0-BF4C-06F5698FFBD9 Event.System.Provider.#attributes.Name: Microsoft-Windows-Sysmon Event.System.Security.#attributes.UserID: S-1-5-18
```

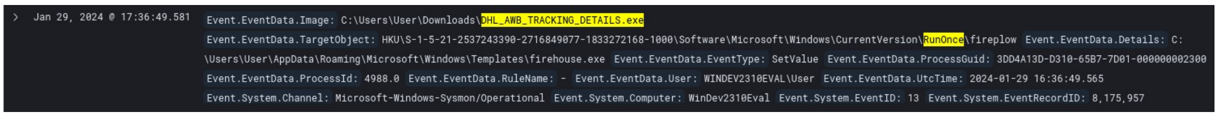
Figure 2.4: Log that shows the unzipping of the malicious file `DHL_AWB_TRACKING_DETAILS.exe`

Afterwards, the malware is executed by a simple command line execution (Mitre T1204) and creates another file called `firehouse.exe` to establish a method for persistence and privilege escalation on the host machine (see Figure 2.5). For persistence, the new file is added to the "RunOnce" in the registry (see Figure 2.6) causing this file to be executed whenever a user logs in, meaning the file will be executed under the context of the current user and will have the permissions level associated with this account (Mitre T1547.001).



```
> Jan 29, 2024 @ 17:36:4  Event.EventData.Image: C:\Users\User\Downloads\DHL_AWB_TRACKING_DETAILS.exe Event.EventData.TargetFilename: C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe Event.EventData.CreationUtcTime:
                          2024-01-29 16:36:49.565 Event.EventData.ProcessGuid: 3DD4A13D-D310-65B7-7D01-000000002300 Event.EventData.ProcessId: 4988.0 Event.EventData.RuleName: - Event.EventData.User: WINDEV2310EVAL\User
                          Event.EventData.UtcTime: 2024-01-29 16:36:49.565 Event.System.Channel: Microsoft-Windows-Sysmon/Operational Event.System.Computer: WinDev2310Eval Event.System.EventID: 11 Event.System.EventRecordID: 8,175,955
                          Event.System.Execution.#attributes.ProcessID: 4,140 Event.System.Execution.#attributes.ThreadID: 5,004 Event.System.Keywords: 0x8000000000000000 Event.System.Level: 4 Event.System.Opcode: 0
                          Event.System.Provider.#attributes.Guid: 5770385F-C22A-43E0-BF4C-06F5698FFBD9 Event.System.Provider.#attributes.Name: Microsoft-Windows-Sysmon Event.System.Security.#attributes.UserID: S-1-5-18
```

Figure 2.5: Log that shows the file creation of `firehouse.exe`

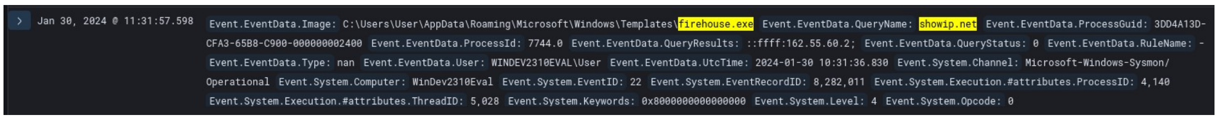


```
> Jan 29, 2024 @ 17:36:49.581  Event.EventData.Image: C:\Users\User\Downloads\DHL_AWB_TRACKING_DETAILS.exe
                               Event.EventData.TargetObject: HKU\S-1-5-21-2537243390-2716849077-1833272168-1000\Software\Microsoft\Windows\CurrentVersion\RunOnce\fireplow Event.EventData.Details: C:
                               \Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe Event.EventData.EventType: SetValue Event.EventData.ProcessGuid: 3DD4A13D-D310-65B7-7D01-000000002300
                               Event.EventData.ProcessId: 4988.0 Event.EventData.RuleName: - Event.EventData.User: WINDEV2310EVAL\User Event.EventData.UtcTime: 2024-01-29 16:36:49.565
                               Event.System.Channel: Microsoft-Windows-Sysmon/Operational Event.System.Computer: WinDev2310Eval Event.System.EventID: 13 Event.System.EventRecordID: 8,175,957
```

Figure 2.6: Log that shows the persistence tactic of adding `firehouse.exe` to the "RunOnce" registry, causing this file to be executed the next time the user logs on

The main goal of the malware is to harvest credentials and send them to a Command and Control (C&C) server. Thus, the harvested information is stored in multiple files which are sent to the malicious IP address. Some examples of such files are the `WINDEV2310EVAL-User.txt` and `cookies.db` which are created by `firehouse.exe` (see Table A.3) and are used to collect data before exfiltration (according to Mitre T1074, there are 2 known procedure examples for local data staging which are similar to the logs from our database: S1029 AuTo Stealer for the `.txt` file and S0261 Catchamas for the `.db` file).

Another suspicious action executed by `firehouse.exe` was identified in the attack. A DNS query for `showip.net` occurs that was initiated by the malware. This is a website that allows a user to find their IP address. Here the malware finds the victim's IP address (see Figure 2.7).

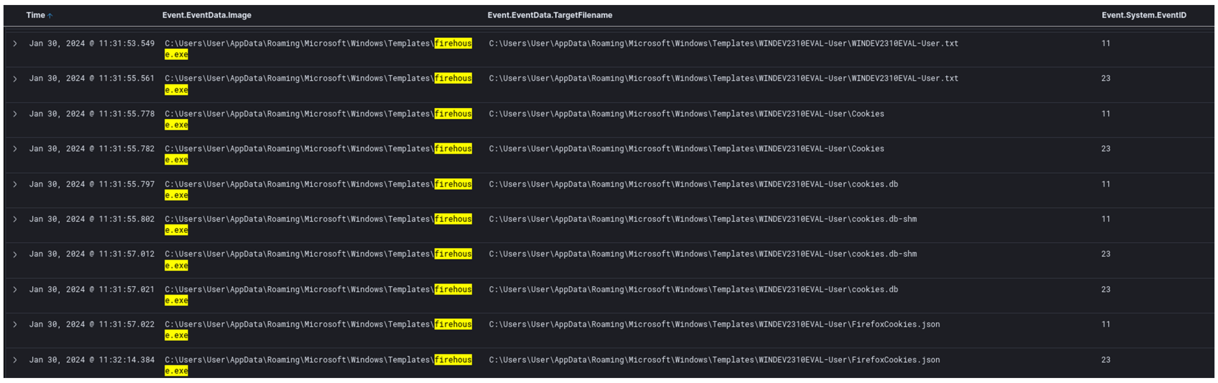


```
> Jan 30, 2024 @ 11:31:57.598  Event.EventData.Image: C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe Event.EventData.QueryName: showip.net Event.EventData.ProcessGuid: 3DD4A13D-
                               CFA3-65B8-C900-000000002400 Event.EventData.ProcessId: 7744.0 Event.EventData.QueryResults: ::ffff:162.55.60.2; Event.EventData.QueryStatus: 0 Event.EventData.RuleName: -
                               Event.EventData.Type: nan Event.EventData.User: WINDEV2310EVAL\User Event.EventData.UtcTime: 2024-01-30 10:31:36.830 Event.System.Channel: Microsoft-Windows-Sysmon/
                               Operational Event.System.Computer: WinDev2310Eval Event.System.EventID: 22 Event.System.EventRecordID: 8,282,011 Event.System.Execution.#attributes.ProcessID: 4,140
                               Event.System.Execution.#attributes.ThreadID: 5,028 Event.System.Keywords: 0x8000000000000000 Event.System.Level: 4 Event.System.Opcode: 0
```

Figure 2.7: Log that shows that the `firehouse.exe` malware contacts `showip.net` to find the victim's IP address.

For the Command and Control part of the attack, the malware simply uses the application layer protocol to avoid detection by blending in with the existing traffic (Mitre T1071). Hence, the last step, namely the data exfiltration, is done over the C&C channel as the stolen data is sent to the attacker's IP address using the same communication protocol as the protocol used to communicate with the Command and Control server (see Table A.2).

Another tactic for preventing detection that is used by the malware is trying to cover its tracks. All the files that were created during the attack to store the stolen data (see A.3) are deleted immediately afterwards (see Figure 2.8). This has the purpose of minimizing the attack's footprint and leaving traces (Mitre T1070.004).



Time ↑	Event.EventData.Image	Event.EventData.TargetFilename	Event.System.EventID
Jan 30, 2024 @ 11:31:53.549	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\WINDEV2310EVAL-User.txt	11
Jan 30, 2024 @ 11:31:55.561	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\WINDEV2310EVAL-User.txt	23
Jan 30, 2024 @ 11:31:55.778	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\Cookies	11
Jan 30, 2024 @ 11:31:55.782	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\Cookies	23
Jan 30, 2024 @ 11:31:55.797	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\cookies.db	11
Jan 30, 2024 @ 11:31:55.802	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\cookies.db-shm	11
Jan 30, 2024 @ 11:31:57.012	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\cookies.db-shm	23
Jan 30, 2024 @ 11:31:57.021	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\cookies.db	23
Jan 30, 2024 @ 11:31:57.022	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\FirefoxCookies.json	11
Jan 30, 2024 @ 11:32:14.384	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe	C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\WINDEV2310EVAL-User\FirefoxCookies.json	23

Figure 2.8: Logs showing that the files created during the attack to store stolen data are all deleted to prevent leaving traces

2.2 | Attack Mechanics

In this section, the specific stages of the attack will be discussed further continuing on the observed actions discussed previously. The core characteristics of the attack are listed below, including the corresponding TTPs of each step of the attack.

Steps of the attack:

■ Initial Access	(TA0001, *T1566*)
■ Execution	(TA0002, T1204.002, T1053)
■ Persistence	(TA0003, T1547.001)
■ Privilege Escalation	(TA0004, T1547.001)

■ Defense Evasion	(TA0005, T1070.004)
■ Credential Access	(TA0006, T1003.002)
■ Discovery	(TA0007, T1087)
■ Collection of Data	(T1005)
■ Command and Control/Exfiltration	(TA0011, TA0010, T1041)

2.2.1 | Initial Access

The Initial Access stage is where the adversary tries to gain access to the victim's machine and/or network. For this attack, the attacker tries to plant the malware onto the victim's machine. For the incident that was identified in our data set, the victim's computer was infected with the malware by a download from Firefox. The malware was located in a `.zip` file that was downloaded and subsequently unzipped into the Program Files folder as seen in Section 2.1. However, another common method that is often deployed for the initial infection is phishing (Mitre T1566). The attacker sends a seemingly legitimate email to the victim with the malicious file in the attachments. The victim downloads the attachment and executes it on their system, completing the infection.
This method for initial access is not unique to this particular attack. And there are many other methods for infecting victim machines, for example, content injection or drive-by compromise where particular weaknesses in websites are exploited such that malicious content can be downloaded onto the victim's system.
Downloading `.zip` files or even directly downloading `.exe` files is very common benign behaviour, as this is often a step for installing applications. Therefore, finding logs of downloading `.zip` or `.exe` files does not indicate suspicious behaviour. It is thus also difficult to detect the initial access step in an automated way. Especially, since the `DHL_AWB_TRACKING_DETAILS.exe` that was unzipped is not the main malware, it is only a dropper that in a later stage of the attack drops the actual malware file. The unzipped file, in this case `DHL_AWB_TRACKING_DETAILS.exe`, can have different names for every new attack incident.
Phishing emails could maybe be detected automatically, for example, by analyzing email content, the reputation of the sender, and header analysis to detect spoofing and flagging suspicious attachments. But this is also challenging as it is quite normal to get emails with `.zip` attachments.
However, it would be possible to check the file hashes against a list of known malicious hashes, for example by using the VirusTotal API.

2.2.2 | Execution

The execution phase is the stage that results in the malicious code running on the victim's system. In the analyzed incident, the first execution occurs by running the `DHL_AWB_TRACKING_DETAILS.exe` executable from the command line. This is user behaviour, the user chooses to execute the malicious file (Mitre T1204.002). The attacker has to rely on the user to open/execute the malicious file.
The execution of `DHL_AWB_TRACKING_DETAILS.exe` includes two important activities. Firstly, the malicious executable drops the actual malware file. It creates a new file (EventID 11) with the name `firehouse.exe` which contains the actual malicious code. Secondly, it arranges the execution of this malicious dropped file using a version of the Scheduled Task/Job technique (Mitre T1053). With this technique, the malicious `firehouse.exe` is executed automatically at a certain time, more on this in section 2.2.3.
The `firehouse.exe` file contains the malicious code that allows the attacker to steal data from the victim's machine (see section 2.2.8). For this stage of the attack, an automated technique to detect the dropper functionality of the malware is possible. This method would involve checking all the file creation events executable files. This may not be effective as this can be benign behaviour, but it could be effective if the dropper file had been identified already. For example, by knowing that the dropper is `DHL_AWB_TRACKING_DETAILS.exe`, a signature can be created that flags all `.exe` files that were created by this dropper.

2.2.3 | Persistence

Persistence techniques are methods deployed by the attacker to maintain access to the victim's machine or to make the attack survive, for example after the victim restarts their computer or logs off. For this attack, the attacker makes use of the technique Registry Run Keys (Mitre T1547.001). As described above, the original dropper file `DHL_AWB_TRACKING_DETAILS.exe` enables the execution of the `firehouse.exe` file. It does this by adding the program to the Run Keys registry (under the key `fireplow`). For this, the following registry value is set (EventID 12):

```
HKU\S-1-5-21-2537243390-2716849077-1833272168-1000\Software\
↪  Microsoft\Windows\CurrentVersion\RunOnce\fireplow
```

and it is set (EventID 13) to:

```
 C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\firehouse.exe
```

The `RunOnce` key ensures that when the user logs on, the program `firehouse.exe` is executed and afterwards the key is deleted [6]. This ensures that even if the system reboots, the `firehouse.exe` program will still be executed the next time the user logs on.
A signature can be developed that checks for new Run Keys being created. However, the creation of new Run Keys can also be part of benign behaviour. For example, installing new applications can also include adding those programs to the Run Keys registry to make sure the applications start executing when the user logs on.
An alternative technique for the attacker, instead of using Run Keys, is to make use of the DLL side-loading technique (Mitre T1574.002). With this method, the adversary can execute their own malicious payload by hijacking which DLL is loaded by certain legitimate programs. This is a persistent method and also well hidden as it makes use of legitimate and trusted programs.

2.2.4 | Privilege Escalation

The technique of using Registry Run Keys (Mitre T1547.001) can also be used by the attack to elevate their permissions on the system. As the `firehouse.exe` is added to the RunOnce registry, the program will be executed under the context of the current user. Therefore, it will have the permissions level that is associated with the account of that user.

2.2.5 | Defense Evasion

It is key for the attack to remain undetected while in progress, to avoid being stopped before the main goal is achieved. One particular defense evasion technique was identified in this attack, namely Indicator Removal (Mitre T1070). Specifically, the File Deletion technique (Mitre 1070.004) is used, as discussed in section 2.1. The files that were created by `firehouse.exe` that store the stolen data (see 2.2.8) are all deleted (EventID 23). This makes sure that the attack's footprint is minimized and as few traces as possible are left that could indicate that an attack took place.
File deletions are of course very common and do not indicate suspicious behaviour. However, it is remarkable that most of the files created by `firehouse.exe` are deleted within 2 minutes. This could be the basis for an automated detection method, for example, one that flags images that create several new files (EventID 11) and delete these same files (EventID 23) in a short period.

2.2.6 | Credential Access

For this mechanic, the attacker used the technique of OS Credential Dumping (Mitre T1003) targeting the Security Account Manager (SAM) to gain access to credentials. It can be seen in multiple logs that the malware tried to request a handle for an object from SAM (EventID 4656) or to access the object directly (EventID 4663). Besides the previous, the malware also tries to harvest credentials by accessing the available data from the machine's browser. This can be seen as `firehouse.exe` accesses (opens or overwrites them, EventID 11) the `FirefoxCookies.json` and `WebData` files multiple times. This is a key step in the attack since this is the moment when the malware steals the information from the victim's machine.
Despite the previous, the attacker could have used another technique for gaining access to the credentials such as attempting to create a copy of the Active Directory domain database. In this way, the attacker could have obtained information about the credentials as well as other data about domain members such as users, access rights, and devices (Mitre T1003.003).
Checking for this attack mechanism in an automated way would imply knowing all the applications that can legitimately access SAM or the web data stored on the machine. Hence, there could be an efficient signature for this step only if it checks the name of the process that tries to access the SAM/web data (which in this case would be `firehouse.exe`), resulting in a signature that would not work on a more general case.

2.2.7 | Discovery

Two techniques are used for discovery: permission group discovery (Mitre T1069) and query registry (Mitre T1012). The former technique determines which user accounts and groups are available along with what permissions each account has. This information is helpful because the malware can find the membership of users in particular groups such that it discovers accounts/groups that have elevated permissions. On the other hand, the query registry technique can help the attacker gather information about the system and its configuration. The Registry contains data about the installed software, operating system, and its security. A more elevated approach for this stage of the attack would have been the usage of the Security Software Discovery technique (Mitre T1518.001) in which the malware could have discovered the firewall rules and anti-virus of the infected machine and deactivated them.
For this stage of the attack, an automated way of checking for the attack mechanism would be to verify all the actions related to user accounts and groups (for example to search EventIDs such as 4720/4722/4725/4726 for user account management or 4731/4727/4754 for security group management).

2.2.8 | Collection of Data

In this part of the attack, the malware makes use of the technique of collecting data from the local system (Mitre T1005). The malware searches the file systems and configuration files or local databases, to find sensitive information that could be of interest for exfiltration. During this step, the malware targets files from which it can harvest credentials as explained in the Credential Access stage. For this stage, other similar techniques could have been used, such as Archive Collected Data (Mitre T1560) in which the attacker also compresses/encrypts data before exfiltration or Input Capture to collect keyboard and mouse events.

2.2.9 | Command and Control/Exfiltration

The last two steps are combined since there is no incoming traffic from the actual command and control server, but there are outgoing network connections from the machine to the malicious domain (see Table A.2). As the timestamps of the logs indicate that all connections are being done at the end of the attack, it could mean that these connections are used for exfiltration rather than for command and control. In this step, the stolen data is encoded into the normal communications channel using IPv4 such that the network connections are hidden in the normal traffic (Mitre T1041). For this stage, there can be signatures created that verify the IP address of the malicious domain (resulting in a very specific signature) as well as check for connections that are not IPv6 secure (but in this way the probability of false positives increases).

3 | Detection Method Development and Evaluation

The signature detection methods are derived by exploring the dataset for interesting or suspicious events surrounding our found malware. Then for each step in the attack as discussed in section 2.2 we develop a signature Sigma rule that detects every such log entry.
The philosophy behind the signatures is that they would be checked on a real system along with many other signatures each designed to detect other types of attacks. Therefore, the following signatures are specific enough to always trigger an alert if this specific attack step is detected but reduce the chance of a false positive alert.
Then if multiple (or all) of these signatures are matched successfully in a log a system administrator would be able to determine with high confidence that this malware has indeed infected the system and take appropriate actions.
The common disadvantage of all signatures where we match on `DHL_AWB_TRACKING_DETAILS` or `firehouse.exe` is that these detection methods are too specific on the file name. If the malware's name changes this detection method can completely miss the intrusion.

3.1 | Initial Access Signature

```
title: DHL_AWB_TRACKING_DETAILS.[zip/exe] created
description: Detects the creation event of the known malware 'DHL_AWB_TRACKING_DETAILS.[zip/exe]'
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 11
        TargetFilename|contains: "DHL_AWB_TRACKING_DETAILS."
    condition: selection
level: high
```

This signature detects when a file is created (`EventID 11`) that has the name `DHL_AWB_TRACKING_DETAILS`, which in our case detects both the event where the `.zip` file is downloaded and the event where it is unzipped into the malicious `.exe` file.
In either case, this event should immediately be addressed by a system admin or an anti-malware software since at this point the malware has not yet had the opportunity to cause harm, hence we have `level: high`.
In the given host events dataset this signature detects 5 log entries related to downloading/unzipping the executable file, including the entries shown in Figure 2.3 and Figure 2.4 as expected.
The advantage of this signature is that it is generic in what form or from what source `DHL_AWB_TRACKING_DETAILS` first infects the machine.

3.2 | Execution Signature

```
title: firehouse.exe created
description: Detects the creation event of the known malware 'firehouse.exe'
    (most likely created by 'DHL_AWB_TRACKING_DETAILS.exe')
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 11
        # Image|contains: "DHL_AWB_TRACKING_DETAILS.exe"
        TargetFilename|contains: "firehouse.exe"
    condition: selection
level: high
```

This signature detects when some process creates the `firehouse.exe` file (`EventID 11`). Note that this type of log does not contain the hash of `firehouse.exe` which would have been better to match. To be maximally generic, the expected source (in this case `DHL_AWB_TRACKING_DETAILS.exe`) is commented out.
In the given host events dataset this signature detects the single log entry also shown in Figure 2.5 as expected.
Similarly to the previous signature, the advantage is that it is generic in what form or from what source `firehouse.exe` first infects the machine.

3.3 | Persistence Signature

```
title: program added itself to runonce registry
description: Detects if a program adds itself to the runonce registry
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 13
        TargetObject|contains: "RunOnce"

    common_software:
        Image|contains:
            - "OneDrive"
            - "Discord"
            - "Steam"
            - "Edge"
            - "Skype"
            - "Docker"
            - "Dropbox"
            - "Spotify"
            - "Whatsapp"
            - "EpicGamesLauncher"
            - "Teams"
    condition: selection and not common_software

level: low
```

This signature detects when some program is added to the `RunOnce` registry (`EventID 13`). Note that this signature does not match specifically on `firehouse.exe` but instead matches on any program that is not one of the more common (legitimate) programs that are usually installed on a personal computer.
Since it is not expected that a benign program edits the `RunOnce` registry multiple times this signature is not expected to cause multiple alerts. Therefore a few false positives coming from this signature are deemed acceptable.
In the given host events dataset this signature detects 3 log entries, including the entry also shown in Figure 2.6 as expected.
The other 2 entries are from benign programs.
The advantage of this approach is that this signature will never miss any program that tries to add itself to the RunOnce registry.
The downside is that this signature may cause some false positive hits on programs that really are benign. And the list of benign programs needs to be expanded manually.

3.4 | Privilege Escalation & Defence Evasion

Since these steps of the attack happen implicitly by adding `firehouse.exe` to the programs that are run under the user's context there is no specific signature that detects this step.

3.5 | Credential Access & Collection of Data Signatures

3.5.1 | Signature Detection

```
title: looked up IP address
description: Detects when a program looks up the computer's IP address
    and other system information on some web browser
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 22
        QueryName:
            - "showip.net"
    condition: selection
level: low
```

This signature detects specifically when a process visited a website (DNS query with `EventID 22`) that reveals the device's IP address and other system data that might be useful to the attacker.
In this instance, it visited showip.net which is a normally benign website but reveals private information if accessed on the victim's computer.
The list of URLs could be expanded further with other common websites that reveal the device's private information.
This signature possibly produce false positives in the case where a normal user visits one of these websites. This is considered a rare false positive since most normal users won't need to find their own IP address like this.
In the given host events dataset this signature detects 41 nearly identical log entries at different days and hours, including the entry also shown in Figure 2.7 as expected.

The downside of this approach is that there exist an unmanagable number of options (including different websites) to fetch a computer's IP and other system information. If the malware's creator is aware of this signature it is trivial to circumvent this detection.

```
title: firehouse created file
description: Detects if the known malware 'firehouse.exe' creates an information file
    (like cookies or private user info) that will be sent to the attacker
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 11
        Image|contains: "firehouse.exe"
    condition: selection
level: medium
```

```
title: firehouse deleted file
description: Detects if the known malware 'firehouse.exe' deleted an information file
    (like cookies or private user info) to cover its tracks
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 23
        Image|contains: "firehouse.exe"
    condition: selection
level: medium
```

These signatures detect when `firehouse.exe` creates (`EventID 11`) or deletes (`EventID 13`) a file.
In this attack's case, it creates many such cookie/personal information database files it has scraped from (among other sources) the user's web browser.
Then as detailed in section 2.1 we see that the malware immediately after creation deleted those same information files to (attempt to) hide the evidence.
In the given host events dataset the creation signature detects 13 log entries of various user files being created, including but not limited to `FirefoxCookies.json`, `cookies.db`, and `WINDEV2310EVAL-user.txt`. The deletion signature detects 12 log entries of files being deleted, including the entries also shown in Figure 2.8 as expected.

The upside of this approach is (given the name of the malware .exe file) this will find all files that may be compromised.
The limitation is that we can't retroactively inspect the contents of these temporary files. So it is impossible to know what information was leaked.

3.5.2 | Suspiciously Quick Creation and Deletion

In addition to the above-mentioned 'pure' Sigma signatures we also developed a tiny Python notebook (Appendix B) that compares the creation and deletion timestamps (`Event.System.TimeCreated.#attributes.SystemTime`) to find all files that were created and then deleted within 5 seconds by `firehouse.exe`.
Using this (not so pure signature-based) detection method we were able to find 5 of the 12 temporary files we found by manual inspection.
The upside of this approach is that this no longer requires manually joining the outputs of the 2 separate signatures. But the downside is that this detection method can be evaded by the malware simply waiting longer before deleting their temporary files. The false positive rate for this Python-based signature is also extremely high. This signature picks up a lot of regular Windows events. Further development is needed to refine the signature and reduce the false positive rate.

3.6 | Discovery

```
title: firehouse edited registry keys
```

```
description: Detects when the known malware exe 'firehouse.exe' edited registry keys
    to find information about the system
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 13
        Image|contains: "firehouse.exe"
    condition: selection
level: medium
```

This signature detects when `firehouse.exe` sets the value of a registry key (`EventID 13`).
In the given host events dataset the creation signature detects 12 log entries where among others DWORD entries are modified in the registry. The advantage of this signature is that it detects exactly when and which registry entries `firehouse.exe` updates. This is useful to understand the level of persistence that the malware is trying to establish. However, the signature is not very general and could be better by detecting when other types of malware update the registry. This is however very difficult to do since identifying a legitimate and illegitimate process modifying the registry is difficult.

3.7 | Command and Control/Exfiltration Signature

```
title: firehouse connected
description: Detects when the known malware exe 'firehouse.exe' makes a TCP/UDP connection
author: Group 4
logsource:
    product: windows
    category: sysmon
detection:
    selection:
        EventID: 3
        Image|contains: "firehouse.exe"
    condition: selection
level: medium
```

In this signature, we detect any TCP/UDP network connection (`EventID 13`) being created by `firehouse.exe`.
In the given host events dataset the creation signature detects 3 log entries where the malware is connected to the servers `mail.abybay.com` and `static.2.60.55.162.clients.your-server.de` and as outlined in table A.2. This signature is advantageous since it detects the moment that the malware contacts its command and control server. This is useful for further analysis to mark malicious domains and IPs in public databases. The disadvantage here is that this signature only checks for `firehouse.exe` contacting its command and control server. The signature could be more general to identify other malware contacting their command and control servers.

4 | Mitigation and protection

The common source of infection for this attack is phishing emails. The infection occurs when a user receives an email that looks somewhat real and proceeds to download and run the attachment present in the email. This is a common method of infection as it is very effective due to the fact it is quite difficult to protect against it. The main mitigation strategy for phishing emails is constant user training where users are taught how to identify suspicious emails. For an organization, they should try to build a culture in which their users always double-check emails and are constantly on alert for fake emails. Apart from user training, there are also more automated solutions. One example is setting up email filters that can block phishing emails before they arrive in the user's inbox. These filters can analyze the content of an email, the reputation of the sender, and so on.
Another important mitigation strategy is having some sort of intrusion detection system in place. This can range from complex solutions to just an up-to-date anti-virus running on each computer. Suspicious files or links in emails can be run in a sandbox to determine potential malicious behaviour. Generally, the malicious files in phishing emails are known by most reputed anti-virus' so the malware can be quickly

detected and isolated.
If an infection has already occurred, it is important to isolate the incident to ensure that it does not compromise other computers on the network. Most anti-virus' can also isolate the malicious files locally, which is why having up-to-date anti-virus software running in the background is important. For example, given the attack investigated in this report, if the infected computer's internet access is disabled quickly enough the malware will not be able to contact its Command and Control server and thus not be able to upload the stolen data or spread it to other machines. This minimizes the opportunity for the malware to harm the system.

5 | Lab Session Setup

The laboratory activity consists of a presentation about the attack that was investigated. This will give the students some background information on the malware they will be investigating and the objective of the malware. Moreover, the dataset is introduced here as well with basic information like the type of available logs and which kind of events are present in the logs. Next, the students will be given the opportunity to do some threat hunting by themselves to find the discussed malware. This part is split into three sub-parts. Firstly, exploring the dataset and finding the initial infection. Then, identify the different steps the malware takes during its life. Finally, the students will develop signatures to detect this malware.

The three sub-parts are split into 10, 10, and 15-minute intervals. This is done to make sure that everyone in the class progresses at a similar pace. At the end of each interval, the expected outcome will be revealed. This makes sure that any students who were not able to complete the task do not get disadvantaged for the next step. This also helps ensure that all students maintain a similar pace throughout the lab activity. The students will be expected to develop the following signatures:

1. Presence of a file with the malicious hash (SHA: EC0EA1DA845EDA2DDE1D04E8B715EB8396B4000E).
2. Contact of Command-and-Control server at mail.abybay.com.
3. Bonus: look up of current computer's IP. (This is a bit tricky!)

These are signatures that we also developed which allows students to compare their results with ours. The first two signatures are, in principle, quite straightforward to develop to ensure that students can quickly develop them or at least a very simple version of them. The 3rd signature to be developed by the students is intentionally made a bit trickier and is not trivial to implement by just looking at the problem description. This is done to introduce some challenges to the lab activity and encourage out-of-the-box thinking. This also makes the lab activity more interesting.

The lab activity ends with the presentation of some signatures that we developed. These signatures are picked to be the same as those requested to be developed by the students, so they can verify their work and also gain some insights into how to improve their signatures for the future. When presenting our signatures, the thought processes behind why certain decisions were made will be explained and the signature will be evaluated in terms of its usefulness and false positive (and false negative) rate. Again, this is to provide more intuition for signature development to the students and to highlight which factors are important during signature development.

6 | Discussion

6.1 | Investigation and Development of Detection Method

We first tried to scan through all the host logs and found things normally out of place. Since the number of logs is large, we developed some filters to focus on suspicious executable files. After some initial trial and error, we focused on finding executable files with process creation (EventID 1) and file creation (EventID 11) activities and filtered out all logs from a safe path (`system32`, `SystemApps`, or `Program Files`), which was ultimately successful and helped us discover a critical malicious file `DHL_AWB_TRACKING_DETAILS.exe`. We then investigated all related logs, its SHA hash, and the logs related to the file `firehouse.exe` it

created, and gradually discovered more information. We also went through all the 279 logs containing `firehouse.exe` and wrote down every interesting step, which greatly aided our discovery of infection flow and subsequent development of detection methods.
There are some things that could have sped up this initial investigation. We should have looked for more obvious indicators of the intrusion, such as downloads of executable files. Reading about typical sources and how to identify them in the logs also helped. Applying such techniques could have meant we would have detected the malicious executable earlier.
We also should have sought possible guidance earlier when we encountered difficulties. Sometimes a small but crucial discovery can bring significant progress to intrusion detection.

6.2 | Detection Method Development and Evaluation

Developing signatures that can correctly identify malicious behaviour was a fairly complicated task however using behaviour seen in the logs was very helpful and ensured that the signature was able to detect malicious activity. However, all developed signatures are aimed at detecting another `firehouse.exe` infection. As such, evasion is possible by changing the name of the executable (this would be caught by hash signature if the file is not updated otherwise). More general signatures would be better to detect other types of malware and protect other types of infections that are not just from the `DarkCloud` family.

6.3 | Future opportunities

Currently, one of the real limitations is that our signatures could probably miss the same malware if the executable files were named a little differently. We expect that a more advanced signatures-based detection method would be able to adapt the string it matches on to be able to detect intrusions that perform similar steps that `DHL_AWB_TRACKING_DETAILS.exe` and `firehouse.exe` do.
The main future work would be the generalization of the signatures such that they are applicable to a wider range of malware that are not only from the `DarkCloud` family.

7 | Conclusion

Overall, a piece of malware from the DarkCloud family was investigated in this report to find how the initial infection occurs, its objectives, and the consequences that it brings. Moreover, each step of the malware was analyzed in detail to understand exactly how it achieves its objectives. Such an investigation is important for the development of signatures to capture the intrusion steps. Having an in-depth understanding is also important to be able to generalize the signatures to apply to other similar types of malware. Such a process allowed the development of the signatures in this report.
A focus was placed both on detecting this particular variant of the malware and on creating generalized signatures to detect other types of malware. However, during signature development, the limitations of signature-based detection immediately became obvious. The signature language used, `Sigma`, is also not very powerful meaning that the developed signatures are not very expressive. This made it more difficult to detect the complex behaviour of the malware. While signature-based detection is excellent if the threat is known in advance, it does not work so well on an unseen threat. In this case, the DarkCloud malware family is well-known, allowing fast signature development. However, if it was unprecedented, other methods like anomaly-based detection would have a much better chance of spotting the intrusion.
In the bigger picture, this goes to show that, to properly secure a system, only employing signature-based detection is likely not enough. Other methods like anomaly-based detection should also be employed for the best chance to catch an intrusion and quickly deal with it.

8 | References

[1] “Windows security log events,” Ultimate IT Security. [Online]. Available: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/

[2] “Sysmon documentation,” Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

[3] Virustotal. [Online]. Available: https://www.virustotal.com/gui/home/upload

[4] ANY.RUN, “Darkcloud malware analysis.” [Online]. Available: https://any.run/malware-trends/darkcloud

[5] Cyble, “Decoding the inner workings of darkcloud stealer,” 2023. [Online]. Available: https://cyble.com/blog/decoding-the-inner-workings-of-darkcloud-stealer/

[6] Microsoft, “Run and runonce registry keys,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys?redirectedfrom=MSDN

A | Appendix A Atomic Indicators of Compromise

In the tables below, all the indicators of compromise that were located are given. These include hashes of the malicious files, command-and-control servers contacted by the malware and files that were generated by the malware.

Table A.1: File hashes for the identified files associated with the malware infection.

Filename	Hash type	Hash value
firehouse.exe	SHA1	EC0EA1DA845EDA2DDE1D04E8B715EB8396B4000E
firehouse.exe	MD5	D2C0FDA1ECE3CC90733E291661A10162
firehouse.exe	SHA256	7950963B742A8B0D9F4E1FD6C642C8B8245A9DC668CE361C9F5390A86C8FD4AF
DHL_AWB_TRACKING_DETAILS.exe	SHA1	EC0EA1DA845EDA2DDE1D04E8B715EB8396B4000E
DHL_AWB_TRACKING_DETAILS.exe	MD5	D2C0FDA1ECE3CC90733E291661A10162
DHL_AWB_TRACKING_DETAILS.exe	SHA256	7950963B742A8B0D9F4E1FD6C642C8B8245A9DC668CE361C9F5390A86C8FD4AF

Table A.2: Command-and-Control servers contacted

Domain	IP	Port
mail.abybay.com	147.189.173.146	587
static.2.60.55.162.clients.your-server.de	162.55.60.2	NA

Table A.3: Files created by malware

File location and name
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\LogfrogeyeYfIToHcjSxADMKWhtnUiTukStgyePrjtKZqCizvXformamide
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\WebData
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\LogfrogeyeYfIToHcjSxADMKWhtnUiTukStgyePrjtKZqCizvXformamide
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\keyDBPath.db
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\CURRENT_USER.txt
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\Cookies
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\cookies.db
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\cookies.db-shm
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\cookies.db-wal
C:\Users\User\AppData\Roaming\Microsoft\Windows\Templates\CURRENT_USER\FirefoxCookies.json

B | Appendix B Detection of File Deletions through Python

Python codes to very fast detect the file deletions.

```
import pandas as pd

creations = pd.read_csv('creations.csv',
    parse_dates=['Event.System.TimeCreated.#attributes.SystemTime'], date_format=%b %d, %Y @
    %H:%M:%S.%f)
deletions = pd.read_csv('deletions.csv',
    parse_dates=['Event.System.TimeCreated.#attributes.SystemTime'], date_format=%b %d, %Y @
    %H:%M:%S.%f)

# extract only relevant info
creations_clean = creations[[Event.System.TimeCreated.#attributes.SystemTime,
    Event.EventData.TargetFilename]]
deletions_clean = deletions[[Event.System.TimeCreated.#attributes.SystemTime,
    Event.EventData.TargetFilename]]

# remove duplicates to avoid negative time
creations_clean = creations_clean.drop_duplicates(subset='Event.EventData.TargetFilename',
    keep=False)
deletions_clean = deletions_clean.drop_duplicates(subset='Event.EventData.TargetFilename',
    keep=False)

# merge on file name
merged = pd.merge(creations_clean, deletions_clean, on='Event.EventData.TargetFilename',
    how='inner')
merged = merged.rename(columns={Event.System.TimeCreated.#attributes.SystemTime_x: Creation
    Time, Event.System.TimeCreated.#attributes.SystemTime_y: Deletion Time})

# files with a existance time of less than 5 seconds are marked as suspicious
suspicious_files = merged[merged['Deletion Time'] - merged['Creation Time'] <
    pd.Timedelta('5sec')]['Event.EventData.TargetFilename']

print(Number of suspicious files found: , len(suspicious_files))

if len(suspicious_files) > 0:
   print(Suspicious files found:)
   for file in suspicious_files:
      print(file)
```