# OPERATION SOFT CELL – A WORLDWIDE CAMPAIGN AGAINST TELECOMMUNICATION PROVIDERS

*Mor Levi, Amit Serper & Assaf Dahan*
Cybereason, USA

{mor, amit, assaf.dahan}@cybereason.com

## ABSTRACT

During 2018, we identified a series of attacks targeting telecommunication companies. The attacks shared the same TTPs, consisting of a webshell execution followed by the deployment of Poison Ivy, a well-known RAT attributed to Chinese APT groups. This was the beginning of the APT saga. The targeted companies are in the telecommunication industry, have a global spread, hundreds of millions of customers and thousands of employees. Like other telecommunication operators, some of these companies have also gone through several M&A processes (mergers and acquisitions), which inevitably affected their IT infrastructure.

In some of these companies, we have identified that the same threat actor – which we have been tracking since early 2018 – has been operating in the company's environment for at least two years. The threat actor (like any well-organized R&D department, or organization) has a tasks Gantt for every quarter. We performed a retrospective analysis of the attacks we were able to identify, and found changes in the attack patterns and new activity every quarter.

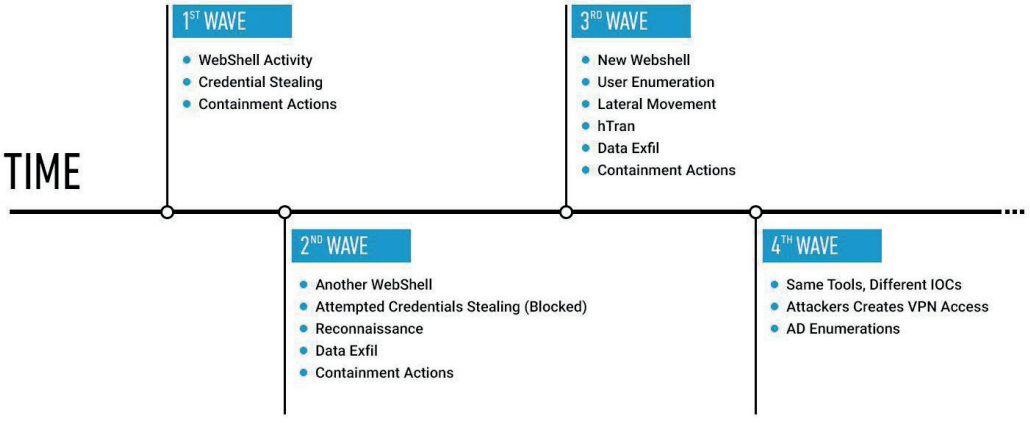



*Figure 1: Changes and new activity every quarter.*

## THE ATTACK

The initial indicator that 'sparked our imagination' was malicious activity performed by w3wp.exe, an IIS process, which was eventually classified as a webshell activity. By investigating the webshell

(which was later classified as the 'China Chopper' [1, 2] webshell), we were able to unravel several attack phases and TTPs used by the attackers.

The attackers leveraged the webshell to run reconnaissance commands and credential-stealing activities.

Base64-decoded parameters z1 and z2:

```
z1=cmdz2=cd /d "c:\inetpub\wwwroot\"&whoami&echo [S]&cd&echo [E]
```

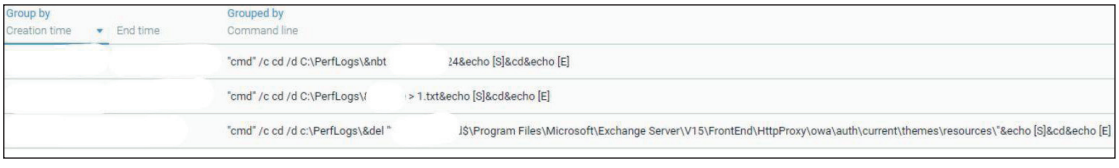



*Figure 2: Reconnaissance commands.*

One of the reconnaissance actions was to run a modified *NirSoft* NetBIOS scanner in order to identify available NetBIOS name servers locally or over the network.

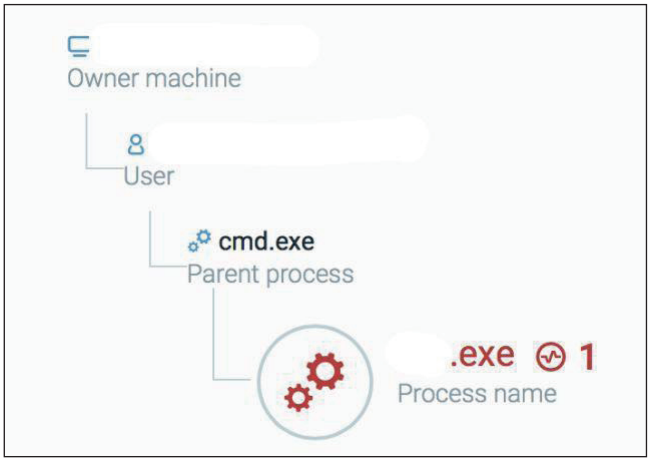



*Figure 3: A modified NetBIOS scanner was run to identify available NetBIOS name servers.*

The credential-stealing tool was a modified Mimikatz version, which, when executed, only dumps NTLM hashes (note that it does not require any command line arguments). We renamed this sample maybemimi.exe.

Reverse engineering shows the string similarity between maybemimi.exe and Mimikatz, as shown in Figures 5 and 6.

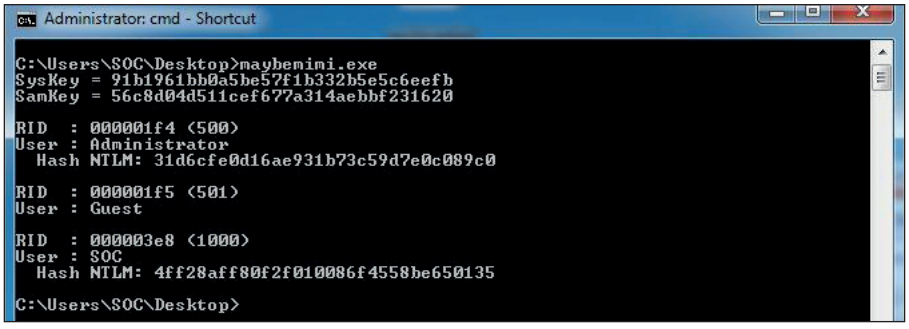



*Figure 4: We reamed the modified Mimikatz version maybemimi.exe.*

```
        }
        break;
default:
        PRINT_ERROR(L"Unknow SAM_HASH revision (%hu)\n", pHash->Revision);
}
if(status)
        kuhl_m_lsadump_dcsync_decrypt(cypheredHashBuffer.Buffer, cypheredHashBuffer.Le
if(cypheredHashBuffer.Buffer)
        LocalFree(cypheredHashBuffer.Buffer);
```

*Figure 5: Mimikatz code from GitHub.*

```
                align 20h
; char aErrorUnknowSam[]
aErrorUnknowSam:                        ; DATA XREF: sub_140002704+77↑o
                text "UTF-16LE", 'ERROR Unknow SAM_HASH revision (%hu)',0Ah,0
                align 10h
aNtlm:                                  ; DATA XREF: sub_140002704+209↑o
                text "UTF-16LE", 'ntlm',0
                align 20h
aNtlm_0:                                ; DATA XREF: sub_140002704+213↑o
                text "UTF-16LE", 'NTLM',0
                align 10h
aLm_0:                                  ; DATA XREF: sub_140002704+225↑o
                text "UTF-16LE", 'lm  ',0
                align 20h
aLm:                                    ; DATA XREF: sub_140002704+21A↑o
                text "UTF-16LE", 'LM  ',0
                align 10h
```

*Figure 6: Maybemimi strings.*

In mid-2018, while we were continuing to observe the attackers' activities, we spotted new webshells emerging, as well as reconnaissance and credential-stealing activity. In addition, we uncovered the usage of PiVy (Poison Ivy) by the attackers.

## PiVy

Poison Ivy (or PiVy for short) is a RAT that is associated with Chinese threat actors. PiVy is a powerful, well-featured RAT that allows an attacker to take total control of a machine. Among its notable features are:

- Registry editor
- Screenshot grabber
- Credential stealer
- Interactive shell
- File manager with upload and download support
- Process monitor
- Keylogging and various other surveillance features

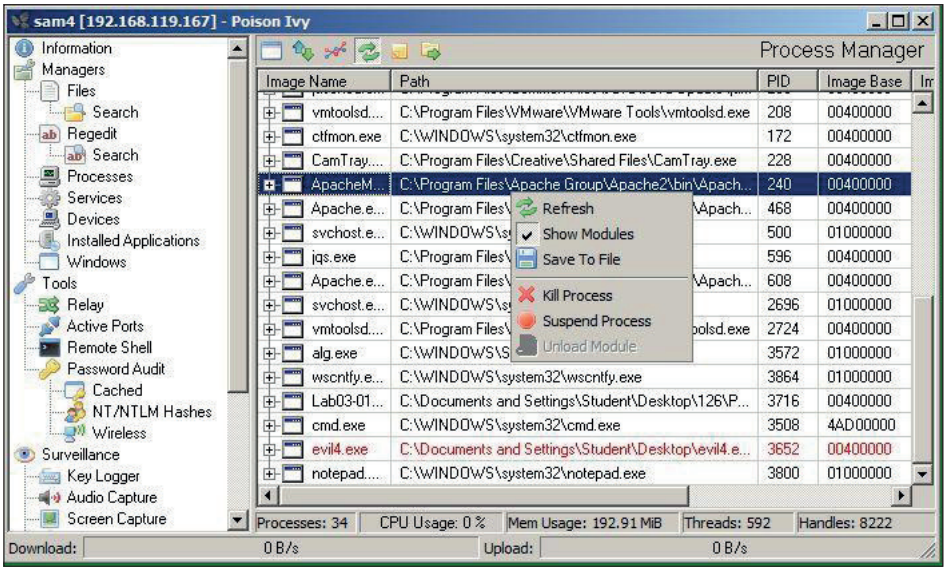



*Figure 7: Poison Ivy control panel (courtesy of Sam Bowne – samsclass.info).*

The strain of PiVy that we detected and investigated (and which also helped us to attribute those TTPs to the same attack group) is dropped by combining several methods:

1. A nullsoft installer package (NSIS) is created with a legitimate, signed *Samsung* tool in it.
2. Once executed, the installer script within the NSIS package extracts the *Samsung* tool (RunHelp.exe) and a DLL that's required by it: ssMUIDLL.dll.
3. That DLL contains a Poison Ivy stager, which will now be loaded by the RunHelp.exe application.
4. The machine is now infected with Poison Ivy.

The PiVy sample that we observed was running as a DLL that was loaded into the *Samsung* RunHelp.exe tool. This is another indicator that assisted us in attributing multiple attacks to the same threat actor. The method of using the NSIS installer in conjunction with the tool is not new and has been covered before by *Palo Alto* [3] (albeit with a different trojan).

Those findings, and more, led us to contact the attacked vendors and advise them on active containment actions that, later on, were put in place on an ad-hoc basis. We kept track in our research of the different TTPs of the threat actor, and there were no new indicators until the end of 2018. During that time, we detected interesting compressing activity carried out by the attackers in order to exfiltrate data. The attackers used *WinRAR*, which they downloaded from their C2 server, to compress the data they wanted to steal. By leveraging (another) webshell, and a renamed cmd.exe version, the attackers executed reconnaissance commands, collected data, dropped tools like portqry.exe and hTran, and performed lateral movement using `net use` and wmic.exe.

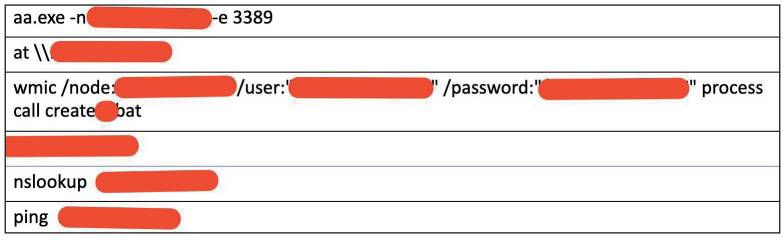



*Figure 8: Reconnaissance commands.*

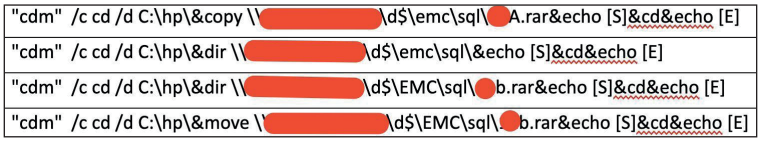



*Figure 9: Compressed stolen data.*

## hTran

One of the tools that the attackers deployed in order to exfiltrate the data from network segments that were not connected to the Internet was a customized version of hTran, a 'connection bouncer' tool which allows the attacker to redirect ports and connections between different networks. hTran's code is publicly available on *GitHub* [4].

When we first executed the version of hTran that was dropped by the attackers, we noticed that the debug messages were a bit hard to understand (Figure 10). It was very obvious that this had been done deliberately to throw off researchers.

However, since hTran's source code is available, comparing the debug output to the code allowed us to determine that it was indeed a modified version of the tool, as shown in Figure 11.

In Figure 11, we can see a disassembly of the modified hTran. We can see that printf is being called (dubbed by us as 'looks_like_printf') and the output is 'C e.'. However, when we look at the source code (Figure 12), we can see that this actually means 'connect error'.

## Understanding the motive

When you think of extensive breaches in large companies the first thing that comes to mind is usually payment data. A company that provides services to a large customer base has a lot of credit card data,

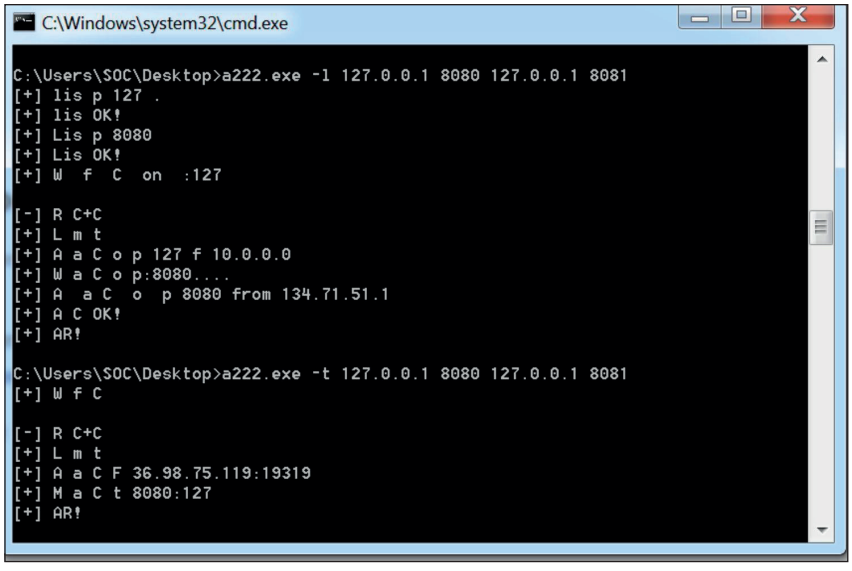



*Figure 10: Difficult to understand debug messages.*

```
v3 = name;
v4 = s;
v5 = gethostbyname(name);
if ( v5 )
{
  namea = 0i64;
  namea.sa_family = 2;
  *(_WORD *)namea.sa_data = htons(hostshort);
  *(_DWORD *)&namea.sa_data[2] = **(_DWORD **)v5->h_addr_list;
  if ( connect(v4, &namea, 16) >= 0 )
  {
    result = 1;
  }
  else
  {
    looks_like_printf((const wchar_t *)"[-] C e.\r\n");
    result = 0;
  }
}
```

*Figure 11: Disassembly of the modified hTran.*

```
if(connect(sockfd,(struct sockaddr *)&cliaddr,sizeof(struct sockaddr))<0)
{
        printf("[-] Connect error.\r\n");
        return(0);
}
return(1);
}
```

*Figure 12: Connect error.*

bank account information, etc. on its systems. If such a malicious operation is conducted by a cybercrime group then the motive of the attackers, usually, is to go after that payment data since the end goal is to make money.

However, this case was different. When a nation-state-sponsored group attacks a large company, in most cases, the end goal is not financial but rather intellectual property or sensitive information and data about one (or more) of its clients, which is exactly what happened in this case.

As we mentioned before, the breached companies were in the telecommunications industry (or telco for short).

One of the most valuable pieces of data that telcos hold are CDRs – call detail records. CDRs are basically a large subset of metadata containing all the details about calls, for example:

- Source and destination of a call
- Device details (IMSI, IMEI, MSISDN, etc.)
- Physical location (which phone was connected to which cellular site)
- Device vendor and its version (*iPhone*, *Samsung Galaxy* etc.)

Obtaining access to such data allows the threat actor to know more about the individual(s) they are targeting:

- Who they are talking to
- Which devices they are using (helpful for additional targeting of the victim's phone)
- Location patterns can be detected (e.g. what time the victim goes to work every morning, which routes the victim takes, what time the victim comes home, etc.)

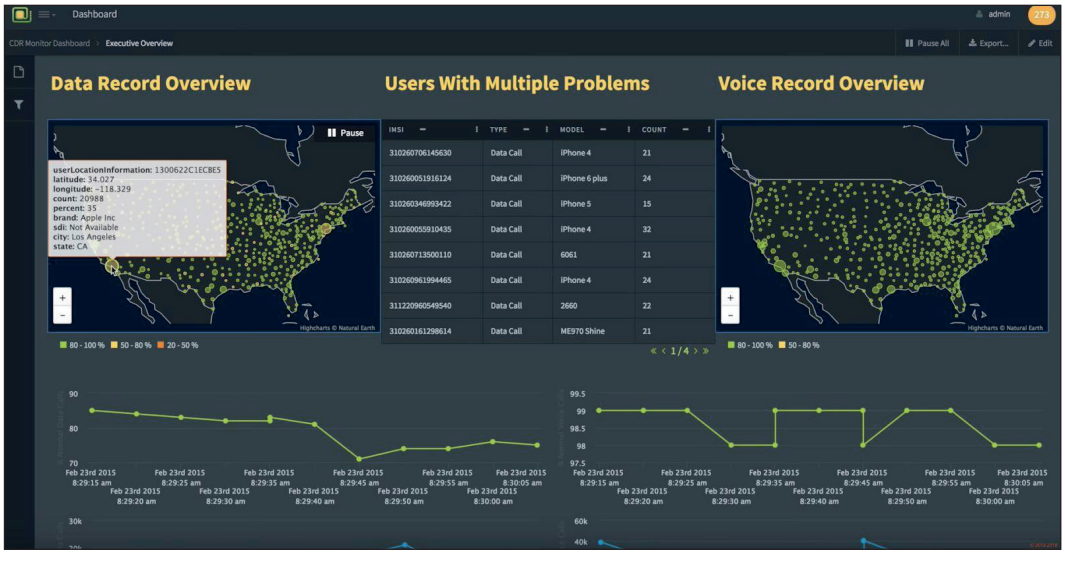
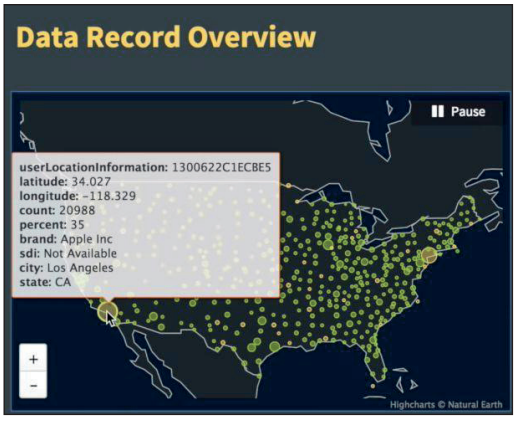



*Figure 13: Example CDR data.*




| IMSI | TYPE | MODEL |
|---|---|---|
| 310260706145630 | Data Call | iPhone 4 |
| 310260051916124 | Data Call | iPhone 6 plus |
| 310260346993422 | Data Call | iPhone 5 |
| 310260055910435 | Data Call | iPhone 4 |
| 310260713500110 | Data Call | 6061 |
| 310260961994465 | Data Call | iPhone 4 |
| 311220960549540 | Data Call | 2660 |
| 310260161298614 | Data Call | ME970 Shine |

*Figure 14: Details of CDR data.*

As a result of that information, the different TTPs we identified, and the data that was stolen during this campaign, we were able to attribute those attacks to a nation-state threat actor.

## THREAT INTEL RESEARCH

### Turning a pile of IOCs (data) into knowledge

When we research such a campaign, it is very important for us to manage all of our findings and ask very simple questions about them:

- What findings do we have?
  - Are there executable files?
  - If so, how were they executed and did they connect to any servers?
  - Were any of the executables packed?
  - Were they injecting code?
- How did the malicious software get there?
  - Who was patient zero?
  - Which resources were affected?

Our research started with one binary file and a single domain with which the file's process has communicated. Our methodology is described in the next section.

### Methodology

- Identify all the file hashes that communicate with the same IP or have the IP embedded in their strings
- Look for other samples with similar/identical functions and flow

- Check to which domain(s) this IP resolves
  - Check all the hashes that communicate with the resolved domain(s) or have the domain(s) embedded in their strings
  - Check the DNS history of the C2 domain
  - Check if those hashes communicate with additional IPs
  - Recursive process of the above
- Identify file similarity by large-scale string analysis, specific metadata of the file (company name, signature, version, icon, etc.) and tie the files to the same attacker
  - Can be done by online sandbox analysis
  - File scanning tools
  - Internal repository of files

At that point, we knew the following:

Webshells were used to execute multiple commands, both for reconnaissance and as well as for lateral movement and tool execution. We had the hashes of these tools and we also had a hostname to which all of the tools were connecting. We noticed that the attackers were using the same tools with minor changes (different names, string changes) across their campaigns, where all the payloads were connecting to the same IP address as the C2 server, albeit to a different hostname.

We started to try and make sense of everything by feeding all of that data into multiple threat intelligence sources that we have access to (both our own and third party).

***Note: Since we cannot share any IoCs, we will refer to file hashes, hostnames, IP addresses and other IoCs as generic placeholders.***

*Hostname1* is the hostname that was used for the C2 server targeting the telco companies.

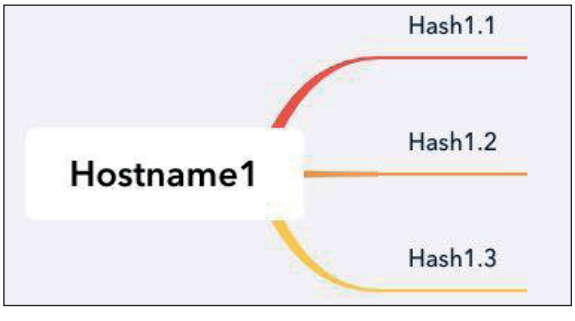



*Figure 15: All files were contacting hostname1.*

After analysing the files we saw that all of them were contacting the same host (hostname1), which was used as the C2 to which all of the malware and webshells were connecting.

Once we determined that all of the hashes in the scope of the attack were only connecting to hostname1, which is a dynamic DNS hostname, we wanted to see if we could find more information about the C2 server.

A simple WHOIS query revealed that the IP was registered to a colocation hosting company in Asia. There was no other publicly available information about this IP address.

We then started to query all of our threat intel resources about this IP address, and discovered that it was associated with multiple dynamic DNS hostnames.

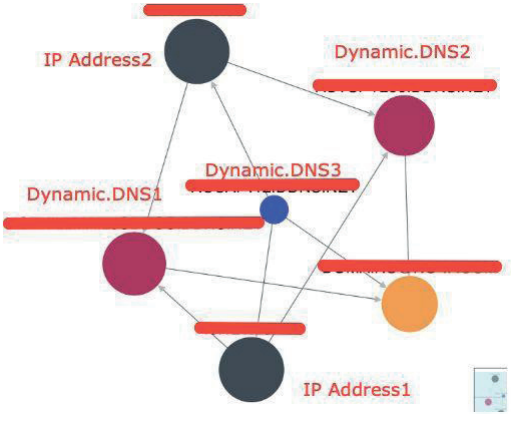



*Figure 16: The IP address was associated with multiple dynamic DNS hostnames.*

When we looked for any signs of connections to Dynamic.DNS2 and Dynamic.DNS3 we could not find any. However, they were registered and they were associated with IP.Address1. It was then time to determine the purpose of the other dynamic DNS hosts.

By leveraging various threat intel repositories, we crafted queries that searched for executables with these IPs and hostnames in their string table. One of the queries returned a few DLLs, with identical names to the DLL that we had initially investigated. However, the hashes were different. We took those DLLs, patched them back into the NSIS installer and detonated these samples in our testing environment. Once they were detonated, we saw that the configuration code that was injected into *nslookup.exe* did not have IP addresses and domains that were related to the first company that we identified in our initial research, but instead contained domains that were related to a different telecommunication company. Naturally, we reached out to that company, alerted them, and gave them all of the necessary information.

## Operational security and infrastructure

This pattern of behaviour allows us to understand the threat actor's modus operandi.

The threat actor uses one server with the same IP address for multiple operations. This server is their 'non-attributable' infrastructure.

The separation between operations is performed via different hostnames per operation, albeit hosted on the same server and IP address. While this method may be cheap and efficient for the attackers, it is almost transparent for a well-seasoned researcher with access to the right threat intelligence tools.

In addition, monitoring this infrastructure allows us to know if the attackers are starting new campaigns, engaging in new activities, making changes, etc.


*Figure 17: Strings from the dumped memory section of the injected shellcode. We can see many details about the campaign including domains and C2 IPs.*

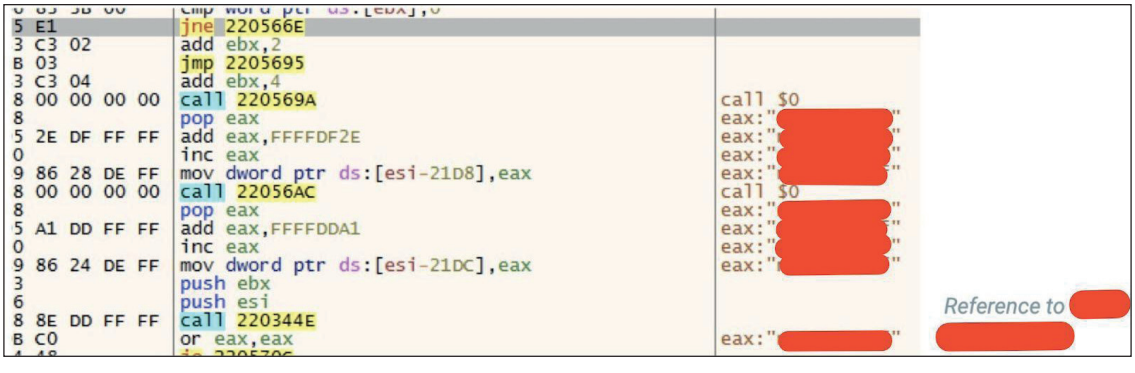



*Figure 18: Shellcode being unpacked and injected into a remote process. The redacted segments contain the name of the customer, C2 IPs and domains.*

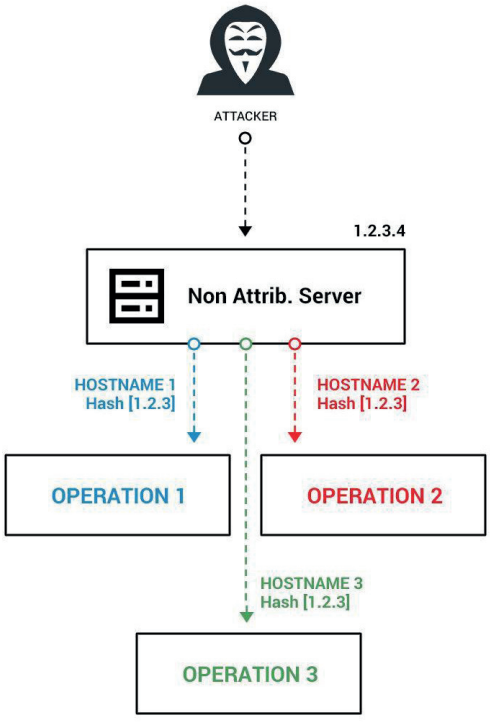



*Figure 19: The threat actor's modus operandi.*

When researching C2 servers one should watch for:

- Association with domains, especially if they are dynamic DNS domains
- File hashes which are associated with the IP or the domain of the C2
  - Static information and metadata from associated samples could be used to broaden the search for more information.

This demonstrates the importance of proper operational security (OpSec) and proper separation between tools and operations on the threat actor's side.

Throughout this investigation we have uncovered the complete infrastructure of the aforementioned malicious operations by this threat actor. We have observed a wide campaign against multiple telecommunication companies. The data exfiltrated by this actor, in conjunction with the TTPs used, allows us to determine with very high probability that the actor behind these malicious operations is backed by a nation state.

Due to multiple and various limitations we cannot include all of the information that we have on these attacks in this report. Our team will continue to monitor and track that malicious actor's activity in order to find more tools and compromised companies.

## CLOSING NOTES

This research, which is still ongoing, has been a huge effort for the entire *Cybereason Nocturnus* team. Special thanks go to Assaf Dahan, Noa Pinkas, Josh Trombley, Jakes Jansen, and every member of the *Nocturnus* team for the countless hours and effort that was put into this research. We will continue to monitor and update our blog with more information once available.

## REFERENCES

[1] China Chopper. MITRE ATT&CK. https://attack.mitre.org/software/S0020/.

[2] Lee, T.; Ahl, I.; Hanzlik, D. Breaking Down the China Chopper Web Shell – Part I. FireEye. August 2013. https://www.fireeye.com/blog/threat-research/2013/08/breaking-down-the-china-chopper-web-shell-part-i.html.

[3] Falcone, R. PlugX uses legitimate samsung application for dll side loading. Palo Alto. May 2015. https://unit42.paloaltonetworks.com/plugx-uses-legitimate-samsung-application-for-dll-side-loading/.

[4] hTran. https://github.com/HiwinCN/HTran.