New wave of PlugX targets Hong Kong

insights.oem.avira.com/new-wave-of-plugx-targets-hong-kong/

January 31, 2020

Mustang Panda is a well-known APT with a long history of targeting non-governmental organisations (NGOs). It utilizes shared malware like Poison Ivy, PlugX and Cobalt Strike payloads in order to gather intelligence. Since 2008, PlugX as a RAT (Remote Access Trojan) malware family has been used as a backdoor to control the victim's machine fully. Once the device is infected, an attacker can remotely execute several kinds of commands on the affected system to retrieve machine information, capture the screen, manage services, and manage processes.

Overview

Avira's Advanced Threat Research team, has been tracking Mustang Panda APT for a while. According to Avira's telemetry data, Mustang Panda mostly targets Asia-Pacific (APAC) countries and uses Cobalt or PlugX as payload.

Avira's Advanced Threat Research team discovered a new version of PlugX from the Mustang Panda APT that is used to spy on some targets in Hong Kong and Vietnam. The way that the APT actor infects the target, and launches the malicious payload is similar to previous versions—but with some differences.

PlugX executes DLL hijacking with benign applications such as ESET antivirus, Adobe Update etc. However, the way the PlugX loader launches the payload is different from how it was done for the previous versions. Also, the PlugX that Mustang Panda APT uses has some extra features, including spreading through USB, gathering information, and stealing documents in air-gaped networks via USB.

Mustang Panda APT uses a package of binaries to load the actual payload and it is intentionally designed this way to bypass file scanners and sandboxes. PlugX contains 3 files: benign EXE file for DLL hijacking, DLL (just a loader to execute the payload), and the encrypted payload (usually with ".dat" extension). The Advanced Threat Research team at Avira, have found different types of loaders (PlugX loader), and we will discuss it in more detail below. Obviously, file scanners or sandboxes can't detect the PlugX payload without the encrypted DAT file.

Anomali's Threat Research Team published a post about this campaign with the focus on the initial infection. In this article, we are going to dig into the PlugX loader and new PlugX payloads that Mustang Panda APT is using to spy on the targets.

First stage: Loader

The Loader is a tiny DLL file, the payload is an encrypted DAT file.

The DLL is responsible for decrypting the DAT file and executing it like a shellcode in memory. It reads the encrypted payload from disk by calling CreateFile and ReadFile APIs, and then allocates memory via VirtualAlloc. After decryption, it changes the protection via VirtualProtect and at the end executes it via direct call instruction.

DLL utilizes junk codes and a simple obfuscation to hide the API calls via stackstrings and load them dynamically.

The DAT file contains the decryption key inside of itself. The size of the key can vary. The loader reads offset zero of encrypted DAT file until it reaches null and takes that as an XOR key to decrypt the rest of the file.

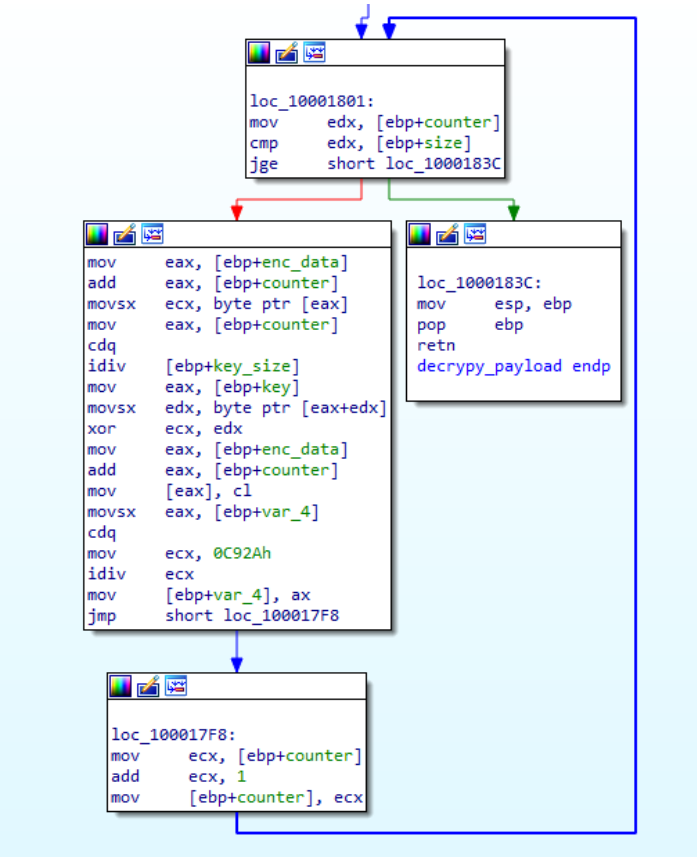
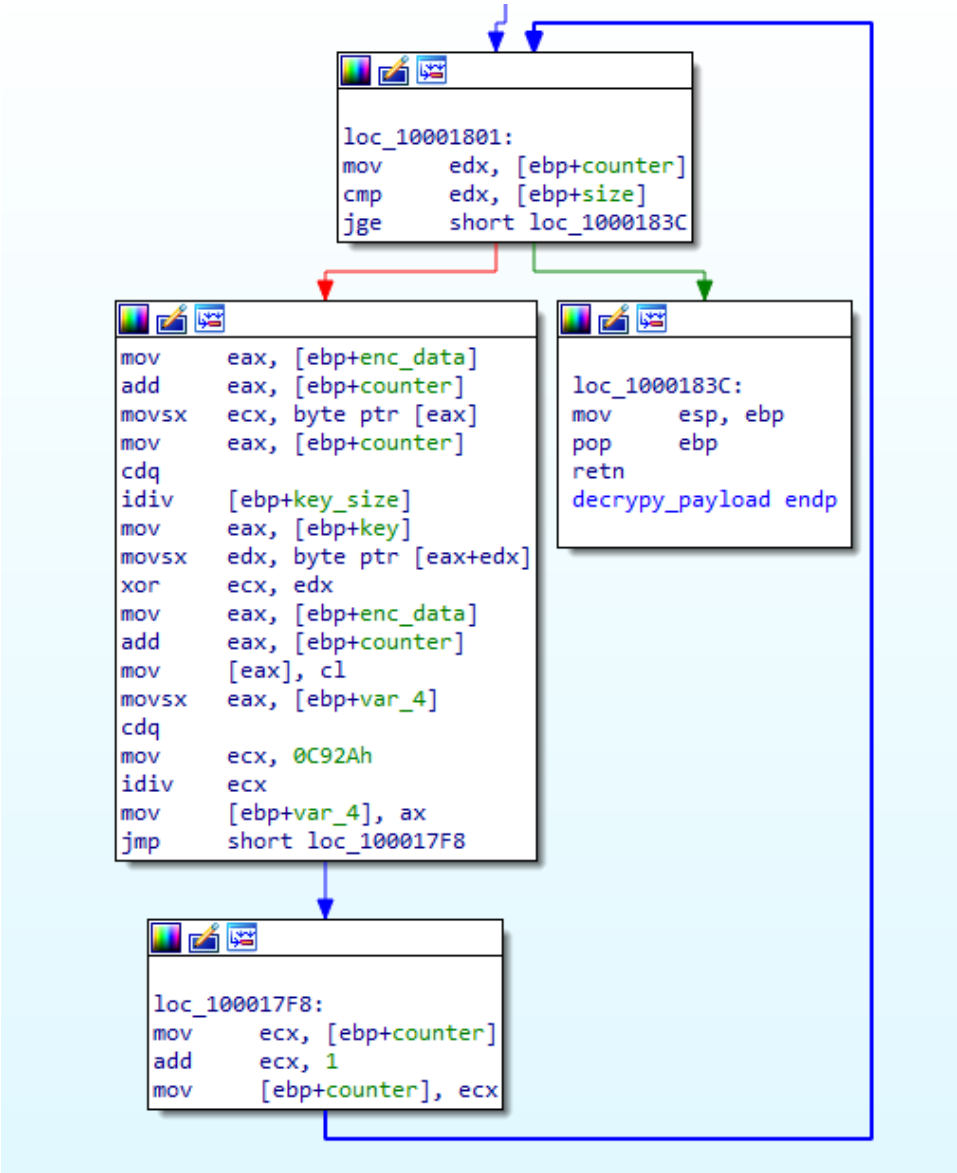


Figure 1: decryption routine for loader

Some of the loaders have hardcoded time inside the binary and they simply decrypt and execute the payload in a specific period of time.

For example, look at the following picture from one of the loader samples with time checking. The date is hardcoded “20190929.”



We have found 15 unique payloads.

We observed these DAT file names from our telemetry:

File Name

adobeupdate.dat
clntcon.dat
http_dll.dat
log.dat
mpsvc.ui
mp.dat

Payload

As explained previously, the payload is executed like a shellcode but surprisingly, it's a full PE binary. The loader will start the payload from the zero offset of the payload that it means from MZ. The picture below shows a very small crafted shellcode to call the entrypoint.

The decrypted payload is just loaded in memory and there are no footprints on the disk. This is the customized version of PlugX with simple obfuscations to bypass static detection and hide the strings.

Figure 2: crafted shellcode to call the entrypoint

The payload is a DLL with one export function called "Loader." The shellcode at the beginning calls the "Loader" export function.

The export function does the Windows loader job to load the DLL correctly and then executes the entrypoint.

Payload uses the wrapper function for every API call to obfuscate the API calling mechanism. In the wrapper function, there are stackstrings for the API function name and module name. In the picture below, you can see an example.

It can be executed in 3 different ways:

Argument	Description

no argument	Set a registry key for persistence and create a directory for files and run the malware again with -app
-app	• USB infection • Stealing documents and storing in USB for air gap network • Connect to the C&C address and execute the commands
-net	• set the value of key registry “System\CurrentControlSet\Control\Network\Version” to “1” • Set a registry key for persistence and create a directory for files and run with -app

For the persistence, it uses the registry run key:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
```

It creates %appdata%\Intel directory for additional modules and stolen information.

It uses a set of registry keys for storing data:

```
"System\CurrentControlSet\Control\Network\Version" → internet access
status
"Software\CLASSES\ms-pu\PROXY" → C&C server version
```

The PlugX TLS value names in the binary:

Thread name	Functionality
CXOnline::OlStartProc	starts CXOnline::OlStartProcPipe Initializes C&C communication
CXOnline::OlStartProcPipe	Initializes pipe communication objects Parses C&C commands
CXFuncShell::ShellT1 CXFuncShell::ShellT2	Remote Shell
CXSalvation::SalExceptionHandler	Global exception handler for logging in SS.LOG file
CXSoHttp::SoWorkProc	Sends requests to the C&C server

USB infection

One of the features that distinguishes this variant from other PlugX variants is the capability of spreading through a USB stick. This part of the code is different from other parts of PlugX as we don't see any TLS value name for this functionality.

After detecting the removable USB volume drive, the PlugX variant creates a hidden folder with name "RECYCLE.BIN" and copies the EXE file, the loader DLL, and encrypted DAT file. Then, it hides all of the folders in the root directory and creates LNK for each one in order to deceive the victim to click on the LNK files.

Air gap network

`PlugX checks whether there's no internet access, then tries to find the USB stick and creates the BAT file in the following path, <usb volume>\RECYCLE.BIN\<plugX clsid>\tmp.bat, and executes it. At the end, it deletes the BAT file.`

By running this BAT file, it gathers local and network information about the infected system and tries to send it out via USB.

```
cmd.exe /c systeminfo > <usb volume>\RECYCLE.BIN\<plugX clsid>\sys.info

cmd.exe /c ipconfig /all >> <usb volume>\RECYCLE.BIN\<plugX clsid>\sys.info

cmd.exe /c netstat -ano >> <usb volume>\RECYCLE.BIN\<plugX clsid>\sys.info

cmd.exe /c arp -a >> <usb volume>\RECYCLE.BIN\<plugX clsid>\sys.info
```

cmd.exe /c tasklist /v >> <usb volume>\RECYCLE.BIN\<plugX clsid>\sys.info

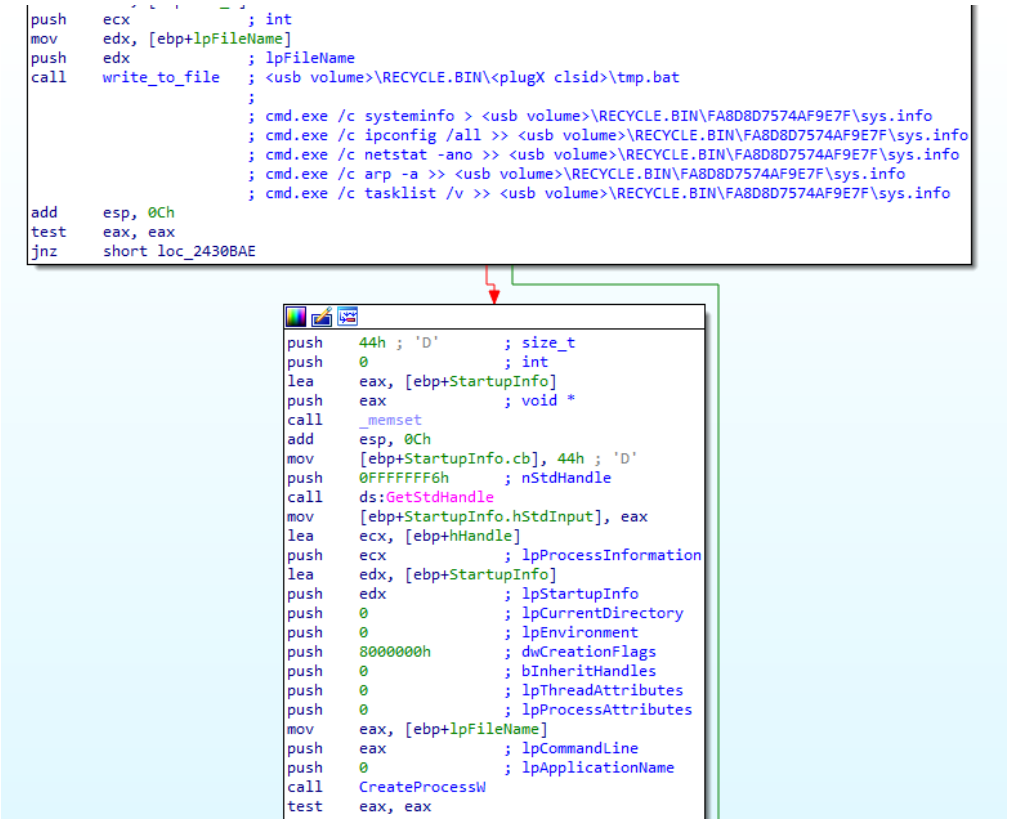


Figure 3: run BAT file for gathering info

It searches the whole system for the following file extensions:

document extensions
.doc
.docx
.ppt
.pptx
.xls
.xlsx
.pdf

It encrypts the mentioned documents on the fly via RC4, then copies them in its specific folder in the usb: “<usb volume>\RECYCLE.BIN\<plugX clsid>”.

The RC4 key is hardcoded in the payload binary.

From our telemetry data, we found a different way of stealing documents. It abuses RAR to search document files and compress them locally in order to send it to its CNC server.

The command line looks like this:

Rar.exe a -r -ed -m3 -dh -tk -hp<password> -v100m -ta<file modified date after> -n*.doc* -n*.xls* -n*.pdf <compressed file path> <searching directory>

As you can see, it's looking for a specific date to compress .xls* and .pdf extensions. It also uses a password to encrypt file data and header.

Configuration

The config can be encrypted or be plaintext in the binary. If the config data starts with "XXXXXXXX," it means the config is not encrypted.

If the config is encrypted, the config data is decryped via XORring with the hardcoded key inside of the binary. In different payloads that we could find, the key was "123456789".

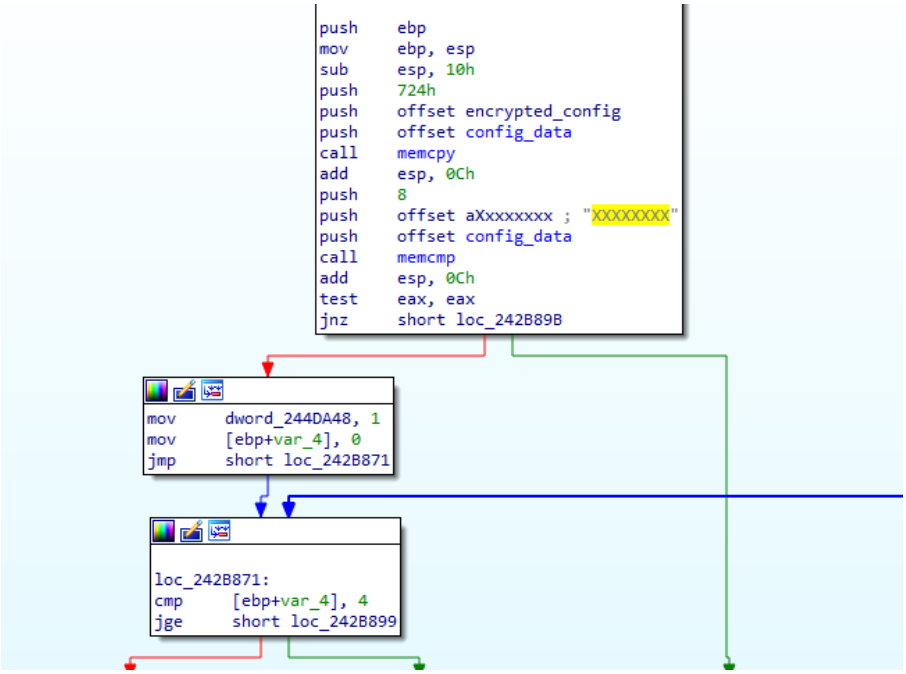


Figure 4: check config data before decryption

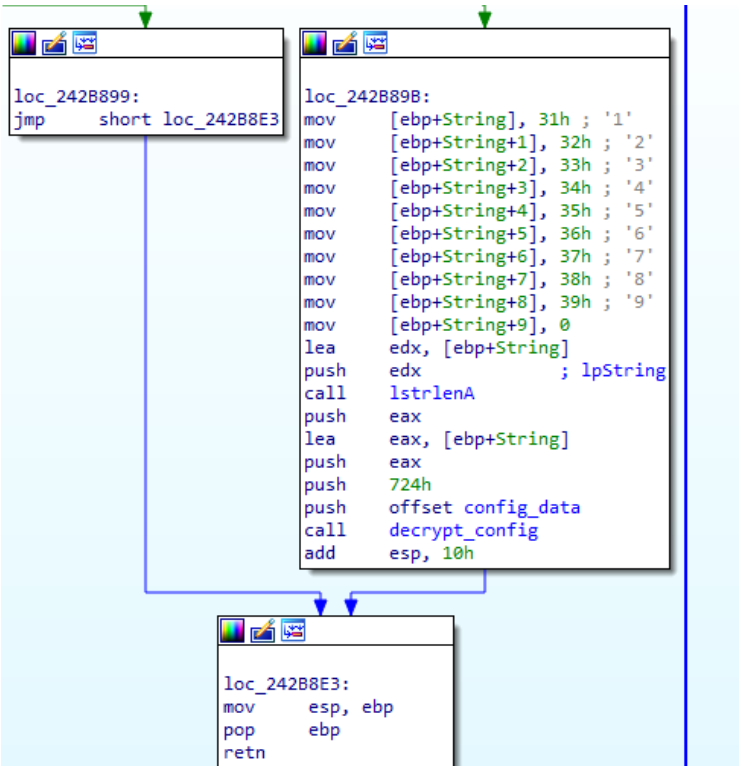


Figure 5: hardcoded key for config decryption

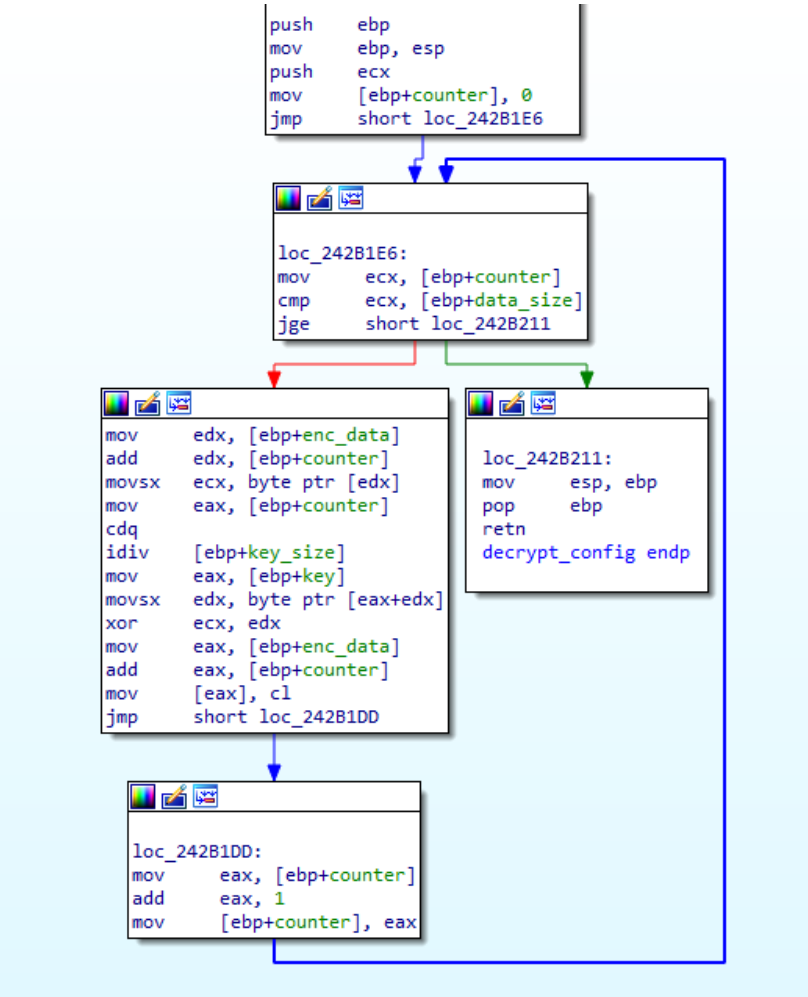


Figure 6: config decryption routine

Inside the config, there is information about the folder name for PlugX modules, mutex name, C&C IP and domain address, and port number.

We found 17 unique pieces of config data. This is the information we extracted from them:

C&C domains

www.apple-net.com
www.mmfhlele.com
www.olk4.com
update.olk4.com
infosecvn.com
aridndvn.ccom
www.freesmadav.com
update.freesmadav.com
www.lameers.com

C&C IP address

154.223.150.105
43.251.182.114
185.239.226.61
167.88.180.132
45.251.240.55

Folder Names

ESET Malware ProtectionrYF
ESET Malware ProtectionWgM
ESET Malware ProtectionTCp
ESET Malware ProtectionOWT
ESET Malware Protectionmld

```
Microsoft Malware ProtectionGCg

Microsoft Malware Protectionfpx

Microsoft Malware ProtectionwSA

Microsoft Malware ProtectionpiW

Microsoft Malware ProtectionSaG

Microsoft Malware ProtectionDQA

Microsoft Malware ProtectioncdB

Microsoft Malware ProtectionNSP

Microsoft Malware ProtectionGHQ

Adobe Update HelpereQU

Adobe Update HelperzGI

Adobe Update HelperatX
```

Mutex names

VVubPDixKeBURoQIIyfb
ypFRoazKbRHpMwnXoLtW
XUzeONpJmKCaBUtvnRGB
NNOFQoZIxAphdklhtaWw
GRNtLeLPnuJGPsTxTilb

Interestingly, we could also find the test sample that the malware author may have used to test the malicious binary before releasing with the IP address 127.0.0.1 for C&C in the config data.

C&C Communication

The PlugX variant uses a new way to communicate to its C&C server but it also supports the old communication mechanism. So, when it initializes the connection and gets the C&C version, it can decide how to communicate.

The CNC commands in this PlugX variant are similar to other versions with some minor differences.

It has 2 different groups of commands: group IDs 0x1001 and 0x1002. Once it gets the C&C group ID, it decides how to respond to the commands.

For command group ID 0x1001:

Command	Description
0x1001	get system information → memory status, computer name, username, OS version, PlugX CLSID, cpu speed, cpu architecture, width and height of screen
0x1002	start pipe communication → CXOnline::OlStartProcPipe
0x1003	echo input back
0x1005	exit process

For command group ID 0x1002:

Command	Description
0x3000	get disk information
0x3001	find file
0x3004	read file
0x3007	write file
0x300A	create directory
0x300B	check if the file existed
0x300C	create process in hidden desktop
0x300D	file operation (rename, move, delete, copy)
0x300E	get expanded environment information
0x300F	get plugX directory

0x7002	remote shell

Old communication mechanism

It uses HTTP post request to start the communication. The URL it uses to send the request to the server is “/update?wd=<random 8 digit number>”

It uses this hardcoded user-agent:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;SV1;
```

It adds these parameters to the HTTP request header

```
x-debug
x-request
x-content
x-storage
```

New communication mechanism

PlugX sends an encrypted random ID to initialize the connection via raw TCP and then receives a response from server.

The first 16 bytes are the header:

- The first 4 bytes are the encryption key.
- The second 4 bytes are CNC command code (the 29th bit shows whether the data is compressed and 30th bits shows whether the data is encrypted).
- The next 2 bytes show the size of data (16 bytes header is not included).
- The header is encrypted as well with the first DWORD sent by the server.

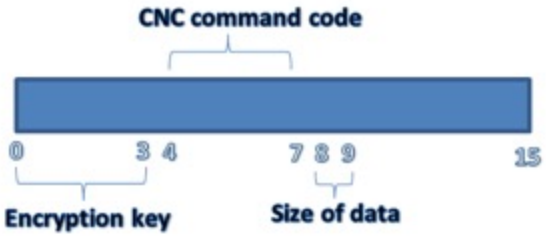


Figure 7: structure of C&C communication header

The encryption algorithm used for communication is XOR but adds a hardcoded value to the key for every round of encryption. The hardcoded value for the binaries that we found was “6666”.

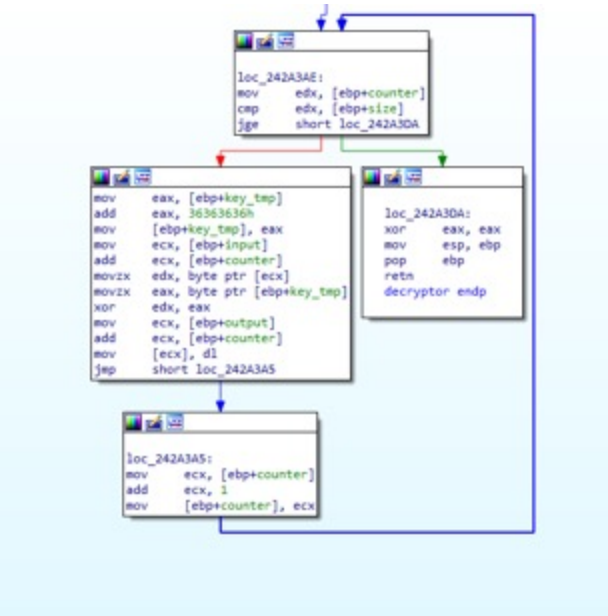


Figure 8: decryption routine for communication data

Conclusion

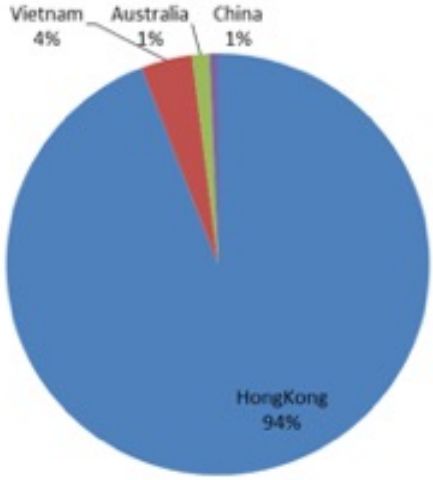
The Mustang Panda APT actor uses PlugX with minor changes, in an attempt to evade detection. This time, we found new features, new config structure, and loader—which caught our attention.

Loading the payload in multiple stages with different modules and encrypted data is a well-known technique but still quite effective to bypass sandbox systems and file scanners.

It is always challenging to have all of those modules together in order to reach the final payload.

Based on our telemetry, we found several victims from Hong Kong, Vietnam, Australia, and China but most of the victims are from Hong Kong.

Learn how Avira prevents zero-day attacks, detect targeted malware and deliver the highest detection rates in the cyber-security industry.



Indicators of Compromise

c90cae0a4365cb31f171b051520f6c8053dd0c3e798c59b2ae418bf99ddad02c
14f9278f3515fae71ccb8073cfaf73bdcc00eab3888d8cee6fb43a4f51c9e699
6fc8c2e28dfa39b20154ecb3339eb784a025ea1a7c79e15e0930f679280bb63e
f331eb3d7f6789e48f2e3bfb1a87595561722f45aaec150df537488587024096
b9f3cf9d63d2e3ce1821f2e3eb5acd6e374ea801f9c212eebfa734bd649bec7a
8be6c10e9e150d01601f77485444e409667ba905100982f57743e01d20a26121
6b23a388ddb3b697004fdd37a0d393455ae5702040c2b694f9158112698c2ec1
6924581b5fee4699a1ce0182cdf8462d5fcb5389985ec129d554dd6ca45d0c9f


Shahab Hamzeloofard

Shahab Hamzeloofard is a threat researcher and software engineer in Avira's Advanced Threat Research team. With a decade of experience in the cyber-security industry he enjoys implementing modules for hunting advanced malware and reverse engineering malware binaries. He is passionate about CPU internals, hypervisor and reversing complex and obfuscated binaries. He loves playing CTF and researching about new exploitation techniques.