Nov 19, 2015

# Decrypting Strings in Emdivi

Hello, this is You 'Tsuru' Nakatsuru at Analysis Center.

As introduced in the previous blog post, my colleagues presented on the attacks arising in Japan at CODE BLUE 2015, entitled "Revealing the Attack Operations Targeting Japan".

In this entry, I will introduce the details of an IDAPython script "emdivi_string_decryptor.py", which JPCERT/CC developed to analyse Emdivi, a remote control malware. The script was also introduced in our presentation at CODE BLUE 2015. Please utilize the script along with the codes, etc., that are already published on JPCERT/CC's GitHub account.

> JPCERTCC/aa-tools · GitHub
> https://github.com/JPCERTCC/aa-tools

## Encrypted Strings within Emdivi

Emdivi encrypts strings such as URLs that they connect to and stores them in itself. Depending on the sample, encrypted strings are Base64-encoded and stored as in Figure 1, or in other cases, just saved in an encrypted binary format.

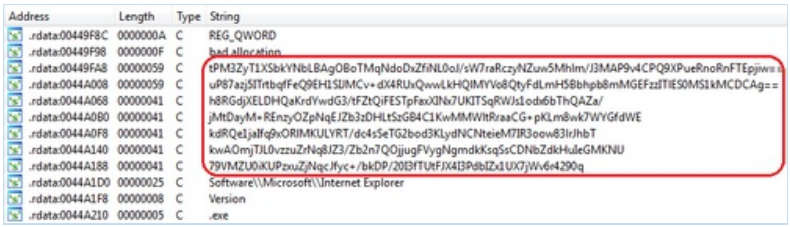


Figure 1: Encrypted strings encoded with Base64

In the incident analysis phase, these encrypted strings need to be decrypted in order to identify information such as URLs that the malware connects to, etc. For this purpose, JPCERT/CC developed emdivi_string_decryptor.py.

## Analysis on Emdivi's Decryption Process

Emdivi runs the following process in order to decrypt the strings in itself.

Upon its startup, it calculates the key for decryption, based on the following string information:

- The sample's version strings

  e.g. t17.08.26..3340.4444

- Random long strings in the sample

  e.g. jp5cQEhSR7xMEdv1JOjh5eKGsMxSCAE5M57CijC8VgN1KMbBvP9
  (Omitted hereafter)

It then combines these strings with Base64 encode, MD5 hash value calculation and others to calculate the decryption key as in Figure 2. Depending on Emdivi's version, it sometimes combines addition and other arithmetic process for this calculation.

R E C E N

Decrypting Strings in Emdivi
A Volatility Plugin Created for Detecting Malware Used in Targeted Attacks
Emdivi and the Rise of Targeted Attacks in Japan
The 5th CERT-RO Annual International Conference in Bucharest and Latest Cyber Security Trends in Romania
APCERT Annual General Meeting and Conference 2015 in Kuala Lumpur
VPN Servers Altered by Attacker Leading to Scanbox, a Reconnaissance Framework
Enhanced Protected Mode in Internet Explorer
PoisonIvy adapts to communicate through Authentication Proxies
The 27th FIRST Annual Conference in Berlin
Protected Mode in Internet Explorer

C A T E G

#APCERT #FIRST #Incident management #JPCERT news #Threats #Trends in Japan #Tsubame #Vulnerabilities Africa India Indonesia Laos Mongolia Myanmar Pacific Islands Sri Lanka Thailand

A C C E S

A Volatility Plugin Created for Detecting Malware Used in Targeted Attacks
Emdivi and the Rise of Targeted Attacks in Japan
Analysis of a Recent PlugX Variant - "P2P PlugX"
A New UAC Bypass Method that Dridex Uses
Decrypting Strings in Emdivi
The 5th CERT-RO Annual International Conference in Bucharest and Latest Cyber Security Trends in Romania
APCERT Annual General Meeting and Conference 2015 in Kuala Lumpur
VPN Servers Altered by Attacker Leading to Scanbox, a Reconnaissance Framework
PoisonIvy adapts to communicate through Authentication Proxies
Fiddler Core's insecure Default flag may lead to Open Proxy Issue

L I N K S

JP JPCERT homepage
Follow us @jpcert_en
RSS feed

```
push    8
pop     ecx
mov     edi, offset encryption_key      ; ed0cffb45ea86a40f43f98025f4a0d6c
push    ebx                             ; MaxCount
rep movsd
push    1                               ; char
lea     ecx, [ebp+var_2C]
call    sub_401C6A
call    __EH_epilog3_GS
retn
aa_make_encryption_key endp
```

Figure 2: Calculated decryption key

Using the decryption key calculated here, Emdivi performs decryption just before it uses the strings. Processes related to the encryption are implemented as classes. Figure 3 shows information of those classes.

```
                dd offset const CryptoLibs::Base::`RTTI Complete Object Locator'
const Base::`vftable' dd offset __purecall
                                        ; DATA XREF: XxTea::XxTea(void)+193↑o
                                        ; sub_43D62D:loc_42730C↑o
                dd offset __purecall
                dd offset __purecall
                dd offset const CryptoLibs::XxTea::`RTTI Complete Object Locator'
const XxTea::`vftable' dd offset aa_XxTea_convert
                                        ; DATA XREF: XxTea::XxTea(void)+107↑o
                dd offset aa_XxTea_decrypt
                dd offset aa_XxTea_encrypt
```

Figure 3: Encryption related classes defined within Emdivi

Many samples of Emdivi use XxTEA as in Figure 3, but we have confirmed that some versions use AES. Also, there are some versions that switch the encryption and decryption process, and we have seen that some use XxTEA encryption process for decryption.

After analysing various samples of Emdivi, we were able to summarise the method for the decryption key calculation and the decryption process as in Chart 1 below. For details of the decryption key generation process, please refer to the source codes of emdivi_string_decryptor.py.

Table 1: Decryption process for each Emdivi version

| | t17 | t19 and t20 mid versions | t20 early and late versions |
|---|---|---|---|
| Decryption process | XxTEA Decryption | XxTEA Encryption | AES Encryption |
| Method to calculate decryption key | MD5(<br>MD5(base64(ver))<br>+<br>MD5(key_string)) | scanf(<br>"%x%x%x%x",<br>Inc_Add(ver17_key)<br>) | Inc_Add(ver17_key)<br>[:24] |

## Before Using emdivi_string_decryptor.py

Since emdivi_string_decryptor.py is an IDAPython script, it requires disassembler IDA for execution. Also, the version string used when calculating the decryption key is required for string decryption.

Before actually using the tool, you have to obtain the version string from the memory or communication data.

If you are obtaining the string from the memory, execute Emdivi in an analysis environment and then search for the string in the memory by using debugger, etc. You do not have to worry about virtual environment detection, since the version string is generated before the detection process.

If you are obtaining the version string from the communication data, you would need to decode that data. For decoding, you can use emdivi_postdata_decoder.py which is also made public together with emdivi_string_decryptor.py. Here below is what you will see when executing by giving the data you want to decode to emdivi_postdata_decoder.py's argument.

```
> python emdivi_postdata_decoder.py
"r13ftV=C%5DZ%03k%07%06%7Edkgd%05%19%7Dq%05%05%1E%0D%02%0C;yhmsuRvo=%00;date=-
```


Contributor info

A R C H I

November 2015
October 2015
September 2015
August 2015
July 2015
June 2015
May 2015
April 2015
March 2015
February 2015

More...

C A T E G

#APCERT
#FIRST
#Incident management
#JPCERT news
#Threats
#Trends in Japan
#Tsubame
#Vulnerabilities
Africa
India
Indonesia
Laos
Mongolia
Myanmar
Pacific Islands
Sri Lanka
Thailand

```
b%27f.4%60%25%23%3A%24%2C%3A%26%22%3A%3A%27%27%20%24%3A%20%20%20%20%1Dh%1Dz%40.4%22%3A%25%3A%23%22%24%25%
%3D"
[*] 3 field(s) found in postdata
"r13ftV"        ->      "win7_32JP_SP1-IE11*968"
"yhmsuRvo"      ->      "1"
"date"  ->      "9v3r: t17.08.26..3340.4444     |       NT: 6.1.7601    [ja-JP]
|       MEM: 2048M      |       GMT(9)"
```

Please note that the version string included in the communication data may be different from the string required for the decryption. Therefore, we recommend obtaining the string from the memory.

## Executing emdivi_string_decryptor.py

If you have obtained the version string, you are all set. Load the target Emdivi into IDA, and execute emdivi_string_decryptor.py. Then, it will process as follows:

1. Input version string
2. Calculate decryption key
3. Search for encrypted string
4. Decrypt string
5. Output results and insert comments in the corresponding section

Upon execution of emdivi_string_decryptor.py, the following dialog box appears to input the version string.

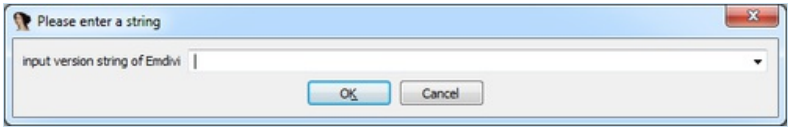


Figure 4: Dialog box to input version string

After you input the version string, the tool will display the decrypted string in the console, and it will be inserted as a comment to where the encrypted string is stored. This is shown in Figure 5.

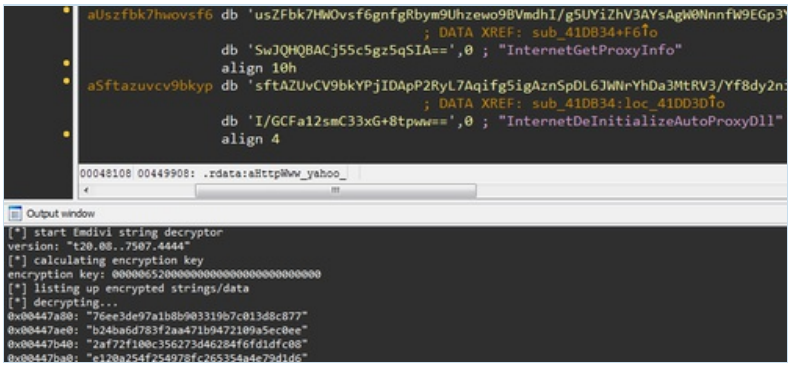


Figure 5: Screenshot after executing emdivi_string_decryptor.py

Now you can obtain various information including URLs that the malware connects to. Based on these and other related pieces of information, JPCERT/CC coordinates and handles incidents caused by attack operations involving Emdivi.

## In Summary

We hope that the scripts that we made public will contribute in dealing with the attacks relating to Emdivi, and in improving malware analysis techniques.

A blog entry by Kaspersky (see Reference) has revealed that a few versions of Emdivi use the infected users' SID. Unfortunately, the current version of emdivi_string_decryptor.py is not yet adapted to input SID. Furthermore, it is possible that new versions of Emdivi with other encryption methods may emerge in the future. We welcome any pull requests on GitHub.

Thanks for reading.

-You Nakatsuru

---

**Reference**

New activity of The Blue Termite APT - Securelist
https://securelist.com/blog/research/71876/new-activity-of-the-blue-termite-apt/

Posted on Nov 19, 2015 in #JPCERT news, #Threats, #Trends in Japan | Permalink