Classifying Malware using Import API and Fuzzy Hashing – impfuzzy –

Hello all, this is Shusei Tomonaga again.

Generally speaking, malware analysis begins with classifying whether it is known malware or not. In order to make comparison with the enormous number of known malware samples in the database in a speedy manner, hash values are used, derived by performing hash functions to the malware sample.

Among the different hash functions, traditional ones such as MD5 and SHA1 derive totally different hash values if the input data is different even by 1 bit. So this would not be useful when you have a sample that is similar to a known malware, and would like to classify it as "known".

These days, malware used in attacks is mostly customised for each case, so it is ideal to use hash functions which can classify those samples as similar to known ones.

Therefore, fuzzy hash function (of which the hash value for modified codes is close to that of the original codes) is used if modifications are simple and mechanical, and for Windows executable file samples, imphash [1] (import hash) is used, which calculates values from PE (portable executable)'s import table.

One of the known examples of fuzzy hash is ssdeep [2].

However, there are various issues with these methods used for malware classification; hash values derived by ssdeep often do not coincide with the similarity of malware samples, while imphash derives different values once new functions are added to the malware.

This blog article proposes a new method "impfuzzy", and demonstrates how it can accurately classify similar malware, in comparison to existing methods.

impfuzzy

The proposed method, as in imphash, calculates values from Import API, however, it also uses Fuzzy Hashing to calculate hash values of Import API, in order to supplement the shortcomings of imphash. With this process, a close value will be derived if just a part of Import API was added or modified.

Furthermore, it reduces time for calculation and enables efficient comparison by specifying the object of the hash value calculation to Import API (and not to the Fuzzy Hashing value of the whole executable file).

Implementing impfuzzy

"pyimpfuzzy", a Python module for calculating and comparing impfuzzy, is available on GitHub (a public web service for software development projects). Feel free to download it from the following URL for your use:

JPCERTCC/aa-tools GitHub - impfuzzy
https://github.com/JPCERTCC/aa-tools/tree/master/impfuzzy/

Volatility Plugin is also released on the portal, which searches for similar files based on hash values of executable files loaded from memory images by using impfuzzy.

In order to use pyimpfuzzy, the following tools need to be installed.

For this implementation, ssdeep is used as Fuzzy Hashing.

- pefile (Version: 1.2.10-139 and above)
- ssdeep (http://ssdeep.sourceforge.net)

pyimpfuzzy has the following functions:

- get_impfuzzy: calculates hash values from selected PE files
- get_impfuzzy_data: calculates hash values from PE files in a data format
- hash_compare: compares two hash values and returns the degree of similarity within the range of integer 0-100 (100 when same)

The similarity of the two PE files is calculated as follows:

```
import pyimpfuzzy
import sys

hash1 = pyimpfuzzy.get_impfuzzy(sys.argv[1])
hash2 = pyimpfuzzy.get_impfuzzy(sys.argv[2])
print "ImpFuzzy1: %s" % hash1
print "ImpFuzzy2: %s" % hash2
print "Compare: %i" % pyimpfuzzy.hash_compare(hash1, hash2)
```

The following result is derived when executing the above script:

```
mal@works:~$ python impfuzzy.py dyre.1 dyre.2
ImpFuzzy1: 24:tDXrcD3fc3sGLO1jpcWivkOolFOIzE8rxJz28zuKmwX35gg9FKqTezv/xaBWh:tb6bcgOQzlr2RKrmGwTNh
ImpFuzzy2: 24:tDXrcD3fc34yJ7QjpEkWivKOolFOuzEPrFOM3R+pU8zuKmwX35gg9FKqTezv/B8o:t/CEkuOQpKEM3R+pURKrmGwTB3h
Compare: 72
mal@works:~$
```

Evaluation of impfuzzy

This section describes the results of an experiment of comparing, and evaluating, how the 3 different methods (proposed impfuzzy, imphash and ssdeep) are capable in classifying similar types of malware.

For the experiment, 10 different samples for 20 types of malware (200 samples in total) were prepared. For all the combinations that select two samples out of the 200, the similarities of the samples were calculated using the three methods. The two samples were classified as the same if the calculated value was 30 and larger.

For the experiment, packed samples were unpacked beforehand.

Figure 1 shows the results of the experiment.

There were no false positives detected which classified different types of malware as the same.

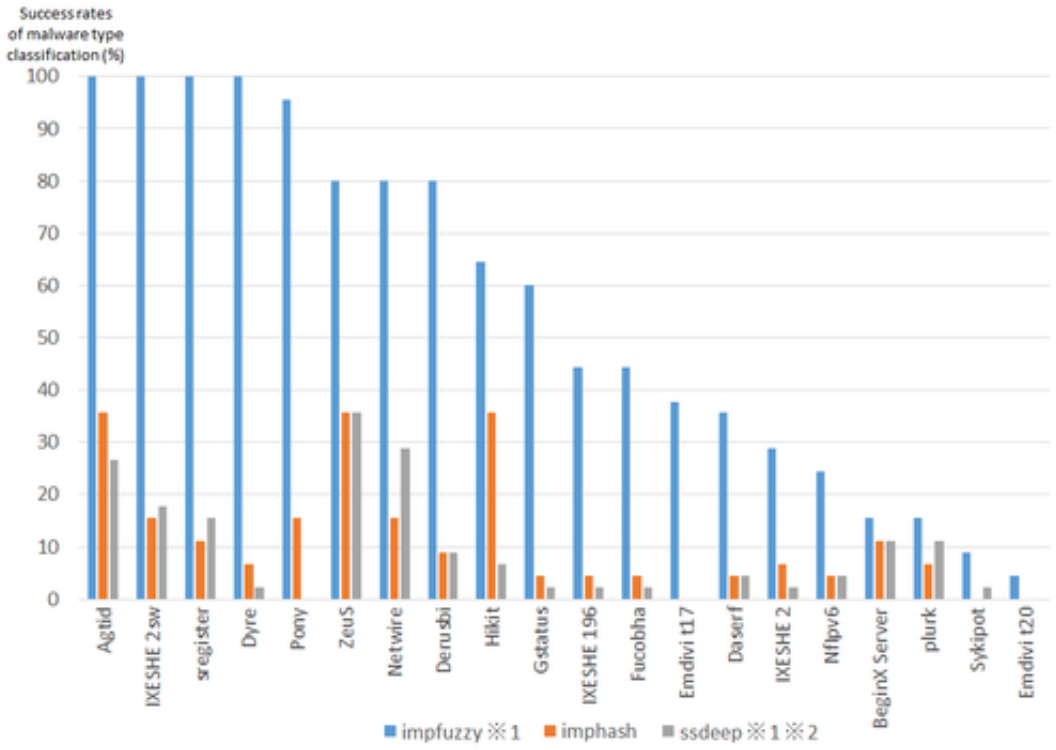

Figure 1: Success rates of malware type classification among impfuzzy, imphash and ssdeep

*1 The threshold value for classifying the same malware type is set as 30, for impfuzzy and ssdeep
*2 ssdeep's target of comparison is the whole executable file

The results reveal that impfuzzy is more capable of classifying the same malware type than the other two methods. For the details of the results, please see Appendix A.

This method judges the samples' similarities based on Import API, and therefore it is clear that the identification rates are higher for samples whose malware generation tools (builders) are publicly available (e.g. Pony and ZeuS), because their Windows API rarely change. On the other hand, samples whose functions are frequently updated or changed (e.g. Emdivi) have lower rates.

For the experiment, the threshold value for impfuzzy was set as 30, however, it may detect false positives depending on the executable files. We recommend adjusting the threshold value for each executable file.

In case impfuzzy is not applied effectively

Some malware have few Import APIs.

For example, some samples called downloader use only around 5 Windows APIs. For those samples, impfuzzy may not be able to compare the hash values accurately due to the limited size of data to compare. Also, samples generated by .NET Framework have different mechanisms from applications which directly execute Windows API. Therefore, hash values for those samples cannot be calculated.

Summary

In the current situation where huge numbers of malware samples emerge day by day, it is important to efficiently classify malware samples, and identify new ones which need to be analysed. We believe that the method introduced here will be one approach to support this process.

Also, Volatility Plugin that we released together with impfuzzy, is useful for memory forensics to check malware infection.

Since this tool can calculate hash values of samples which are unpacked in the memory, it is capable of judging the similarities of the samples, even if the malware is packed.

This Plugin is available at the following URL. We will introduce this tool on this blog at another occasion.

> JPCERTCC/aa-tools GitHub – impfuzzy for Volatility
> https://github.com/JPCERTCC/aa-tools/tree/master/impfuzzy/impfuzzy_for_Volatility

Thanks for reading.

- Shusei Tomonaga

(Translated by Yukako Uchida)

Reference

[1] FireEye - Tracking Malware with Import Hashing
https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html
[2] Fuzzy Hashing and ssdeep
http://ssdeep.sourceforge.net/

Appendix A

Comparison of impfuzzy, imphash and ssdeep

Table 1: Malware Identification Rate

Type	impfuzzy (%)	imphash (%)	ssdeep (%)
Agtid	100	35.6	26.7
BeginX Server	15.6	11.1	11.1
IXESHE 196	44.4	4.4	2.2
IXESHE 2	28.9	6.7	2.2
IXESHE 2sw	100	15.6	17.8
Daserf	35.6	4.4	4.4
Dyre	100	6.7	2.2
Fucobha	44.4	4.4	2.2

Gstatus	60	4.4	2.2
Hikit	64.4	35.6	6.7
Netwire	80	15.6	28.9
NfIpv6	24.4	4.4	4.4
Emdivi t17	37.8	0	0
Emdivi t20	4.4	0	0
plurk	15.6	6.7	11.1
Derusbi	80	8.9	8.9
Pony	95.6	15.6	0
sregister	100	11.1	15.6
Sykipot	8.9	0	2.2
ZeuS	80	35.6	35.6

*Types of malware used in this experiment are those analysed and classified by JPCERT/CC.