Poster: Deep Learning for Zero-day Flash Malware Detection

Wookhyun Jung
Researcher
Cyber Security Research Center
KAIST
Email: pplan5872@kaist.ac.kr

Sangwon Kim
Researcher
Cyber Security Research Center
KAIST
Email: bestksw@kaist.ac.kr

Sangyong Choi
Researcher
Cyber Security Research Center
KAIST
Email: csyong95@kaist.ac.kr

I. Introduction

Adobe Flash is a platform independent multimedia format which is widely used for advanced interactive content in the shape of the file called "SWF" from web, office applications, etc. Adobe have announced that Adobe Flash was installed over 500 million times in the second half of 2013 [1].

However, its popularity makes it a hot target for malware attacks. Adobe Flash vulnerability statistics, provided by The Common Vulnerabilities and Exosures (CVE) system [2], have shown a gradual increase as shown in Figure 1. Moreover, 81% of Adobe Flash vulnerabilities have received 9+/10 score, which means that Adobe Flash malware attacks are one of the most serious threat. In 2014, Adobe Flash vulnerability CVE-2014-0515 was used by a surveillence malware known as "Casper" in the wild [3].

In order to detect Flash malware using machine learning, previous work [4], [5] have been proposed so far. This work use manually predefined malicious features (e.g. shell-code, environment variables, a vulnerability-specific Tag). Figure 2 shows the position of predefined malicious features. When Flash malware uses the predefined features, the previous work is effective, but if new Flash malware uses the Malicious feature set A (named as zero-day Flash malwares) instead of predefined features, it could evade the previous work. Also, we take into account the benign features, since it indicates normal, so the accuracy could be increased. Notice that all features are able to be turned into benign or malicious over a period of time, so decision boundary could be changed.

Therefore, we have to use as many features of SWF as we can. To handle large and complex features, deep learning (which is state-of-art machine learning technique and shows accurate image classification results [7]) is employed to our system. Especially, the abstraction ability of deep learning makes it possible to abstract benign or malicious features among the features.

In this paper, for all of SWF features to apply deep learning, we 1) extract all of the features from both static analysis and dynamic analysis (such as API call sequence), and then 2) convert it to numeric, finally 3) apply the deep learning technique to classify the samples. Therefore, we provide the following main contributions:

(i) We present a novel method, based on deep learning, for detecting zero-day Flash malwares through all of the features SWFs have, which enables to abstract benign or malicious features, but the previous work use only predefined malicious features.
(ii) Therefore, our detection system can keep track of zero-day Flash malwares. In other words, it can learn new features of zero-day Flash malwares, but the previous work couldn't do.
(iii) We evaluate our detection system through Flash malwares collected in the wild.

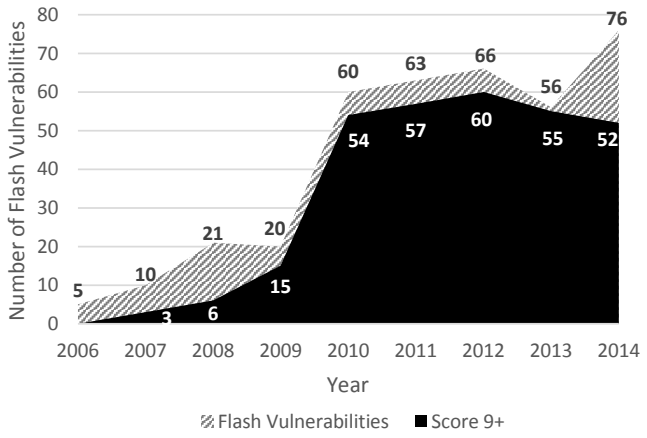

Fig. 1. The number of Adobe Flash vulnerabilities: It shows that the number of Adobe Flash vulnerabilities is increasing, and the percent of 9+/10 score vulnerabilities is 81%.

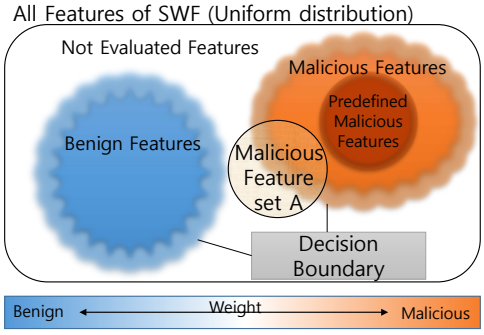

Fig. 2. A benn diagram for benign, malicious and not evaluated features: Predefined Malicious features are not enough to detect the Malicious feature set A.

II. Design

Our design goal is to use deep learning to accurately classify zero-day Flash malwares, so all of the features SWFs have need to be converted to numeric to preserve as possible as we can. Our framework has three phases, 1) Extraction, 2) Conversion, and 3) Classification as shown in Figure 3.

The Extraction phase has two parts: Static analysis module extracts a Header, Tags and Action bytecode by parsing, and Dynamic analysis module generates API calls by executing the given SWF files. The static analyzed data is divided into raw values and a sequence of instructions. The dynamic analyzed data is composed of a sequence of API calls.

The Conversion phase converts from the data generated from the Extraction phase to a scaled value 0∼1 or matrix . If raw values are given, it is scaled to between 0 and 1 to reduce computational complexity. If the sequence of instructions or API calls are given, it is projected to a N*M matrix, where N is the number of sequences and M is the number of types. For example, if a sequence of 5 instructions "ADD, DEL, ADD, MOV, DEL", and 3 types of instruction 'ADD', 'DEL', 'MOV' is given, N is 5 and M is 3. In this case, if we assume 'ADD':[1 0 0], 'DEL':[0 1 0] and 'MOV':[0 0 1], we can make a 5*3 matrix [[1 0 0], [0 1 0], [1 0 0], [0 0 1], [0 1 0]].

The Classification phase trains the deep learning models or classifies the given SWF through the data generated from the Conversion phase. If the scaled values are given, Deep Feed-forward Neural Network will be used. If the matrix is given, Deep Recurrent Neural Network will be used. In other words, the scaled values of a Header and Tags will be trained or classified using Deep Feed-forward Neural Network, and the projected matrices of Action bytecode and API calls will be trained or classified using Deep Recurrent Neural Network which is able to train or classify the sequence of the data. At last, Ensemble method will combine all of the result generated from the Deep learning models.

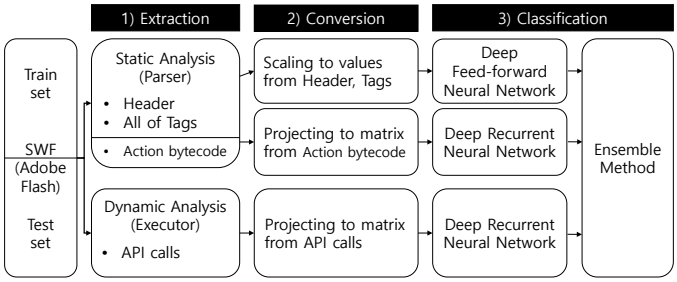

Fig. 3. Deep learning based detection system for zero-day Flash malwares

III. Preliminary Result

We implemented a prototype of the Static Analysis module (which makes boolean depending on the presence of Tags. For example, if the Tag is present, the boolean is 1, and if the Tag is not present, the boolean is 0.) and then apply to Deep Feed-forward Neural Network (1424 input nodes, 2000 nodes * 3 hidden layers, 2 output nodes, DropConnect p=0.5, L2-regression=0.001). We have collected a total of 666 SWFs: 333 malicious set from VirusTotal Intelligence, 333 benign set from Google search following the our approach (iii).

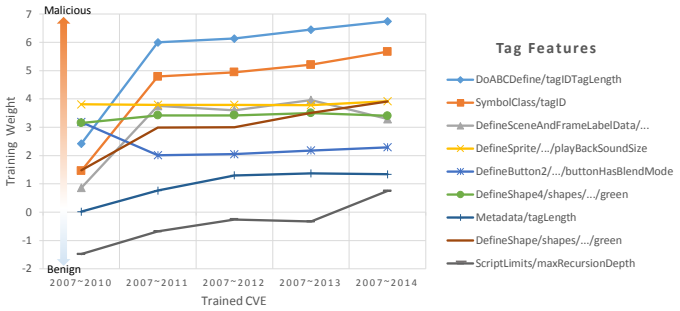

Fig. 4. Weight results of top-ranked malicious Tag according to the CVE based evaluation

Figure 4 shows what features are benign or malicious, which validates our approach (i). Note that "ScriptLimits" feature is benign when 2007∼2010 train set is trained, but "ScriptLimits" feature is malicious when 2007∼2014 train set is trained, which validates our approach (ii).

TABLE I. Detection Accuracy according to the CVE based evaluation

Train	Test	Accuracy
2007∼2010	2011∼2015	51.77%
2007∼2011	2012∼2015	97.75%
2007∼2012	2013∼2015	97.78%
2007∼2013	2014∼2015	98.33%
2007∼2014	2015	100%

Table I shows that our detection system has high accuracy. We plan to implement the other modules, which will improve accuracy and enable to abstract interesting features from Action bytecodes. Therefore, we expect that our detection system could detect zero-day Flash malwares more accurately.

References

[1] Adobe, "Statistics — Adobe Flash runtimes," http://www.adobe.com/kr/products/flashruntimes/statistics.html, 2015, [Online; accessed 31-March-2015].

[2] C. Details, "Adobe Flash Player : CVE security vulnerabilities, versions and detailed reports," http://www.cvedetails.com/product/6761/Adobe-Flash-Player.html, 2015, [Online; accessed 31-March-2015].

[3] ESET, "Syria Targeted by Cartoon Malware," http://www.eset.com/int/about/press/articles/malware/article/syria-targeted-by-cartoon-malware/, 2015, [Online; accessed 31-March-2015].

[4] T. Van Overveldt, C. Kruegel, and G. Vigna, "Flashdetect: Actionscript 3 malware detection," in *Research in Attacks, Intrusions, and Defenses*. Springer, 2012, pp. 274–293.

[5] S. Ford, M. Cova, C. Kruegel, and G. Vigna, "Analyzing and detecting malicious flash advertisements," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE, 2009, pp. 363–372.

[6] S. Van Acker, N. Nikiforakis, L. Desmet, W. Joosen, and F. Piessens, "Flashover: Automated discovery of cross-site scripting vulnerabilities in rich internet applications," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 12–13.

[7] R. Benenson, "Classification datasets results," http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html, 2014, [Online; accessed 31-March-2015].