On the Feasibility of Automatic Malware Family Signature Generation

Xiao Zhang
Palo Alto Networks Inc
Santa Clara, CA
xizhang@paloaltonetworks.com

Zhi Xu
Palo Alto Networks Inc
Santa Clara, CA
zxu@paloaltonetworks.com

ABSTRACT

Malware detection has witnessed a rapid transition from manual signature release to fully automation in recent years. In particular, with the accumulation of huge malware sample sets, machine learning (ML) and deep learning (DL) have been proposed for verdict predicting and family attribution. Despite the high accuracy and efficiency, existing proposals fall short in providing explanation for their detection results. To fill in the gap between classification decisions and reasoning behind, we propose Galaxy, a generic approach for automatic malware family signature generation. Briefly, Galaxy selects meaningful metadata fields from static and dynamic analysis reports of the given samples. Based on the selected fields, all input samples will be clustered into groups according to similarity measurement. The observed similarities will then be converted into patterns and validated against multiple intelligence sources to decide whether it is suitable for malware detection. In the end, Galaxy launches a refine process to improve the grouping results and increase sample coverage. We have applied the Galaxy framework on daily incoming Android samples to our WildFire production since September 2016. Up to know, Galaxy has generated more than 12,500 unique family signatures covering a total of 1.75 million Android malwares. Those family signatures have provided valuable insights for the discovery of undocumented malicious domains and identification of Communication & Control (C&C) servers. Because of our rigid quality requirement, all released signatures have been proven to cause no false positives in production.

KEYWORDS

Galaxy, Malware Categorization, Malware Detection, Clustering, Machine Learning, Intelligence, Android

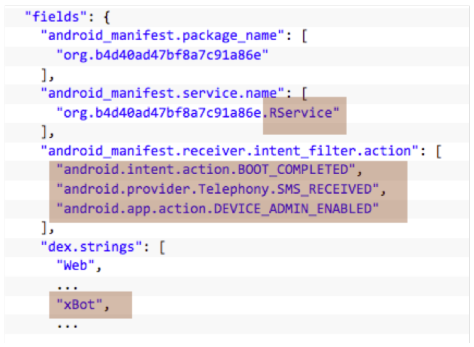
1 INTRODUCTION

Malware detection techniques have evolved from the traditional signature-based method, to behavior-based and heuristic-based approaches. In particular, signature-based detection is the most commonly used method in commercial antivirus products, but it can only be applied to completely known threat cases. Behavior-based approach fills in the gap by observing behavior of a program to conclude whether it is malicious or not [2]. Thus, a single behavior signature can identify various samples of malwares. However, behavior-based malware detection systems usually suffer from high false positive rate and excessive amount of scanning time. To overcome these disadvantages, heuristic-based malware detection methods embrace data mining and machine learning techniques. They learn the behaviors of large set of malware and benign samples, based on which the classification decision can be made for future incoming samples. Because of the dramatically increased sample set and computing power, heuristic-based malware detection techniques have witnessed a rapid advancement alongside the big data wave.

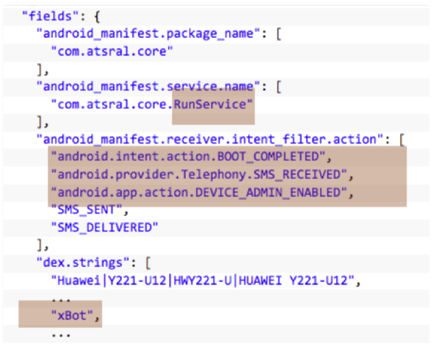
State-of-art application of machine learning techniques have been quickly adopted in research work [3–5] and industry systems [1, 6]. All proposed systems are superior in accuracy, performance and efficiency. However, neither of them can provide meaningful explanations beyond the final classification decision. Knowing that one sample is malicious and belongs to a specific malware family is only the first step in malware detection. What's more intriguing is the underlying logic of that decision. Such logics, once revealed, would naturally become family signatures and be easily turned into lightweight detection rules. Depending on different forms of intel included, they could also provide indications of compromise (IOCs) from a variety of perspectives. However, digging of such hidden gold is extremely challenging due to the data processing inequality, i.e., all context information has been gradually lost during the process of data training in ML or DL.

To fill in the gap between classification decisions and reasoning behind, we propose Galaxy, a generic approach for automatic malware family signature generation. In this part, we'll present the high-level architecture of Galaxy and the essential building modules. A total of four modules will be presented, including metadata generation, sample grouping, pattern extraction and evaluation, and finally group refinement. Briefly, Galaxy selects meaningful metadata fields from static and dynamic analysis reports of the given samples. Based on the selected fields, all input samples will be clustered into groups according to similarity measurement. In this step, the actual metadata information is lost. Galaxy recovers the lost context information by extracting similarities within each group of samples. The observed similarities will then be turned into patterns and further evaluated to see if they can properly represent samples in the group. If so, the pattern will be validated against multiple intelligence sources to decide whether it is suitable for malware detection. In the end, Galaxy launches a refine process to improve the grouping results and increase sample coverage.

To demonstrate the effectiveness and applicability of Galaxy in real-world malware detection system, we have deployed Galaxy on Android APK files within our WildFire production environment. The entire deployment process has leveraged on the APK engine in WildFire for sample analysis, K-means algorithm for sample clustering and VirusTotal for pattern validation. The Galaxy Android system has been running constantly on daily incoming samples to our WildFire production since September 2016. We further combine leftover samples in each day and launch the refinement task on a weekly and monthly basis. Up to know, Galaxy has generated more than 12,500 unique family signatures covering a total of 1.75



```
"fields": {
  "android_manifest.package_name": [
    "org.b4d40ad47bf8a7c91a86e"
  ],
  "android_manifest.service.name": [
    "org.b4d40ad47bf8a7c91a86e.RService"
  ],
  "android_manifest.receiver.intent_filter.action": [
    "android.intent.action.BOOT_COMPLETED",
    "android.provider.Telephony.SMS_RECEIVED",
    "android.app.action.DEVICE_ADMIN_ENABLED"
  ],
  "dex.strings": [
    "Web",
    ...
    "xBot",
    ...
```



```
"fields": {
  "android_manifest.package_name": [
    "com.atsral.core"
  ],
  "android_manifest.service.name": [
    "com.atsral.core.RunService"
  ],
  "android_manifest.receiver.intent_filter.action": [
    "android.intent.action.BOOT_COMPLETED",
    "android.provider.Telephony.SMS_RECEIVED",
    "android.app.action.DEVICE_ADMIN_ENABLED",
    "SMS_SENT",
    "SMS_DELIVERED"
  ],
  "dex.strings": [
    "Huawei|Y221-U12|HWY221-U|HUAWEI Y221-U12",
    ...
    "xBot",
    ...
```

Figure 1: The inspiration of Galaxy

million Android malwares. All those family signatures have been enforced in our WildFire production and consistently capturing new malware samples. Because of our rigid quality requirement, all released signatures have been proven to cause no false positives in production. It should be noted that, Galaxy itself is not bound to any particular analysis engine, platform, clustering algorithm or intelligence source. The trial-and-error experience we learned during the deployment of Galaxy on Android will provide valuable insights for other platforms.

2 DESIGN AND IMPLEMENTATION

The inspiration of Galaxy comes from our own experience with sample analysis. During the manual inspection, we always uncover similar strings belonging to samples within the same malware family. Fig 1 shows such an example. Those two samples share a great deal of similar strings in their service name, intent filter actions, dex strings and receiver names. All these similarities are actually unique characteristics of the malware family they belong to. Clearly, manual analysis will always fall behind the increasing need of malware family discovery. Thus, an automatic approach is in demand.

An overview of Galaxy's architecture is depiced in Fig 2. It takes a set of target samples as input. At its core, Galaxy consists of four major components: metadata generation, sample grouping, pattern extraction and signature validation. Within each round of Galaxy core execution, it is possible that we cannot generate valid patterns that cover certain malware samples. Thus, it is necessary to repeat the process multiple times. In each round, we refine the input samples set by removing the ones that have been covered by high-quality patterns. With less noise in each round of execution, some vague similarities may now stand out, leading to high-quality patterns. We repeat such process until a reasonable coverage rate threshold is achieved. In this section, we'll use Galaxy on Android to illustrate the design and implementation details of each core component.

Metadata Generation. In this step, we perform static and dynamic analysis on the given samples. Such analysis can be done by sample analysis platforms, like WildFire. Then, we extract features and information from the analysis reports. Each sample file will have a metadata file associated with it. Take APK file for example, a non-exhaustive list of fields we have considered in the metadata generation includes package name, app name, activity names, service names, broadcast receiver names, certificate signer and owner, suspicious behaviors, suspicious API calls, suspicious URLs and so on. At the end, we perform a cleaning procedure to remove the noise that might affect the upcoming similarity measurements. All generated metadata are indexed in ElasticSearch for efficient data retrieval and query.

Sample Grouping. With metadata, we cluster samples based on their similarities. For similarity measurement, the algorithms that can be applied include AV Tokenization algorithm, TF/IDF score generation algorithm. For clustering, the algorithms that can be applied include multiple K-means clustering algorithms and deep clustering algorithm. All algorithms run on distributed servers. Here, we propose to treat the sample metadata as text files with different paragraphs and sections. In this way, we convert the problem of similarity measurement on metadata object to text similarity measurement problem. Thus, many text similarity measurement tools and approaches can be applied in our case to find the similarities among a group of samples. The result of sample grouping step will be a set of sample groups. Each group contains a set of samples and a description of similarities uniquely shared by samples within this group.

Pattern Extraction. In this step we evaluate the quality of grouping results from two perspectives: the description of similarities (i.e. the pattern shared by all samples in the group), and the set of samples. The similarity description consists of a combination of conditions, e.g., package name starts with certain prefix string, certificate signer matches with the provided one, activity name is present in app's components, etc. Each similarity is further formalized as a pattern, which is defined below:

$$pattern = (field, operator, value)$$

In particular, field could be package name, certificate signer and other fields we have selected. In terms of operator, we currently support three types: exact match, prefix match and post match. Each of them corresponds to the location of the similarities occurred. At last, the value is the actual content of the similarities which are shared by a group of samples.

Signature Validation. A group of patterns can be combined togehter to form a malware signature. We apply a set of heuristics to

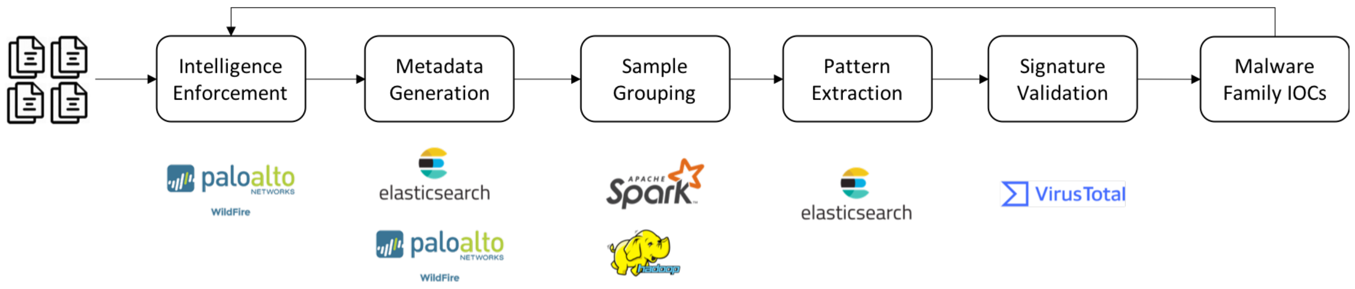


Figure 2: Galaxy Architecture Overview

```
{
    "operator": "all",
    "children": [
        {
            "operator": "contains",
            "field": "sample.tasks.apk_embeded_url",
            "value": "http://xxxx/97ss_source/hy/41.jpg ,
assets/data"
        },
        …
        {
            "operator": "contains",
            "field": "sample.tasks.connection",
            "value": "udp , xxx.xxx.0.252:5355 , -"
         },
         …
        {
            "operator": "contains",
            "field": "sample.tasks.apk_defined_receiver",
            "value": "com.xxxxxxx.HuntReceive"
        },
]
}
```

Figure 3: Malware Signature Example from Galaxy

evaluate the quality of observed similarities. For example, we consider the similarity as high quality only if it can uniquely identify all samples in the cluster group. For samples in each cluster group, we further validate their similarities from other aspects, including the distribution of antivirus verdicts and suggested family names from WildFire/VirusTotal/others. We check whether the samples in this group are considered as from the same malware family by different antivirus vendors. If so, we conclude that the group has a higher possibility to truly belong to the same malware family.

3 EVALUATION

In this section, we first evaluate the effectiveness of Galaxy against Android sample set. Then, we will use preventative cases to show demonstrate other potential benefits of Galaxy.

Effectiveness. The Galaxy Android system has been running constantly on daily incoming samples to our WildFire production since September 2016. We further combine leftover samples in each day and launch the refinement task on a weekly and monthly basis. Up to know, Galaxy has generated more than 12,500 unique family signatures covering a total of 1.75 million Android malwares. All those family signatures have been enforced in our WildFire production and consistently capturing new malware samples. Because of our rigid quality requirement, all released signatures have been proven to cause no false positives in production.

C&C Server Identification. Each family signature generated from Galaxy can be viewed as a behavior profiling for a malware family. Naturally, it consists of important characteristics about this malware family. One notable example is the Commucation & Control (C&C) server identification. C&C server is a natural way to correlate malware samples together to a family. In the signature example from Fig 3, all samples in this malware family contains a special url, i.e., "http://xxxx/97ss_source/hy/41.jpg", in their "assets/data" file. In addition, they all talk to a IP address of xxx.xxx.0.252 at port 5355. Without Galaxy, acquiring such information would require malware experts' manual analysis on a bunch of samples in this family. Then, experienced malware analysts may be able to come up with a partial description of this family. It is far less efficient and comprehensive compared to the automatic approach employed in Galaxy.

4 CONCLUSION

Direct application of machine learning techniques in malware detection fails to provide meaningful explanations beyond the final classification decision. Thus, malware family discovery and sample correlation is still very labor-intensive work and greatly depends on malware researchers' expertise. With the proposed Galaxy framework, we can achieve highly automation in malware family discovery and sample correlation. Galaxy is capable of extracting, understanding and explaining the similarities among samples that are discovered by ML algorithms. More importantly, it can automatically detect and verify the malicious indicators from the discovered similarities. Galaxy on Android has been demonstrated to be effectiveon a large sample set. The automatically generated intelligence also has a wild range of real-world applications in malware analysis.

5 ACKNOWLEDGEMENT

We would like to thank the colleagues at Palo Alto Networks for their valuable comments. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Palo Alto Networks.

REFERENCES

[1] Konstantin Berlin. 2016. An AI Approach to Malware Similarity Analysis: Mapping the Malware Genome With a Deep Neural Network. *Black Hat USA* (2016).

[2] KALPA 2017. Introduction to Malware. (2017). Retrieved Jan 27, 2018 from http://securityresearch.in/index.php/projects/malware_lab/introduction-to-malware/8/

[3] Munam Ali Shah Qudsia Jamil. 2016. Analysis of machine learning solutions to detect malware in android. In *Innovative Computing Technology (INTECH)*. 226–232.

[4] J. Sahs and L. Khan. 2012. A Machine Learning Approach to Android Malware Detection. In *2012 European Intelligence and Security Informatics Conference*. Odense, 141–147.

[5] Igor Muttik Suleiman Y. Yerima, Sakir Sezer. 2014. Android Malware Detection Using Parallel Machine Learning Classifiers. In *Next Generation Mobile Apps Services and Technologies (NGMAST)*. 37–42.

[6] Lei Want. 2016. AI Based Antivirus: Detecting Android Malware Variants With a Deep Learning System. *Black Hat Europe* (2016).