Mobile Malware Visual Analytics and Similarities of Attack Toolkits

RiskSense Technical White Paper Series

Written by Anand Paturi, Manoj Cherukuri, John Donahue, and Dr. Srinivas Mukkamala

Executive Summary

This document is part of the RiskSense Technical White Paper Series, which provides practical advice from a variety of RiskSense cyber security experts to assist security analysts in their day-to-day operations. This paper discusses the use of Normalized Compression Distance (NCD) - based on its capabilities to perform similarity measure of unstructured data - to enumerate code similarity between malicious Android applications and visualize their clusters. The illustrated classification methods and visual analytics can help the anti-virus community to ensure that a variant of a known malware can still be detected without the need of creating a signature. This paper also showcases that the proposed methods can be used to understand the similarity / behavior with known malware families when a new malware is released.

About RiskSense

RiskSense®, Inc., is the pioneer and market leader in pro-active cyber risk management. The company enables enterprises and governments to reveal cyber risk, quickly orchestrate remediation, and monitor the results. This is done by unifying and contextualizing internal security intelligence, external threat data, and business criticality across a growing attack surface.

The company's Software-as-a-Service (SaaS) platform transforms cyber risk management into a more pro-active, collaborative, and real-time discipline. The RiskSense Platform™ embodies the expertise and intimate knowledge gained from real world experience in defending critical networks from the world's most dangerous cyber adversaries. As part of a team that collaborated with the U.S. Department of Defense and U.S. Intelligence Community, RiskSense founders developed Computational Analysis of Cyber Terrorism against the U.S. (CACTUS), Support Vectors Intrusion Detection, Behavior Risk Analysis of Vicious Executables (BRAVE), and the Strike Team Program.

By leveraging RiskSense cyber risk management solutions, organizations can significantly shorten time-to-remediation, increase operational efficiency, strengthen their security programs, heighten response readiness, reduce costs, and ultimately minimize cyber risks. For more information, please visit www.risksense.com or follow us on Twitter at @RiskSense.

Table of Contents

1.0 Introduction

Malicious applications installed on smart phones target them the same way as traditional malware would target personal computers since smart phones provide nowadays the same capabilities as handheld personal computers. A recent malware threat report from Intel Security McAfee [1] indicates that malicious applications increase by 100,000 samples a day and target smart phones, employing attack vectors that are the same as x86 and x64 malware. Most of the malicious mobile applications in applications markets are the result of repacking, which is a very common technique used by adversaries whereby a legitimate application is modified by injecting malicious code. Under this scenario, benign mobile applications are disassembled, embedded with malware, re-assembled, and then re-packaged to look like benign applications. Finally, the embedded payload is launched when the application is activated.

In prior reports, security researchers from the North Carolina State University observed that much of Android malware is repackaging other legitimate (popular) applications. After analyzing more than 1,200 Android malware samples, they found that 86 percent repackaged legitimate applications to include malicious payloads [2]. In their recent experiments on Android malware, they have observed that the detection rates of well-known anti-virus engines range from 51.02 percent to 100 percent, while the detection rate of the Google App Verification Service in Android 4.2 (JellyBean) is 20.41% [3].

Authors in [4] have implemented an application similarity measurement called DroidMOSS (using fuzzy hashing technique) to effectively detect repackaged applications. Authors first perform feature extraction, which includes extracting of instructions in the application and author information. They extract opcodes from the Dalvik bytecode for instructions and META-INF subdirectory for author information of the application. Next finger print generation is performed, in which a sliding window hashing technique is used, whereby a sliding window starts from the very beginning of the instruction sequence and moves forward until its rolling hashing value equals a pre-selected reset point, which determines the boundary of the current piece. This hashing technique is implemented on all instructions of the application, starting from the first instruction a hash value is computed for $(i + 1)^{th}$ instruction to $(j, (j +1)^{th})$ instruction to n^{th} instruction. Additionally, a hash value is calculated for the entire instruction sequence. The primary reason to calculate window hashes is to find changes that could occur in a subset of instructions, which may be either adding or deleting instructions. Finally, authors perform similarity scoring to compute edit distance between two finger prints. This approach does not detect the malicious payload at source code level in the repackaged application and suffers from extracting features to perform the whole process.

Authors in [5] propose a scalable infrastructure for code similarity analysis among Android applications. Initially, authors perform application pre-processing to represent it in a common XML-based format and then perform feature extraction using feature hashing. Finally, similarity measurement is performed to enumerate the similarity measure and extract the similar code. This methodology incurs the complexity of feature extraction, which might make it infeasible for quick analysis of repackaged applications over a large dataset.

The approach that RiskSense cyber security experts propose, does not suffer from feature extraction, since they do not implement feature extraction to detect similarity in applications but rather use NCD for obtaining quick measurement of similarity analysis and then detect the degree of code similarity. In RiskSense's current experiments for Mobile malware analysis (focusing on *DroidKungFu* family), we calculate NCD between applications to measure their similarity and represent results in a distance matrix. We apply a hierarchical clustering to group malware; the clusters are represented using a Pythagoras tree fractal. In our fractal visualization, all samples from the distance matrix are considered as a

single square in the first step and form there on, two squares are constructed with each scaled down by a linear factor of *½ v2*.

1.1 Web Malware

With perimeter defenses getting stronger over time, the attackers have targeted the Web for distributing malware. Researchers have conducted several studies to understand the life cycle of Web malware [6][7][8]. Google had observed that about 1.3 percent of the search queries result in a malicious link [9] and more than 60 percent of the Web attacks happen through the attack kits [10]. Detection of shellcode for detecting the drive-by download attacks is one of the more focused approaches for preventing Web-based malware attacks. Previous researchers have published several approaches relying on heuristics and emulation for the detection of shellcodes [11] [12] [13].

In our initial approaches we used Cosine measure to calculate the similarity between malware samples (Financial Crimeware and Attack Toolkits: Eleonore is similar to Fragus, IEKit, JustExploit, MyPloySploits, Neon, ExploitPack, and ZeroExploit). Cosine Similarity is measured as the cosine of the angle between the two vectors (API sequence of malware samples).

Our analysis shows that the shellcodes used as payloads, across different attack kits, were nearly similar with scores over 90%. We observe that some of the attack kits released across different years, had the same shellcodes.

With respect to Mobile Malware (DroidKungFu family) we can classify all the samples correctly and represent clusters (malware families and benign applications) using a Pythagoras tree fractal.

Based on the results, similarity measures (Cosine Measure and Normalized Compression Distance) can be an effective static mechanism to detect malware variants, shellcode based drive-by download attacks, attack toolkits, mobile malware, and repacked mobiles applications with malware.

2.0 Mobile Malware Analysis: Detecting Malcode In Repackaged Applications

In our methodology, mobile applications collected from multiple application stores are passed through a pre-processing phase, in which we covert each mobile application in to a pre-defined format. Later we use gzip, bzip2, and rsynccompressors to compress the pre-processed files (by defining a compression block size) and then calculate the NCD between them. The output of this method is a distance matrix based upon we visualize clusters of mobile applications, which shows the similarity between mobile applications based on their source code similarity. Using NCD enables us to perform similarity analysis without prior domain knowledge.

We chose DroidKungFu family for our experiments as a proof-of-concept for a couple of reasons: Google App Verify has the lowest detection rate or had failed in a few instances to detect the DroidKungFu malware and the overall anti-virus detection rate on the DroidKungFu family is close to 50 percent. With an exception of five anti-virus tools all others failed to detect DroidKungFu4Sap, while only four anti-virus tools detected DroidKungFu4Update; details of the analysis can be found in the

latest report: An Evaluation of the Application ("App") Verification Service in Android 4.2 (JellyBean). Once installed and successfully executed this malware can collect device information (IMEI number, phone model, and OS version), performs privilege escalation to gain admin / root access, installs backdoors and remote access Trojans, as well as can collect and send out sensitive data. The latest versions of Android and properly patched phones may not be susceptible to this malware.

2.1 Methodology

All high-risk applications resulting from our permission analysis are subjected to this analysis to detect if they have been re-packaged from a benign application, software piracy, and their similarity in code to existing malware. Figure 1 represents our pre-processing phase, in which .apk files are converted to our custom XML-based file format.

We calculate NCD (derived from Kolmogorov complexity theory) - between applications (essentially their XML output) - to measure their similarity and represent results in a distance matrix. The AndroGaurd team also presented several similarity measures for detecting mobile malware [14].

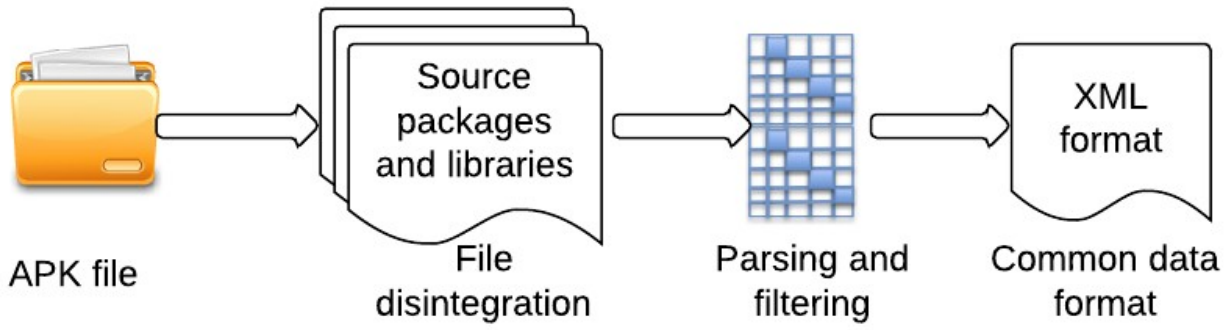

FIGURE 1: File pre-processing process

Formally NCD between two strings x and y is defined as:

$$NCD(x,y) = \frac{\max\{K(x \mid y), K(y \mid x)\}}{\max\{K(x), K(y)\}}$$

K(x|y) is the length of the shortest program that outputs x on input y and K(x) is the length of the shortest program that outputs x on the empty input. An advantage of using NCD is the parameter-free data mining, whereby no features should be extracted from the data set.

Figure 2 represents clustering of malware based on the distance matrix from NCD of DroidKungfu family and benign applications. All applications are scattered, as leaves in the tree, and the cluster of applications that are close to each other on the right side (enclosed in black box) are malicious applications that share similar code. All applications on the left side of the tree are applications that have no resemblance in code with applications from DroidKungfu family.

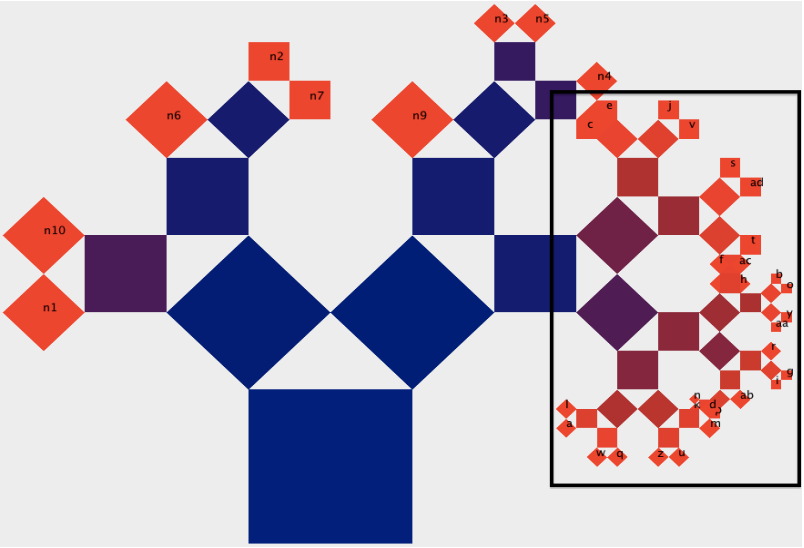

FIGURE 2: Clustering of malware based on NCD

Clusters are represented using a Pythagoras tree fractal. All samples from the distance matrix are considered as a single square in the first step and from there on, two squares are constructed each scaled down by a linear factor of ½ √2. Squares are constructed (recursively) in each iteration with samples that have close NCD between them. Finally, in the n^{th} iteration there are 2^n squares of size $(½ √2)^n$ squares, whereby n is the order of the tree.

3.0 Attack Toolkits

Attack toolkits are bundles of pre-written malicious code for exploiting vulnerabilities to customize, deploy, and automate widespread attacks. A few of these contain various tools for remote access, remote administration, and command-and-control (C&C) administration.

Attack toolkits are significantly advancing the evolution of cyber-crime into a self-sustaining, profitable, and increasingly organized economic model worth millions of dollars. Do-it-yourself (DIY) malware is often pitched as one with low detection rate due to its proprietary nature, and persistence, as very few people will have the knowledge or the tools to defend against such malware, and it would remain undetected for a longer period.

Attack toolkits do one thing very well that every security researcher envies – obfuscation. Most crimeware kits have a capability to generate a new binary file on every use that is radically different from others and conceal the exploit code every time a binary is created; which evades detection from anti-virus or security technologies that rely on signature-based detection.

Influxes of affordable, complex DIY malware toolkits that don't require any coding skills, combined with a new breed of cyber criminals increasingly enable to mine information from social media are making it near impossible for security researchers and practitioners to defend against them.

A recent attack detected by Facebook's security team is one of the most common one used by attack toolkits, whereby a legitimate website is compromised and exploit code is planted. Then, the attacker waits for end users to visit the site with vulnerable browsers to compromise the end system.

In most of the recent documented cases, exploit kits such as Blackhole or Eleonore were used with known vulnerabilities, however a few zero day vulnerabilities are also used like the recent Facebook attack. A few well known attack toolkits circulating around the Internet are Eleonore, MPack, Neosploit, ZeuS, Nukespolit P4ck, and Phoenix [10].

Descriptions of DIY malware toolkits are:

- **Eleonore** represents a very sophisticated and popular attack toolkit, which is used for carrying out drive-by download assaults. It features a model that is subscription-based and modular, which facilitates buyers to acquire the toolkit by paying one price and subsequently less and less fees to acquire future updates, as well as extra exploits.
- **SpyZeuS**: SpyEye and ZeuSTwo teamed up to form a super Trojan to be known as SpyZeuS.
- **DLoader** is a Web-based administration tool that allows Botnet operators to manage the malware that they force the bots under their control to install.
- **Incognito 2.0** is known to have evolved from the original Fragus and has the capability to automatically install different variants of well-known malware families like ZeuS, Fake AV, Gbot, Optima DDOS botnet, Ransomware, and Trojan Downloader. However, it now has a new "cloud-based" administration interface and is offered as exploit-as-a-service model.
- **BOMBA** is designed based on an exploit-as-a-service model and offers its own Web interface and Web management console with authentication.

4.0 Similarity Analysis of Attack Toolkits

No matter how smart and how different DIY malware kits are, most of them share a few common behavioral patterns such as an ability to penetrate browser processes, take screenshots of a victim's machine, or control it remotely; hijacking e-banking sessions, logging them to the level of impersonation, or add additional pages to a website and monitor them; or steal passwords that have been stored by popular programs and use them.

A few advanced features they might possess are:

- Persistence hooks into the operating system;
- Ability to use low-level API (Application Programming Interface) calls to carve out new disk volumes totally hidden from the infected victim;
- Advanced anti-virus by-passing mechanisms; and
- Anti-forensic technology.

RiskSense's proprietary algorithm named Behavioral based Risk Analysis of Vicious Executables® (BRAVE) produces a matrix of similarity scores that can be utilized to determine the likelihood that a piece of code or binary under inspection contains a particular malware or malware variant.

The hypothesis is that all versions of the same malware family or similar malware family share a common core signature that is a combination of several features of the code (binary). After a particular malware has been first identified, it can be analyzed to extract the signature (several quantitative features, semantics, and syntax), which provides a basis for detecting variants and mutants of the same malware or similar ones in the future.

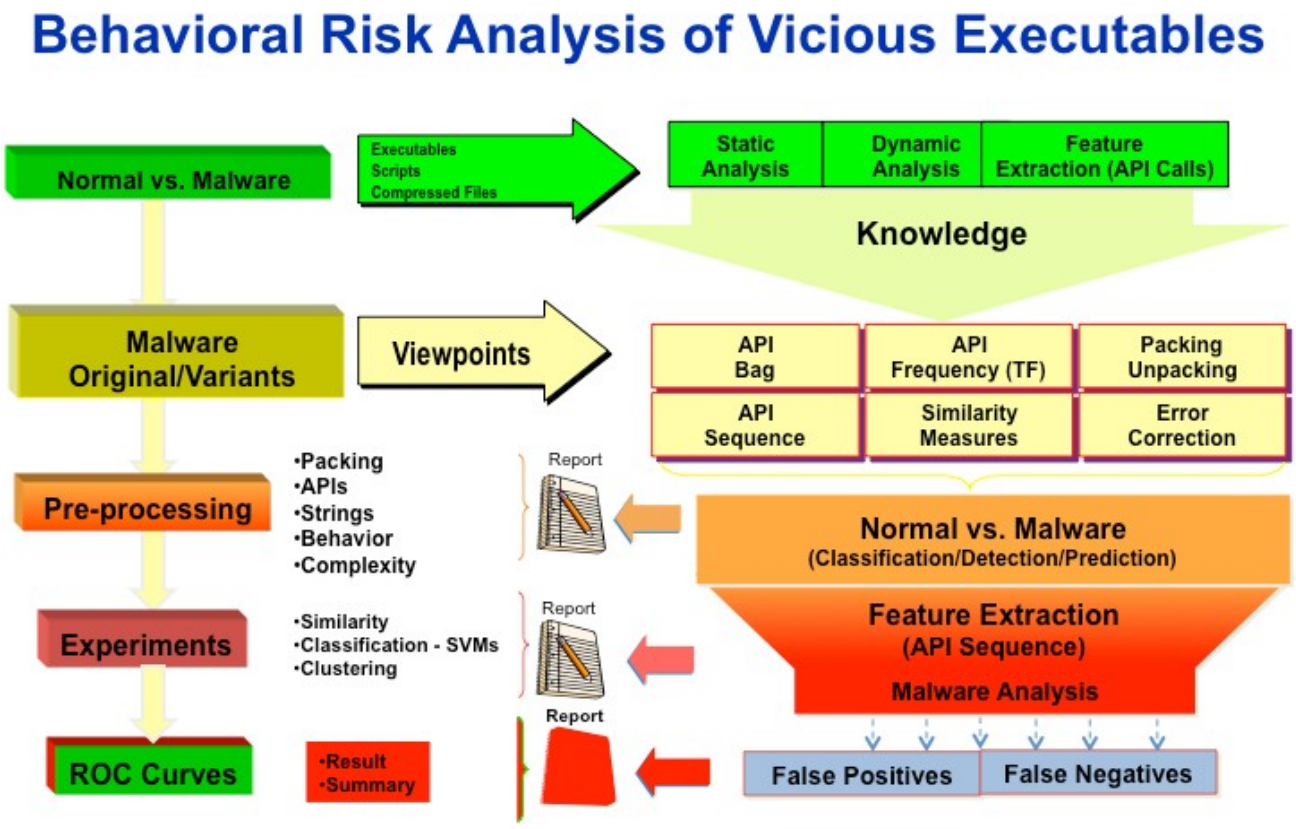

FIGURE 3: Malware analysis and analytics framework - BRAVE

BRAVE functionally classifies malware and malicious code by using well-known computational intelligent techniques and similarity measures that go undetected by traditional security tools and anti-virus scanners. Results from recent experiments illustrate, using the RiskSense proprietary detection algorithm BRAVE on different well-known financial crimeware (represented in Table 1) and attack toolkits (represented in Table 2) shows very high similarity scores (over 70 percent).

Interestingly Zeus variants have high similarity scores with other banking Trojans (Torpig, Bugat, and Clampi) and **SpyZeus dubbed as super Trojan.** The paper presents experimental results that indicate that our proposed techniques can provide a better detection performance against banking Trojans like Zeus crimeware. API sequence was considered as feature vector for banking Trojans. The Cosine measure between different banking Trojans is presented in Table 1.

	Bugat.A	Silentbanker	SpyZeus	Torpig.C	Torpig.E	Trojan.Spy.Z Bot.FT	VUNDO5
Bugat.A	100.00	82.67	68.08	75.12	79.29	64.55	79.91
Silentbanker	47.55	100.00	45.60	61.72	84.19	61.11	64.32
SpyZeus	77.15	75.53	100.00	70.78	73.41	66.73	85.14
Torpig.C	41.01	60.80	43.29	100.00	76.55	50.41	63.14
Torpig.E	77.79	35.47	22.43	71.88	100.00	18.93	63.00
Trojan.Spy.ZBot.FT	43.52	58.47	72.76	60.44	55.85	100.00	78.35
VUNDO5	42.41	70.83	70.71	65.05	84.09	69.45	100.00

Table 1: Similarity analysis of banking Trojans

The opcodes defined for *x86* processor are considered as the set of features for attack toolkits. The frequency of the occurrence of the opcodes in the disassembled code was considered as the feature value. The similarity analysis was performed among the attack toolkits.

Cosine Similarity is measured as the cosine of the angle between the two vectors. Cosine similarity is 1 if the angle between the two vectors is 0 degrees and is 0 if the angle between the two vectors is 90 degrees.

If S', S" are two vectors then,

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} S'_i \times S''_i}{\sqrt{\sum_{i=1}^{n} (S'_i)^2} \times \sqrt{\sum_{i=1}^{n} (S''_i)^2}}$$

We analyzed the shellcodes found in the attack kits to identify the similarity among them. Table 2 shows the similarity measures of the shellcodes in an attack kit with the shellcodes of other attack kits. As presented in our results, Eleonore is similar to Fragus, IEKit, JustExploit, MyPloySploits, Neon, ExploitPack, and ZeroExploit. The attack kits listed in Table 2 were grouped based on the year of their release. The shellcodes remained the same across some attack kits that were released across different years. For example, from our results it is evident that the same shellcodes were used in Armitage, Mpack, FirePack, and Neon attack kits that were released in 2007, 2007, 2008, and 2009 respectively.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Max
1	1.00	0.97	0.24	0.97	0.94	0.24	0.26	0.17	0.19	0.21	0.18	0.97	0.26	0.19	0.19	0.97
2	0.97	1.00	0.25	1.00	0.99	0.25	0.27	0.17	0.20	0.21	0.19	1.00	0.27	0.20	0.20	1.00
3	0.24	0.25	1.00	0.25	0.25	1.00	0.83	0.89	0.90	0.81	0.86	0.25	0.83	0.90	0.90	1.00
4	0.97	1.00	0.25	1.00	0.99	0.25	0.27	0.17	0.20	0.21	0.19	1.00	0.27	0.20	0.20	1.00
5	0.94	0.99	0.25	0.99	1.00	0.25	0.27	0.17	0.20	0.20	0.18	0.99	0.27	0.20	0.20	0.99
6	0.24	0.25	1.00	0.25	0.25	1.00	0.83	0.89	0.90	0.81	0.86	0.25	0.83	0.90	0.90	1.00
7	0.26	0.27	0.83	0.27	0.27	0.83	1.00	0.83	0.85	0.81	0.84	0.27	1.00	0.85	0.85	1.00
8	0.17	0.17	0.89	0.17	0.17	0.89	0.83	1.00	0.99	0.90	0.98	0.17	0.83	0.99	0.99	0.99
9	0.19	0.20	0.90	0.20	0.20	0.90	0.85	0.99	1.00	0.90	0.99	0.20	0.85	1.00	1.00	1.00
10	0.21	0.21	0.81	0.21	0.20	0.81	0.81	0.90	0.90	1.00	0.91	0.21	0.81	0.90	0.90	0.91
11	0.18	0.19	0.86	0.19	0.18	0.86	0.84	0.98	0.99	0.91	1.00	0.19	0.84	0.99	0.99	0.99
12	0.97	1.00	0.25	1.00	0.99	0.25	0.27	0.17	0.20	0.21	0.19	1.00	0.27	0.20	0.20	1.00
13	0.26	0.27	0.83	0.27	0.27	0.83	1.00	0.83	0.85	0.81	0.84	0.27	1.00	0.85	0.85	1.00
14	0.19	0.20	0.90	0.20	0.20	0.90	0.85	0.99	1.00	0.90	0.99	0.20	0.85	1.00	1.00	1.00
15	0.19	0.20	0.90	0.20	0.20	0.90	0.85	0.99	1.00	0.90	0.99	0.20	0.85	1.00	1.00	1.00

1.Armitage	6.Fire Pack	11.JustExploit
2.Cry217	7. Phoenix Exploit-2.x	12.MyPolySploits
3.Icepack	8. Eleonore	13.Neon
4.Mpack-099	9. Fragus	14.ExploitPack
5.El Fiesta	10.IE Kit	15.Zero Exploit

Table 2: Similarity analysis of attack toolkits

5.0 Summary

In this technical white paper, we present similarity measures that can assist the anti-virus community to ensure a variant of a known malware can still be detected without the need of creating a signature; a similarity measure is calculated to produce a matrix of similarity scores that can be utilized to determine the likelihood that a piece of code or binary under inspection belongs to a particular malware family.

Our experiments indicate that all versions of the same malware family or similar malware family share a common core signature that is a combination of several features of the code (binary). Results from our experiments on 40 different variants of Zeus show very high similarity scores (over 85 percent). Interestingly Zeus variants have high similarity scores with other banking Trojans (Torpig, Bugat, and Clampi) and a well know data stealing Trojan *Qakbot*.

Our results show that the shellcodes extracted from the attack kits are similar by 85 percent to at least one shellcode extracted from a different attack kit. The high similarity among the shellcodes of the attack kits released in different years show that there is only a minor variation in the payloads used and the attackers often relied on obfuscation methods using JavaScript to evade the detection mechanisms.

We also used NCD (derived from Kolmogorov complexity theory) - between applications - to measure similarity as a viable method to detect mobile malware. Using NCD we are able to detect all the DroidKungFu malware family (*DroidKungFu 1-4*, *DroidKungFu4Update, DroidKungFu4Sapp*) samples. Using hierarchical clustering we grouped malware families into clusters and represented the clusters using a Pythagoras tree fractal. We envision extending this process to detect similar code fragments that could be potential malicious payloads.

Similarity measures can be used as an effective mechanism to detect mobile malware, malware variants, obfuscated malware, a polymorphic version of known malware, and shellcodes in drive-by downloads arising from the attack kits.

6.0 References

[1] A. Saita. (2012, September). Mobile Malware Is Up - Way Up - in McAfee Q2 Threat Report [Online]. Available: http://www.http://threatpost.com/en_us/blogs/mobile-malware-way- mcafee-q2-threat-report-090412.

[2] Y. Zhou, X, Jiang, "Dissecting Android Malware: Characterization and Evolution," InProceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012), San Francisco, CA, May 2012.

[3] X. Jiang. (2012, December 10). An Evaluation of the Application ("App") Verification Service in Android 4.2 (JellyBean) [Online]. Available: http://www.cs.ncsu.edu/faculty/jiang/appverify/.

[4] W. Zhou, Y. Zhou, X. Jiang, and P. Ning. "Detecting Repackaged Smartphone Applicationsin Third-party Android Marketplaces," In Proceedings of the Second ACM Conferenceon Data and Application Security and Privacy, CODASPY'12, pages 317-326, 2012.

[5] S. Hanna, L. Huang, E. Wu, S. Li, C. Chen, and D. Song. "Juxtapp: A Scalable System forDetecting Code Reuse Among Android Applications," In Proceedings of the 9th Conferenceon Detection of Intrusions and Malware & Vulnerability Assessment, DIMVA'12.

[6] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. "The Ghost In The Browser Analysis of Web-based Malware," In First Workshop on Hot Topics in Understanding Botnets, 2007.

[7] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert. "Rozzle: De-cloaking internet malware," Technical Report MSR-TR-2011-94, Microsoft Research Technical Report, 2011.

[8] M. Polychronakis, and N. Provos. "Ghost turns zombie: Exploring the life cycle of web-based malware," In First USENIX Workshop on Large-Scale Exploits and Emergent Threats, 2008.

[9] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. "All your iframes point to us," In USENIX Security Symposium, 2008.

[10] Symantec Report on Attack Kits and Malicious Websites [Online]. (2011). Available: http://www.symantec.com/content/en/us/enterprise/other_resources/b-symantec_report_on_attack_kits_and_malicious_ websites_21169171_WP.en-us.pdf .

[11] M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos. "Comprehensive shellcode detection using runtime heuristics," In Annual Computer Security Applications Conference (ACSAC), 2010.

[12] Y. Fratantonio, C. Kruegel, and G. Vigna. “Shellzer: a tool for the dynamic analysis of malicious shellcode,” In Recent Advances In Intrusion Detection, 2011.

[13] Libemu – x86 Shellcode Detection. Available:http://libemu.carnivore.it.

[14] A. Pouik. (2012, April) Similarities for Fun & Profit Phrack Issue#68.

RiskSense New Mexico
4200 Osuna Road NE, Suite 3-300
Albuquerque, NM 87109
United States

RiskSense Silicon Valley
530 Lakeside Drive, Suite 170
Sunnyvale, CA 94085
United States

General Inquiries
+1 505.217.9422
+1 844.234.RISK (toll free)
info@risksense.com

Media Inquiries
media.relations@risksense.com