IDS for Detecting Malicious Non-Executable Files Using Dynamic Analysis

Ahmad Bazzi
Graduate School of Engineering
Gunma University
1–5–1 Tenjin-cho, Kiryu, Gunma 376–8515, Japan
Telephone/Fax: (+81) 277–30–1837
Email: ahmad@nztl.cs.gunma-u.ac.jp

Yoshikuni Onozato
Division of Electronics and Informatics
Faculty of Science and Technology
Gunma University
1–5–1 Tenjin-cho, Kiryu, Gunma 376–8515, Japan
Telephone/Fax: (+81) 277–30–1837
Email: onozato@cs.gunma-u.ac.jp

***Abstract*—Attackers are increasingly relying on non-executable files to launch their attacks. Anti-virus solutions can detect a high percentage of malicious files but usually cannot reach and maintain a 100% detection rate. We propose a file-level IDS that relies on automated dynamic analysis system (sandbox) to detect malicious PDF files. We achieved a 99.2% detection rate, where the rates of both the false positives and the false negatives are less than 1%. Because it does not rely on anti-virus signatures, this solution can detect malicious documents that utilize malware not covered by the anti-virus database.**

I. Introduction

Computer users face myriad threats against their computer systems; these threats range from viruses and Trojans to other forms of malware. Current protection mechanisms include firewalls, antivirus systems, intrusion detection systems (IDS) among others. Unfortunately, any security component suffers from certain inherent limitations.

An antivirus system relies on a virus signature database which allows it to efficiently detect viruses with known signatures. This database needs to be updated regularly to cover any new viruses encountered by the antivirus vendor. Unfortunately, this means that such an approach is not efficient at detecting previously unseen malware. Therefore any antivirus solution fails to achieve a 100% detection rate [1], [2].

IDS are generally divided into network-based IDS [3] and host-based IDS [4]. The two main approaches for intrusion detection are attack signature database and anomaly detection [5]. Most commercial solutions, such as Snort [6], rely on attack signature set, which is more mature and reliable; however, it fails to detect attacks not covered by its database.

Attackers used to rely exclusively on executable files to launch their attacks; however, this has changed in the past years. Common types of non-executable files can be made malicious if a suitable software vulnerability is discovered in its assigned viewer. This includes office documents, PDF files, etc. Moreover, users have been trained to trust non-executable files and this makes them more prone to get infected.

Let us consider PDF files as an example. Modern PDF file format support JavaScript [7]. When a vulnerability is discovered inside a PDF viewer, attackers attempt to exploit it by writing suitable JavaScript code [8]. For further details, Ab Rahman guides the reader through the analysis of several malicious PDF files using commonly available tools [8]. Similarly, Stevens discusses the analysis of various malicious PDF files using publicly available tools that he has created for PDF analysis [9].

The limitations of the antivirus and IDS, along with the gravity of software vulnerabilities, are amplified when the attacker is a rich corporation or even a government. These players have shifted the complexity of the attacks to a new level. Motives now include cyber-espionage [10] and cyber-warfare [11].

In brief, the sophistication of the players and the limitation of the current solutions require a different approach to handle the new threats, such as malicious non-executable files especially if implementing zero-day exploits. We think that automated dynamic analysis approach is one promising solution, and we implement a file-level IDS utilizing this approach. A file-level IDS inspects the transferred files instead of the network packets.

This paper is organized as follows: Section II discusses selected previous works. After we present the general steps of the proposed solution in Section III, we discuss our working prototype and the experimental results in Section IV. Finally we evaluate the advantages and limitations of this solution in Section V before writing our conclusion in Section VI.

II. Previous Works Related to Malicious PDF Detection

There have been several modern attempts to detect malicious PDF files. In particular, the two approaches are static analysis which studies the code without running it, and dynamic analysis which requires running the inspected code.

Laskov and Šrndić apply static analysis on the JavaScript code extracted from the PDF documents in order to detect the malicious ones [12]. Smutz and Stavrou also use static analysis; they extract the features from the document metadata and structure and apply an ensemble learning method for classification [13]. Tzermias et al. combine static analysis of certain document objects along with dynamic analysis of the embedded JavaScript code [14]. We follow a different approach by relying on dynamic analysis of the whole document file without the use of static analysis.

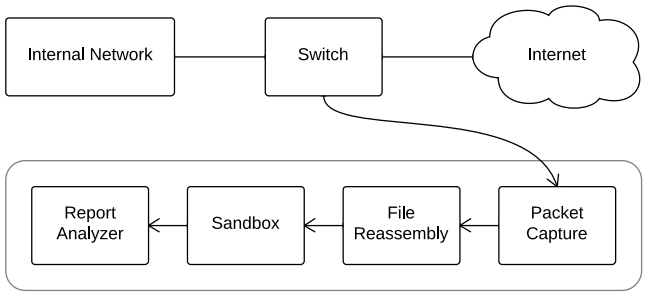

Fig. 1. File-level IDS with automated dynamic analysis system: the transferred files are assembled, submitted to the sandbox and the resulting sandbox report is automatically analyzed.

III. Proposed Solution

We propose a file-level IDS with an automated dynamic analysis system. The proposed system is shown in Fig. 1. The protected network is connected to the Internet through a switch that mirrors all the traffic to the IDS. The IDS captures all the passing packets and reassembles the selected file types. Each reassembled file is submitted to the sandbox for analysis. The sandbox opens the submitted file and generates a report of the observed activities. This report is analyzed using machine learning to detect whether the file is malicious.

The performed tasks are the following:

1) Capture all packets
2) Assemble the chosen file types
3) Submit the assembled file to the sandbox
4) Wait for the sandbox to open the submitted file and generate a report
5) Extract certain features from the resulting report and use machine learning for analysis
6) Notify the administrator if the file is detected as malicious

IV. Prototype Implementation and Results

A. Implementation

The system is comprised of the following main components:

1) Packet Capture Module: We implemented the packet capture module using libpcap[1], a mature and robust open source C/C++ library for capturing network packets.

2) File Reassembly Module: This module assembles the captured packets of the transferred files of selected file types.

3) Sandbox Solution: Cuckoo's Sandboxsatisfied all our needs for the dynamic analysis system. It is an open source program written mainly in Python and connects to virtualization software to execute or open the submitted file. It reports on the following [15]:

1) Files created, downloaded or deleted by the submitted file
2) Traces of Win32 API calls performed by the processes related to the submitted file
3) Memory dumps of the relevant processes
4) Network traffic capture
5) Full memory dump
6) Screenshots of the Windows desktop during the opening/execution of the submitted file

To process a file, the sandbox starts with a clean virtual machine (VM), opens the submitted file inside this VM, records the observed activities and generates a report.

4) Report Analyzer: We implemented a script in Ruby language to extract certain features from the Sandbox reports. It extracts the *numbers* of the following:

1) Different screenshots
2) Created files
3) Accessed files
4) Accessed registry keys
5) Active processes
6) Successful process actions
7) Failed process actions
8) Registry-related process actions
9) File system-related process actions
10) Process-related process actions
11) Services-related process actions
12) Network-related process actions
13) Synchronization-related process actions

These features are used in the machine learning classification model. We used support vector machines (SVM) and relied on LIBSVM [16].

B. Experiment

The experiment is conducted using 6,052 benign files and 10,852 malicious files. We obtained these files from Contagio Dump[2]. The training set is made of 6,000 files: 3,000 benign files and 3,000 malicious files. The testing set is made from the remaining 10,904 files, i.e. 3,052 benign files and 7,852 malicious files.

We analyzed the sandbox reports of the related files based on the 13 extracted features using LIBSVM. We achieved the following results:

- $Accuracy = 99.175\%$, which is the percentage of correctly classified files to the total number of files.
- $Percentage\ of\ False\ Positives = 0.983\%$, which is the percentage of misclassified benign files to the total number of files.
- $Percentage\ of\ False\ Negatives = 0.764\%$, which is the percentage of misclassified malicious files to the total number of files.

C. Detection of New Malicious Files

In Section IV-B, we used a training set of 6,000 files. The question that arises is whether this model can detect malicious files that implement newer techniques. Hence we obtained another 148 malicious files using exploits newer than the ones used in the training. We obtained the following results:

- $Accuracy = 95.946\%$

[1] http://www.tcpdump.org/

[2] http://contagiodump.blogspot.com/

- $Percentage\ of\ False\ Negatives = 4.054\%$

Although the detection accuracy is not as high as earlier, it is important to note that the PDF viewer in the sandbox VM is actually immune to this new vulnerability, yet the system was still able to reach almost 96% detection rate. In a real life scenario, the sandbox VMs should use the same versions of the vulnerable programs used by the protected clients.

V. Solution Evaluation

In this section we consider the advantages and disadvantages of our system.

A. Advantages

1) Anomaly detection: This approach does not rely on any static malware signature database but rather on a model of the behavior of the system when opening benign and malicious files. This allows it to detect new malicious files whose signatures are not explicitly available. Moreover, this solution can be combined with an anti-virus solution to achieve even higher detection rates considering that their detection methods are relatively independent.

2) Modularity: As for the design, this system is quite modular and might also be implemented as part of a proxy server. This would eliminate the need for the packet capture and reassembly modules.

3) Intrusion prevention system: The proposed design is an intrusion detection system. Converting this to an intrusion prevention system (IPS) can be achieved by implementing it inline, i.e. between the Internet and the local network. The main limitation is that each file requires around two minutes —depending on the configuration of the sandbox— and this might not be tolerated by the users.

B. Disadvantages

1) Processing Time: The processing time is around 2 minutes so that the file is opened in the sandbox and the malicious code is given enough time to execute. In common environments, users might not tolerate such a delay; therefore, an IDS configuration would be more suitable. However, if the network is highly critical, we expect the users to tolerate the added delay imposed on the files accessed from untrusted networks in order to achieve the added security. Accordingly, an inline deployment becomes the recommended option.

2) Virtualization-Aware Malware: Virtualization-based dynamic analysis systems are becoming more common and this is pushing the attackers to create virtualization-aware malicious code. In other words, the malicious code will not completely execute in a virtual computer environment in order to avoid detection. In our approach, we rely on a diversity of features where opening a malicious file would still generate enough anomalous signatures even if the malicious code is not fully executed.

3) Encrypted Connections: Files transferred over encrypted connections cannot be assembled using this approach. Although using some form of the man-in-the-middle attack might help obtain the files in clear text, it is privacy invasive. Special measures might need to be taken to ensure that files transferred over encrypted connections are not malicious.

VI. Conclusion

In this paper, we presented a file-level IDS that relies on an automated dynamic analysis system to detect malicious files. Using SVM, we classified the files using several features extracted from the sandbox reports. The achieved detection accuracy is around 99.2% with relatively low rates of false positives and false negatives.

References

[1] S. Edwards, "Home Anti-Virus Protection, January - March 2013," Dennis Technology Labs, Tech. Rep. April, 2013. [Online]. Available: http://www.dennistechnologylabs.com/reports/s/a-m/2013/DTL_2013_Q1_Home.1.pdf

[2] AV-Comparatives, "Whole Product Dynamics "Real-World" Protection Test - (August-November) 2012," AV-Comparatives e.V., Tech. Rep. November, 2012. [Online]. Available: http://www.av-comparatives.org/wp-content/uploads/2012/12/avc_prot_2012b_en.pdf

[3] M. Cremonini, "Network-Based Intrusion Detection System," in *Handbook of Information Security - Volume 3*, H. Bidgoli, Ed. John Wiley & Sons, Inc., 2006, pp. 713–729.

[4] G. Vigna and C. Kruegel, "Host-Based Intrusion Detection," in *Handbook of Information Security - Volume 3*, H. Bidgoli, Ed. John Wiley & Sons, Inc., 2006, pp. 701–712.

[5] P. Ning and S. Jajodia, "Intrusion Detection System Basics," in *Handbook of Information Security - Volume 3*, H. Bidgoli, Ed. John Wiley & Sons, Inc., 2006, pp. 685–700.

[6] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1039834.1039864

[7] Adobe Systems Incorporated, *Document manage - Portable document format - Part 1: PDF 1.7*, 1st ed. ISO, 2008. [Online]. Available: http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf

[8] M. Ab Rahman, "Getting Owned By Malicious PDF - Analysis," SANS Institute, 2010. [Online]. Available: http://www.sans.org/reading_room/whitepapers/malicious/owned-malicious-pdf-analysis_33443

[9] D. Stevens, "Analyzing Malicious PDF Files," 2010. [Online]. Available: http://didierstevens.com/files/data/malicious-pdf-analysis-ebook.zip

[10] Mandiant, "Mandiant APT1," Mandiant, Tech. Rep., 2013. [Online]. Available: http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf

[11] G. Mcdonald, L. O. Murchu, S. Doherty, and E. Chien, "Stuxnet 0.5: The Missing Link," Symantec Corporation, Tech. Rep., 2013. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/stuxnet_0_5_the_missing_link.pdf

[12] P. Laskov and N. Šrndić, "Static Detection of Malicious JavaScript-Bearing PDF Documents," in *Proceedings of the 27th Annual Computer Security Applications Conference on - ACSAC '11*, ser. ACSAC '11. New York, New York, USA: ACM Press, 2011, pp. 373–382. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076785

[13] C. Smutz and A. Stavrou, "Malicious PDF Detection using Metadata and Structural Features," in *Proceedings of the 28th Annual Computer Security Applications Conference on - ACSAC '12*. New York, NY, USA: ACM Press, 2012, pp. 239–248. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2420950.2420987

[14] Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. P. Markatos, "Combining Static and Dynamic Analysis for the Detection of Malicious Documents," in *Proceedings of the Fourth European Workshop on System Security - EUROSEC '11*, ser. EUROSEC '11. New York, New York, USA: ACM Press, 2011, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/1972551.1972555

[15] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser, "Cuckoo's Sandbox," 2013. [Online]. Available: http://www.cuckoosandbox.org

[16] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.