Digital Investigation 22 (2017) S48–S56

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS 2017 USA — Proceedings of the Seventeenth Annual DFRWS USA

Insights gained from constructing a large scale dynamic analysis platform

Cody Miller [a], Dae Glendowne [b], Henry Cook [c], DeMarcus Thomas [b, *], Chris Lanclos [b], Patrick Pape [b]

[a] Babel Street, 1818 Library St., Reston, VA, USA
[b] Distributed Analytics and Security Institute, 2 Research Boulevard, Starkville, MS, USA
[c] Green Mountain Technology, 5860 Ridgeway Center Parkway, Suite 401, Memphis, TN, USA

Keywords:
Malware
Dynamic analysis
Cuckoo sandbox

A B S T R A C T

As the number of malware samples found increases exponentially each year, there is a need for systems that can dynamically analyze thousands of malware samples per day. These systems should be reliable, scalable, and simple to use by other systems and malware analysts. When handling thousands of malware, reprocessing a small percentage of the malware due to errors can be devastating; a reliable system avoids wasting resources by reducing the number of errors.

In this paper, we describe our scalable dynamic analysis platform, perform experiments on the platform, and provide lessons we have learned through the process. The platform uses Cuckoo sandbox for dynamic analysis and is improved to process malware as quickly as possible without losing valuable information. Experiments were performed to improve the configuration of the system's components and help improve the accuracy of the dynamic analysis. Lessons learned presented in the paper may aid others in the development of similar dynamic analysis systems.

Introduction

As the arms race between malware creators and security professionals progresses, new adaptations are needed, and made, every year. One such adaptation on the security side is malware behavior identification via dynamic analysis. AV-Test indicates that the total number of new malware has increased significantly in the past five years from under 100 million in 2012 to over 500 million in 2016. Due to this increasing number of new malware distributed annually, dynamic analysis systems must be able to process tens of thousands of malware samples per day. In order to meet such a large quota, and remain manageable, the systems need to be reliable, scalable, and convenient for the users. In the effort to create such a system, there are many turning points at which a decision impacts performance and efficiency. It is important to consider all the viable options for these turning points, which can be an overwhelming task for an already complex system. The research presented in this paper seeks to improve understanding of these choices by presenting our system design, and the decisions that were made to ensure high performance and quality of dynamic analysis.

The main software component of the system described in this paper is the open-source dynamic malware analysis platform Cuckoo Sandbox developed by Guarnieri et al. (2013). Due to Cuckoo Sandbox's broad range of virtualization software support and customization (as a result of it being open source and of its extensive plugin support), a multitude of customized systems can be developed around it. The system presented in this paper is just one such iteration, optimized and backed up by testing options at various decision points. Some of the optimization areas focused on in this research include: identifying performance bottlenecks, efficient machine/software configurations for virtualization, and malware execution time limit tradeoffs. Along with the optimization solutions, we also discuss how the presented system is scalable due to our distribution scheme and database storage. Along with detailed explanations of the above areas, within the context of our developed system, this paper provides some lessons learned throughout the process which can aid in the future development and improvement of similar systems.

The growth of malware samples found each year puts more expectations on an already taxing responsibility as a digital

* Corresponding author.
E-mail address: dmt101@dasi.msstate.edu (D. Thomas).

http://dx.doi.org/10.1016/j.diin.2017.06.007

forensics examiner. The only way that digital forensics examiners will be able to respond to the enormous amount of malware being developed is through more sophisticated tools that are developed through lessons of past and current tools. The first contribution of this research is the scalable dynamic analysis platform, which could be used by digital forensics examiners to respond to the sheer amount of malware being developed yearly. Secondly, this paper discusses the different experiments that were used to optimize the platform. Lastly, in the process of developing the scalable dynamic analysis platform, lessons were discovered that should be taken into consideration by future digital forensic tool developers. The lessons could be just as important as the scalable tool because of the ever changing digital environment.

Related work

According to Kruegel (2014), three main aspects of a dynamic analysis system must be true for it to be effective: visibility, resistance to detection, and scalability. Visibility is the ability to effectively monitor the activity of a sample in an analysis environment. With the increased environmental-awareness of malware samples, a sandbox must also be effective at hiding their presence to avoid identification. In addition to these, the ability to process samples at a high rate is a requirement due to the sheer volume of malware samples being produced. Kirat et al. (2011) compare the processing and restore speeds for varying analysis environments, and compare the number of samples that can be processed within a minute. In their experiments, they compared results from BareBox, VirtualBox, and QEMU for automated analysis of 100 samples. The results show the ability to run 2.69, 2.57, and 3.74 samples per minute respectively when the samples were executed for 15 s each. Blue coat malware analysis s400/s500 is a sandbox solution that states the ability to process 12,000 samples daily.

When considering a dynamic analysis system, you must also consider the method that tools use to extract information. Guarnieri et al. (2013) suggested that most systems will monitor the API calls (systems calls) to obtain an idea of what may be occurring in the system. In addition to this, some systems will also monitor the steps between API calls (Kruegel, 2014), perform taint analysis to monitor information as it propagates through the system (Song et al., 2008), execute samples multiple times with varying OS's and system configurations to identify environment sensitivity (Song et al., 2008; Provataki and Katos, 2013), execute hardware emulation (Kruegel, 2014), use bare-metal systems to avoid evasive techniques (Kirat et al., 2011; Kirat et al., 2014), incorporate integration with memory analysis frameworks (Guarnieri et al., 2013), etc. Also, when considering an analysis system, the pros and cons of open-source vs. closed-source projects must be evaluated.

An additional aspect to consider for any dynamic analysis environment is selecting an optimal time of execution per sample. Keragala (2016) stated that samples can exhibit stalling behavior to defeat time-out limits of some systems if not properly selected. Several works did not state a specific execution period, but analysis reports showed execution times ranging between 2 and 3 min (Provataki and Katos, 2013; Vasilescu et al., 2014; Rieck et al., 2011). Lengyel et al. (2014) selected an arbitrary duration period of 60 s. To our knowledge, there has only been a single published work (Kasama, 2014) which performed an empirical evaluation of the optimal execution time for samples in a dynamic analysis system. However, this work had a limited number of samples (5,697) and captured API calls. The experiment in this paper is meant to confirm the results presented by Kasama.

System overview

Cuckoo Sandbox

Cuckoo Sandbox is an automated dynamic analysis sandbox created by Guarnieri et al. (2013). Cuckoo allows the submission of files to be run in an isolated environment. Cuckoo first reverts a VM to a base snapshot (one that is not affected by malware), then it runs the malware on the VM. While the malware sample is running, Cuckoo collects information about what it does in the sandbox such as: API calls, network traffic, files dropped, etc. Cuckoo's collected information will be referred to as a Cuckoo sample for the remainder of this paper. Spengler created Spender-sandbox (Cuckoo 1.3), a modified version of Cuckoo 1.2 that adds a number of features and bug fixes. “Cuckoo modified” (a branch separate from the main Cuckoo version branch) was selected over Cuckoo 1.2 because the modified version adds, among other things, new hooks and signatures.

Cuckoo nodes

To process malware, five hosts running ESXi 5.5.0 were used. The hardware varies slightly between the hosts, two of them have 16 physical cores and three of them have 20 physical cores and all five have 128 Gib of RAM. Each host also has an adapter connected to an isolated network that the hosts share. A virtual machine (VM) running CentOS 7 and Cuckoo (i.e., a Cuckoo node) is on each host and has 64 Gib of RAM and 28 virtual cores. The Cuckoo nodes each have 20 Cuckoo agent VMs within them. All together there are 100 Cuckoo agent VMs managed by the five Cuckoo nodes. This setup of VMs inside of a VM was chosen because, according to Kortchinsky (2009) and Wojtczuk and Rutkowska, though unlikely, malware can escape the agent VM and attack the host; keeping the agent inside another VM adds another layer of isolation.

Cuckoo agents

The driving research behind implementing this system focuses primarily on malware that targets the 32-bit version of Windows 7. Therefore, Windows 7 32-bit virtual machines were used for the Cuckoo agents; QEMU version 2.5.1 was used as the virtualization architecture. The agent virtual machines have: a new installation of Windows 7, 512 Mib of RAM, 1 CPU core, Adobe Reader 11, Python 2.7 installed, and have Windows firewall and UAC disabled. All the agent VMs used the network adapter on the node that is connected to the isolated network of VM hosts.

INetSim

All the agent VMs were connected to an isolated network that does not have access to the Internet. INetSim was created by Hungenberg and Eckert and was used to spoof various Internet services such as DNS, HTTP, and SMTP. It runs on its own VM which is on the same network as the agent VMs. Gilboy (2016) stated that INetSim can improve malware execution as it can trick malware that require the Internet. However, this does not help if the malware needs external resources, such as a command and control server, as INetSim does not provide external (Internet) resources to the isolated network.

Results server

The results server is a VM that provided a way to get the Cuckoo samples from the Cuckoo nodes directly without using Cuckoo's built-in API to fetch the results, thus improving transfer and