Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

FIIT-XXXXXX-82385

Tomáš Belluš

Immutable infrastructures for security demanding environments

Výskumný zámer

Study program: Information Security
Field of study: 9.2.4 Computer Engineering
Training workplace: Institute of Computer Engineering and Applied Informatics
Supervisor: Ing. Tibor Csóka, PhD.

December 2019

Contents

Chapter 1

Motivation

How to successfully delivery, exploit and compromise a target system? Cyber criminals (attackers) follow a multi-phase process to completely plan the sophisticated attack. The phases fall under so called Cyber/Intrusion Kill Chain and include - reconnaissance, weaponization, delivery, exploitation, installation, command and control, actions on objective [1]. Briefly, the attack begins with analyzing and discovering the target objective, followed by preparation of the exploit to compromise the system and getting required control to fulfill a goal.

In contrast to the attackers, security experts use equally complex and useful tools and practices to detect, mitigate or reproduce an attack. Tools like sandbox, sinkhole, honeypot and many active detection tools (e.g. WAF, IPS/IDS, FW, etc.) In the "fight" of blue versus red, the blue team must keep up with the red teams' techniques and zero-day exploitation. Zero-day exploitation, or attack, target vulnerabilities not addressed by vendors and not publicly known (beside the attackers) [2].Such vulnerabilities are almost impossible to mitigate and very difficult to detect. White hat or ethical hackers search for these vulnerabilities in their own environments by reading all those lines of code or dynamically - utilizing honeypots.

Detecting zero-day vulnerabilities in a honeypot may result in a absolute failure of the honeypot. The immutable infrastructure paradigm would ensure the whole environment may be recovered. Usage of immutable infrastructure may introduce the possibility of designing a complex honeynet composed of multiple interconnected network segments and devices.

[1] https://maritime-executive.com/blog/the-seven-phases-of-a-cyber-attack

[2] https://us.norton.com/internetsecurity-emerging-threats-how-do-zero-day-vulnerabilities-work-3
html

Chapter 2

Analysis

"The primary technology that makes immutable infrastructure possible at any scale is virtualization (both software and hardware) across networking, servers, and storage" [1]. According to this, virtualization is the inseparable element of immutable infrastructure.

2.0.1 Immutable infrastructure and its orchestration

To understand the applications of immutable infrastructure one must identify its characteristics. Any system or unit of a system (i.e. container, VM, server) is pre-configured and deployed to its final state in a target environment. The state is immutable - it forbids any further configuration changes or additional networking. Required changes are applied in the "infrastructure configuration" (e.g. container images) and redeployed to apply the updated state. In comparison, a mutable infrastructure is the exact opposite, such as almost every computer, that requires manual step-by-step installation of the OS and the underlying applications and network configurations.

Having a complex infrastructure composed of multiple servers/containers (any solo-standing service) allows segmentation of immutable services. Instead of upgrading/modifying a whole infrastructure, only the desired segments are addressed. Although, it introduces a problem of orchestration and segmentation of services to the smallest unit (i.e. container). Possible segmentation of services is discussed in the next section - Escape attack mitigation. Orchestration is the question of how, when and what combined in a single process of the mechanism used.

The immutability has a profound effect on the security of the infrastruc-

[1] `https://www.sumologic.com/insight/mutable-immutable-infrastructure/`

ture. Since every change is made by redeploying the infrastructure, all ongoing attacks vanish despite being as complex as outlast a reboot. **Although, the data stores (i.e. databases, message queues) are very much mutable components of the infrastructure, therefore they should be subject to analysis.**

2.0.2 Escape attack mitigation

Escape attacks exploit vulnerabilities in shared resources or stack level (e.g. kernel for KVM, application for functions or hardware for virtualization). A case study, with a goal to mitigate any form of escape attacks, at the Swedish Police Authority by Christian Abdelmassih - Docker and Kubernetes in high security environments. The author discusses application isolation techniques while providing orchestration with Kubernetes.

One technique is to isolate applications by containerization (e.g. Docker, LXC, etc.), which share the host's kernel. Second technique is separation by bare-metal hypervisor - virtual machines (VM). This means that the VM has it's own operation system (OS) and shares hardware resources of the host. In conclusion, the idea is to utilize bare-metal hypervisor (cloud native) to minimize the attack surface for escape attacks, because containers are vulnerable to kernel driven escape attacks.

To separate non-related application and utilize containers, he suggests segmentation of Kubernetes pods to logical classes (e.i. class O, class P and class PG). These classes are to be organized and orchestrated by Kubernetes and should make sure nodes and pods have the same class tags. Although, the solution was but present and lacks verification which opens opportunities for my thesis, as the the author finalizes that "In order to utilize this in clouds in a scalable manner, there are additional requirements for automation that must be satisfied. For example, automating the creation of virtual machines, attaching them to the Kubernetes cluster. Most importantly one must also implement and verify that the application classifications are respected at all times." [Chistian Abdelmassih, 24.01.2019]

2.0.3 Honeypots & Sandboxes

Both honeypots and sandboxes are similarly used in dynamic threat analysis. Difference is in preparation and isolation. Honeypot monitors the activity of an attacker or a malware in emulated or production-like environment. Sandbox is an isolated environment for dynamic malware analysis with all preparation that is needed based on the knowledge of the malware. Usually,

honeypots are configured to detect specific attacks with emulated applications or not emulated to detect zero-day vulnerabilities and monitor behavior of potential exploitations [2]. Within a sandbox, all activity is monitored as much as possible, while knowing the outcome of the attack, to know the source of the attack.

A particularly interesting topic is to detect zero-day vulnerabilities in a honeypot and, if necessary, replicating and more in depth analyzing the exploitation in a sandbox. Analyzing malware in a sandbox provides virtualization, which mitigates MBR overwrite. On the other hand, most malwares verify the environment they're in - detection of virtualization or sandbox awareness.

[2] https://www.enisa.europa.eu/publications/proactive-detection-of-security-incidents-II-

Chapter 3

Master's thesis overview

3.1 Research questions

How to improve virtualization/emulation detection in a honeypot or a sandbox?

What is the most detectable phase in the Cyber Kill Chain?

The underlying technology for virtualization and containerization (if any) is crucial to reach immutability. What provisioning and orchestration mechanisms are needed to ensure an immutable infrastructure?

3.1.1 Thesis goal

Design a honeypot or a sandbox, in Linux, as a dynamic analytic tool dedicated to detect zero-day vulnerabilities and mitigating detection. Utilize the immutable infrastructure paradigm to ensure automated and a secure system. Mitigating escape attacks will not be in scope of this thesis, but the introduced segmentation technique will be taken into account to segregate related services.

3.1.2 Plan of work

DP I

- analyze virtualization (e.g KVM), emulation and virtualization detection mitigation practices
- analyze cross-platform orchestration and provisioning mechanisms
- comparison of existing solutions and discussion of possible improvements

- design of ensuring immutability and the whole stack of the system supported by analysis

DP II

- discuss possible attack vectors meant for monitoring
- implementation of the system skeleton and configuration of the orchestartion mechanism
  - create base images for the system
  - automated deployment
- document the implementation
- assignment revision of necessary

DP III

- test honeypot implementation on live server
- analyze all retrieved data and indicators of compromise
- apply required modifications to images or core functionality
- finalize the thesis documentation and evaluate reached goals

Chapter 4

Further Literature

https://pdfs.semanticscholar.org/d4e4/5e81b8d8878ca99648c3fc890ede1ae01b49.pdf

https://kth.diva-portal.org/smash/get/diva2:1231856/FULLTEXT02.pdf

https://www.enisa.europa.eu/publications/proactive-detection-of-security-incident

https://www.sumologic.com/insight/mutable-immutable-infrastructure/