"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XIX – 2016 – Issue 2
The journal is indexed in: PROQUEST / DOAJ / Crossref / EBSCOhost / INDEX COPERNICUS / DRJI / OAJI / JOURNAL INDEX / I2OR / SCIENCE LIBRARY INDEX / Google Scholar / Academic Keys/ ROAD Open Access / Academic Resources / Scientific Indexing Services / SCIPIO / JIFACTOR

HONEYPOT SYSTEM BASED ON SOFTWARE CONTAINERS

Sergiu EFTIMIE[1]
Ciprian RĂCUCIU[2]

[1]Inf. Ph.D. Student, Military Technical Academy - Electronic, Information and Communication Systems for Defense and Security Doctoral School
[2]Prof. Eng. Ph.D., Military Technical Academy - Electronic, Information and Communication Systems for Defense and Security Doctoral School

Abstract: *In this paper we explore aspects of building a honeypot system using software containers. Despite their advantages, organizations see honeypots as too complex from a deployment and management perspective. As software containers gain popularity these issues can be addressed using light containers hosted on cloud infrastructures.*

Keywords*: Honeypot, Software containers, Cloud, Automation*

Introduction

In this paper we explore aspects of building a honeypot using software containers. A honeypot system represents an intrusion detection technology that becomes relevant to a security setup when it's being attacked. Although there has been a growing interest over the last years in honeypots and other related technologies, organizations still see honeypots as too complex from a management and deployment perspective, despite the fact that they are now beginning to play an important role in enterprise security. In the first section of the paper we will describe the different types of honeypots and their uses. The second section describes the technology behind software containers and will emphasize the advantages of using them in application development. In the third section we will exploreaspects of implementing a honeypot using container technology.

Honeypots

Honeypots are systems that are built to mimic actual devices on a network [1]. These types of systems were built in the past primarily by researchers with the goal of studying the behavior of the attackers. In the present days they are used to provide early detection of malicious network activity. They take up unused IP address space of an enterprise network and continuously listen for malicious activities performed by attackers [2].Any interaction with a honeypot is by definition considered suspicious.

From a deployment perspective honeypots can be classified into two categories, production and research honeypots. Production honeypots are used by organizations and are placed next to production servers inside the network. They generally work to improve the security posture by detecting attacks and give less information about the modus operandi of the attackers. Research honeypots are more complex both from a deployment and a maintenance perspective. They are run by organizations such as universities, military or government to gather extensive information about hacking methods and tactics. From a design standpoint honeypots can be classified as low-interaction, high-interactionand pure honeypots. Low-interaction honeypots are built to simulate services that are frequently probed by attackers. They consume a small amount of resources and the overall complexity of the system is reduced. Low-interaction honeypots are safer because they emulate vulnerabilities and therefore cannot be leveraged by an attacker.

High-interaction honeypots are more complex and they duplicate some of the activities present in production systems. The user is allowed to interact with the operating systems in order to capture extensive information about the attack. High-interaction honeypots are in general more difficult to detect and have an increased maintenance cost. Pure honeypots are set up by emulating vulnerabilities on actual production systems.

The usage of honeypots has many advantages over traditional intrusion detection systems. Honeypots collect a small amount of data in comparison to traditional intrusion detection systems. This leads to a better response and action on unauthorized activity.

IDSs can generate a lot of false alerts and can also have a difficulty in identifying new types of attacks.

Honeypots on the other hand generate few false negatives and positives because any activity around them is unauthorized by definition.Honeypots also use fewer resources than IDSs and are capable of preventing attacks in multiple ways. They are able to stop worms that scan entire networks looking for vulnerabilities

DOI: 10.21279/1454-864X-16-I2-062

and they can deter human attacks by providing fake resources while giving time for the security officers to respond to the attack.

Honeypots contribute to the overall security by the early detection of unknown attacks and that is where their added value lies. Also honeypots are being used to detect attacks from within an organization. They can also help to respond to attacks and can be used as incident-response tools. New implementations include threat response mechanisms such as the ability to adapt systems based on an unauthorized activity.Because of the fact that honeypots have no use if no one attacks them and because they can introduce new risks by providing attackers new platforms from which they can launch attacks, they must be used in conjunction with other security mechanisms.The honeypot application itself should be secured especially in the high-interaction case because of increased attack surface.

Containers and application development

The container portability has led to an increase use of this technology inthe application development area especially in the cloud computing scene.

The container architecture represents another major benefit because it providesa standard way to divide applications into distributed objects. By splitting applications in this way, containers allow the placement of different application components on different physical or virtual machines. This adds flexibility and a series of advantages regarding workload management.Docker takes advantage of a series of Linux kernel security features such as kernel namespaces to isolate users, processes, networks and devices, and cgroups to limit resource consumption [3].

The use of clustering, scheduling and orchestration has improved the scaling and the resilience of the applications based on containers [4].

Open Container Initiative (OCI) is a project built around the concept of software containers with the ultimate goal of developing a standardized platform in which applications and their dependencies are encapsulated in containers.

Containers provide better utilization of computing resources by eliminating the hypervisor, while maintaining separation and isolation tasks without using an operating system.

The project gain wide attention because the promise of portability, agility and interoperability in a broad range of infrastructures. The concept of containerization allows virtual instances sharing a single operating system, along with libraries and relevant drivers. This approach reduces resource consumption, since each container contains only related dependencies. There is a rapidly growing interest in using the container-based solutions. Users can fully adopt new technology without the risk of blocking a long-term technology provider.

Containerization also brings a number of improvements like the fact that virtualization is performed at the operating system level. Containers share the same kernel, and sometimes parts of the host operating system. This use of containers offers enhanced flexibility and a small size advantage compared to the use of a hypervisor.

Containers have a series of advantages and capabilities:

- Container abstractions reduce complexity. Containers do not require dependencies on the application infrastructure and in consequence there is no need for a complex interface with the platform services.
- Containers provide advanced distributed computing capabilities. Application components built on containers can be executed on different cloud platforms. Enterprises can choose cloud providers based on cost and performance.
- Containers can take advantage of the automation in order to maximize portability.
- Containers provide better security and governance. Security and governance services are specific to the platform and not to the application. By placing security and governance outside of the container reduces complexity in a significant way.
- Containers can provide automation services that make use of policy-based optimization. An automation layer can locate a suitable platform to execute containers and migrate automatically to the respective platform.

For hosting providers, the primary benefit of using containers was the increased density. For enterprises, in the past, in most data centers there was no need fora density increase. Virtualization as a technology was suitable for businesses because most servers had low utilization and it represented a good way to use their full capacity. Thus, virtualization has helped increase density but did not bring any real value to businesses. Containerization is in the position now to add real value through the scaling of the applications.

There are two basic approaches to scale applications using containers. One approach is to create a custom system that manages the containers. This will translate into a system that launches new container instances in an automated manner in order to handle an increasing workload. This approach has hidden costs that rest in maintenance.

A second approach is to use one of the container technologies on the market that will provide the basic means to enable scalability through

DOI: 10.21279/1454-864X-16-I2-062

orchestration, scheduling, and clustering. We will present some of the tools available on the market:
GoogleKubernetes is an open-source system for automating deployment, operations, and scaling of containerized applications, basically acontainer cluster manager. The system can schedule a number of container copies across a group of node instances. The container replication and distribution assures the scalingin mostcontainer-based applications.This approach to scaling containers is similar to the ones provided by the other tools.
Cloudify is another service that provides an orchestration tool that has some similar functionswith Docker Swarm and Docker Compose.
The developers have the option to describe complex topologies using YAML (YAML Ain't Markup Language) blueprints. This includes infrastructure, middleware and application layers.
Cloudify is a more orchestration-oriented tool, and should be considered in orchestration and automation tasks and not clustering.
Docker Swarm is a tool that provides clustering, schedulingand integration capabilities. Developers can use this tool to build and ship distributed applications based on multiple containers that include the necessary scaling and management.

Honeypot scheme and implementation

For our purposes we will implement a low-interaction honeypot that will emulate a series of vulnerable web services.We will not pursue high interaction honeypots because they are more difficult to build and the characterizing and the classification of the current anomalies and attacks isa very complex and a time consuming task, done by experts [5].
The web services will be distributed over a number of Docker containers [Fig. 1].

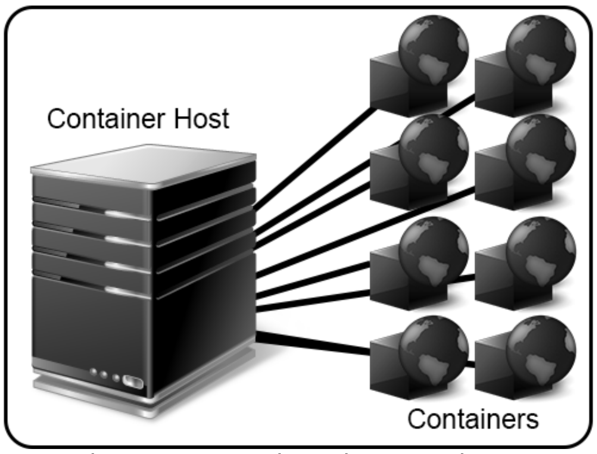

Fig. 1. Honeypot based on containers

Containers have the ability to make connections to the outside world, but not vice versa, so we will use port binding in order to map container ports to the host. Outgoing connections will appear to originate from the host IP addresses. This is accomplished through a masquerading rule, *iptables*that is created by the Docker server on the host machine [6].
Incoming connections are accepted by containers only if they are started using special options when the *run*command is invoked. Two approaches are possible, mapping to an ephemeral port or explicitly to a specific port.
As we mentioned earlier, low-interaction honeypots are built to emulate services that are frequently probed by attackers. In the last few years, Web Services has rapidly evolved by providing attractive features which can be used by businesses and IT organizations [7] but are also introducing new vulnerabilities.In this case we will use explicit mappingthat can be specified using --*publish=SPEC* optionor the *-p SPEC*. This allows the particularization of the port on the Docker server that is mapped to the port in the container.

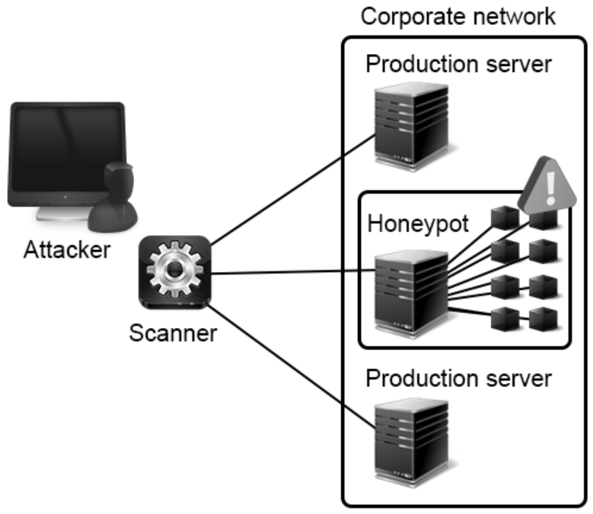

Fig. 2.Vulnerability scanning on a corporate network

A container host will be used to spawn new containers that replicate different types of vulnerabilities. An attacker that uses an automated scanner to scan a network for vulnerabilities [Fig. 2] will trigger an alert in the system if he scans one of the vulnerable ports.
The launch of a container will be accompanied by the introduction of a line in the *iptables* rule. This basically creates a firewall between containers.
Docker runs on the same kernel as the host machine. This enables us to log the actions of the attacker by monitoring the logging at the kernel level.
The containers that will emulate the vulnerabilities will be custom built to replicate the behavior of different operating systems, web services etc. These will be low-interaction honeypots. As stated earlier in the paper, these types of honeypotsare

DOI: 10.21279/1454-864X-16-I2-062

safer because cannot be leveraged by an attacker to perform further attacks.

Future development can be made to improve the efficiency of the honeypots by updating the images with relevant or 0-day vulnerabilities.

CONCLUSIONS

Honeypots are not a new technology. By reducing the number of false alerts, this technology emerges as a key component in a multiple layer approach to intrusion detection. Honeypots do not depend on known patterns of attack but they do have one big disadvantage: their limited field of view. They only capture attacks that are directed towards them and will miss the ones directed against other systems. This is one of the reasons for which honeypots are not recommended to replace existing security technologies. By taking a multi-layer approach, they are used as an important early-detection complementary tool for network and host based intrusion detection.
Honeypots do have a series of advantages.By being a device that is intended to be compromised, means that therewon’t be any production traffic going to or from the honeypot system.
Any connection made to the honeypot, will likely bea possible attack, scan or a probe. If a connection will appear as coming from the honeypotwould mean that the honeypot was compromised.
Although false alerts can be issued, the majority of the honeypot traffic can be seen as malicious activity.
As honeypots are beginning to be deployedin production systems, their advantages become apparent. In time, honeypots have the potential to become an essential part in security operations done at an enterprise-level.
Software container technology has increased in popularity in the last years and has received support from large companies. By combining the advantages of the two technologies, containers and low-interaction honeypots we can obtain an early-detection tool that is effective from a cost perspective and has a low maintenance footprint.
As more enterprises start to encrypt data due to regulations, more and more attacks are performed by using encryption as well. This is a known issue for traditional intrusion detection systems that can be blinded by encrypting network traffic. Honeypots solve this issue because any activity around them as mentioned earlier is considered unauthorized.
By combining the power of containerization and the advantages of the honeypots we can envision the creation of a scalable SecaaS (Security-as-a-Service) system that will possess advanced early-detection capabilities.
We also envision future work on a partially automated system that leverages the CVE (Common Vulnerabilities and Exposures) database in order to create container images suitable for honeypot use.

BIBLIOGRAPHY

[1] Charlie Scott, Richard Carbone- *Designing and Implementing a Honeypot for a SCADA Network*, SANS Institute, 2014
[2] Gokul Kannan Sadasivam, Chittaranjan Hota - *Scalable Honeypot Architecture for Identifying Malicious Network Activities,* International Conference on Emerging Information Technology and Engineering Solutions,2015
[3] Enrico Bacis, Simone Mutti, Steven Capelli, Stefano Paraboschi – *DockerPolicyModules: Mandatory Access Control for Docker Containers,* IEEE CNS 2015 Poster Session, 2015
[4]Andrea Tosatto, Pietro Ruiu, Antonio Attanasio - *Container-based orchestration in cloud: state of the art and challenges,* 9thInternational Conference on Complex, Intelligent, and Software Intensive Systems, 2015
[5]Philippe Owezarski - *Unsupervised Classification and Characterization of Honeypot Attacks,* 10th International Conference on Network and Service Management, 2014
[6] http://www.docker.com/, accessed April 2016
[7] Abdallah Ghourabi, Tarek Abbes, Adel Bouhoula - *Design and Implementation of Web Service Honeypot,* 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2011

DOI: 10.21279/1454-864X-16-I2-062