HoneyPot in a Container with Detailed Analytics

Gaurav Purswani , Hemant N , Raja Kumar Singh ,Raunak Parashar , Divya C D

Vidyavardhaka College of Engineering,Mysuru

gaurav-purswani1234@gmail.com
rk733273@gmail.com
raunakparashar7@gmail.com
divyacd@vvce.ac.in

Abstract. Honeypot is a system that is used as a trap for attackers and can be implemented in organizations and government institutions to trap or leak the origin of the attackers. It can be used in line with other tools such as production servers or even development servers. Some organizations use it in orchestration infrastructure also to figure out any possibilities of an escalation irrespective of which kind of escalation. In this paper we are trying to leverage the containerization capabilities of the modern systems and try to create a honeypot in a container that can be easily installed and operated within a network or outside private networks in the wild very easily with the help of the docker containers.

Keywords: Security, IP Leak, Containerization.

1. Introduction

Many researchers and governments use the honeypots to attract attackers which might think these systems genuine and might try to attack these systems in order to get some sensitive information about the organisation or government institutions[1], These honeypots operate by adding them as components in infrastructure at different places at which they look like legitimate components which the attacker might try to attack. These systems also might contain malicious entities which can be used to leak the origin of the attacker. For example, a honeypot might try to make the browser of the attacker make DNS requests so that in case the attacker is using TOR browser to hide the IP[2], TOR browser sometimes queries DNS with the origin of the attacker and this can be a gem for the honeypot operator as he'll be able to find the exact IP address of the address which was earlier not possible because of the TOR browser. There are lot of features of honeypots some of them are :

Easy Operation
Easy Scalability
Faster Recovery

Sometimes the attacker might try to detect whether the component he is trying to attack is a honeypot or not. This paper tries to deceive the attacker by using a more commonly available application at the front end so that it looks more of a legitimate application than an obvious honeypot.

In today's world of technology the necessary technical security design generally adapted is the design of firewalls along with intrusion detection systems. Still, if the attacker do use masking tools such as VPN provision for a firewall permitted system it is challenging for intrusion detection systems to identify and reduce such attacks. Therefore in a case whereby the attacker recognize any vulnerability in the network through scanning host it is simple to carry out more attacks that can allow the intruder to acquire sensitive data. Most of the security methods used now emphasize more on defense instead of aggressive security issues. One of the defence methods to execute against growing aggressive security worries are the Honeypots[3]. This kind of defence mechanism work as Booby trap device set up as a vulnerability within a system to attract attackers and pick up all required security information used to reduce similar future attacks. For example Honey is one of the widely designed honeypots applied to detect and safeguard information contrary to network intruders. The planned architecture in this paper is built on Raspberry Pi Honeypot by means of current exiting kali linux tools and approaches such as Snort, Modern Honeypot Network (MHN), Kippo.

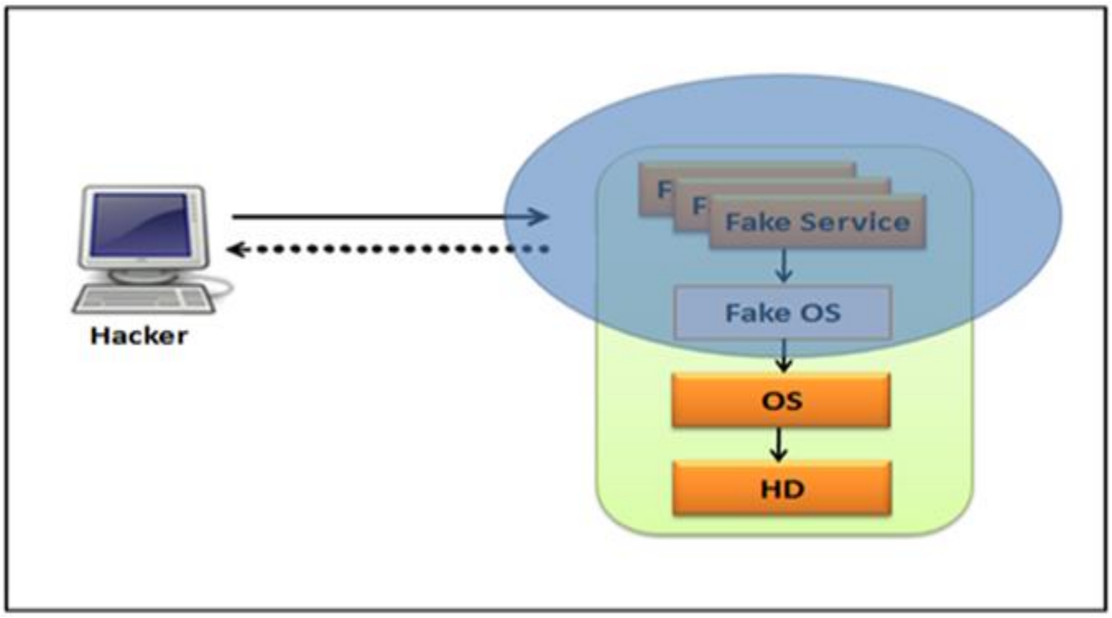

Figure 1. Sample honeypot in an organization.

Dionaea, Blastoff This manner it is extensively adapted because of its ease and cost effectiveness placement in any environment[4].

2. Problem Doman
Presently microservices and its application in containers mainly, has begun to be a mainstream architecture and technology for innovative IT infrastructures. It provides many improvements in variation to conventional physical or virtualized servers and along with new risks[5]. Thus we are facing a question, are container built systems secure adequate to be applied for the construction of honeypots that will be used against highly ambitious and sophisticated malicious actors.

3. System Architecture
Various authors previously researched container effectiveness. However, the promi-nence of the earlier research was to a greater extent on the effectiveness of the use of computing resources (CPU, memory, disk access or network connectivity) in comparison with the efficiency of execution on physical computers, whether virtual or another container implementation[6].

4. Honeypot Overview

Honeypot type main function is to gather as much information as potential about an attacker, both by keeping the attacker inside a "sandbox" or to answer in a particular way to the attacker's actions. Such a model is sustainable if it is effective enough, if it does not threaten its host operating system and, as one of the most important Honeypot types, regardless of the implementation, it needs to stay hidden to the attacker.

4.1 Honeypot Implementation

Honeypots as individual systems are employed either as physical computers or physical devices. They typically have greater bandwidth, but are more complicated to provide and the services are not active enough[7]. Virtual Honeypots are employed as virtual instances, whether as virtual computers or lately as micro services, the so-called containers.

4.2 Honeypot Model

Honeypots can also turn out to be dangerous for an organisation as honeypots are also deployed in the same network as the production or development as its goal is to get an estimate of the attackers that might try to penetrate in the system. Therefore using honeypots in a containerized environment is really required. Containerization also makes the operations of a honeypot easier for the network administrators.

5. Usage Organisation

5.1 Organisation Style for Real Systems

Organisation styles of honeypot is really important for the infrastructure as this defines the efficiency at which honeypot is deployed. Physical systems can be used but using only physical systems is not very efficient as they might be slow and network communication might be as good as cloud. Honeypot can be also implemented in a hybrid infrastructure to detect real life attack analytics[8].

5.2 Organisation Style for Systems on Cloud

Leveraging the cloud as a component of honeypot could turn out to be very useful because of the isolation capabilities of the cloud. Cloud also provides really good capabilities for scalability and fault tolerance. This gives network administrators flexibility to go for cloud while deploying honeypot or at least while a component of honeypot. The network and communication is enhanced by a lot of time which could be used to get more data for further analytics and visualisation. There are various cloud platforms such as AWS/Azure/GCP which services for containerization which a network administrator can use to deploy.

6. Honeypot

6.1 Definition

Honeypot are decoy system designed to collect and record data concerning intruder events in the network system. Honeypots are supplementary network security layer to the usual traditional internet security systems[5]. Generally, honeypots are unit within the firewall design or out of firewall factors in any deliberate hidden location in the network. In a perception, Honeypots can be thought as intrusion detection system alternatives which aim more in data gathering and misleading attackers to begin attacks. Honeypots are arranged using vacant IP address which is checked closely by network administrators. When the attacker open any attack or start collaboration with the system the Honeypot will flag the malicious activity and inform it. The main aim of the Honeypot is to gather data in such a way it will safeguard the system and network from any related attacks in the upcoming thus help administrators to patch any security liability in the network.

7. The Client Honeypot

Last 10 years have seen a lot of changes in the way attackers attack systems and how they try to analyse by making a lot of raw requests. Many times they try to attack browsers by using some browser exploits and media player exploits. These attacks are really important to be monitored as attack surface and attack vectors give an approach to security engineers and knowing the current patterns are really important to make infrastructures safe. This creates a high need for using honeypots in an architecture so that the government institutions and the organisations can understand what are the attacks which are getting more attention by the hackers in the wild. These attacks can be analysed later on and can be used to figure out which are the components attackers try to leverage first. This data can also be used later by machine learning models to create intrusion detection systems which are used to alert network administrators about attacks inside a network. There are some already existing applications which do this but the accuracy of those systems are very less and fail to follow the trend of the changing of methods and surfaces attackers use.

8. Honeypot in the Internet

Honeypot challenge measures the actual attack of a portable computer on the Internet. Honeypot is software that take the form of attractive services, the entire operating system, or perhaps the whole network, yet it is actually a tight-scale environment designed to attract the attacker, effectively avoiding out-siders in production systems for analying. Here the honeypots observes the file for each log and all the actions of the attacker. When any kind of attack, the honeycomb gives 2 things: firstly the data required to attack increases the immediate and appropriate reaction in the real time and secondly the time needed to enforce that reaction. During post-incident attack assessment, a pot of honeycombs may be used to study the attacker's interest to create a long-term strategic action line. With a variety of reaction functions, honey jars can be helpful in detecting internal abuse when viewed from a distance what should be done to draw "invaders". Honey jars can be used to express their effectiveness in 3 special ways: misleading, intimidating, or providing inquiry. To deceive, the honeycomb must provide so many affordable answers that the invader no longer suspects it of being a trap. To intimidate, a pot of bees will increase the level of access to a dangerous route ad in an unauthorized deception. For more information, Honeypot will allow critical attack signature data in a variety of security gear including IDS and Firewalls to lower the amount of fake alarms.

9. Containerization Based Virtualization

From a runtime point of view, a container is a set of limited application processes across the system and other containers. This is fulfilled through two features of Linux kernel: (i) Word spaces and (ii) control groups. Word spaces make global resources the way a process group can see and use a single set of resources while the other group can use a different set of resources. Teams provide a way to integrate and classify task sets, as well as all of their future children, into groups with special behavioral sequences. Allows you to schedule and distribute system resources according to category.

9.1 Platforms in the Containers

There are many field solutions, all depending on collections and word spaces. Therefore, all platforms have the same options and full functionality. The ideology of using the plurals of widely used field structures LXC and Docker differs.

9.2 Platforms in the Real Time Scenarios

The term real-time suggests that the accuracy of the gadget depends on it now which is not very effective in the calculation results yet also in the time when the results are produced. Real-time buildings can be divided into three companies: solid, durable and soft real-time. Losing a deadline for a solid real-time machine can lead to disastrous results, while a lack of a real-time lock on a solid real time device results in all the lack of end-result software.

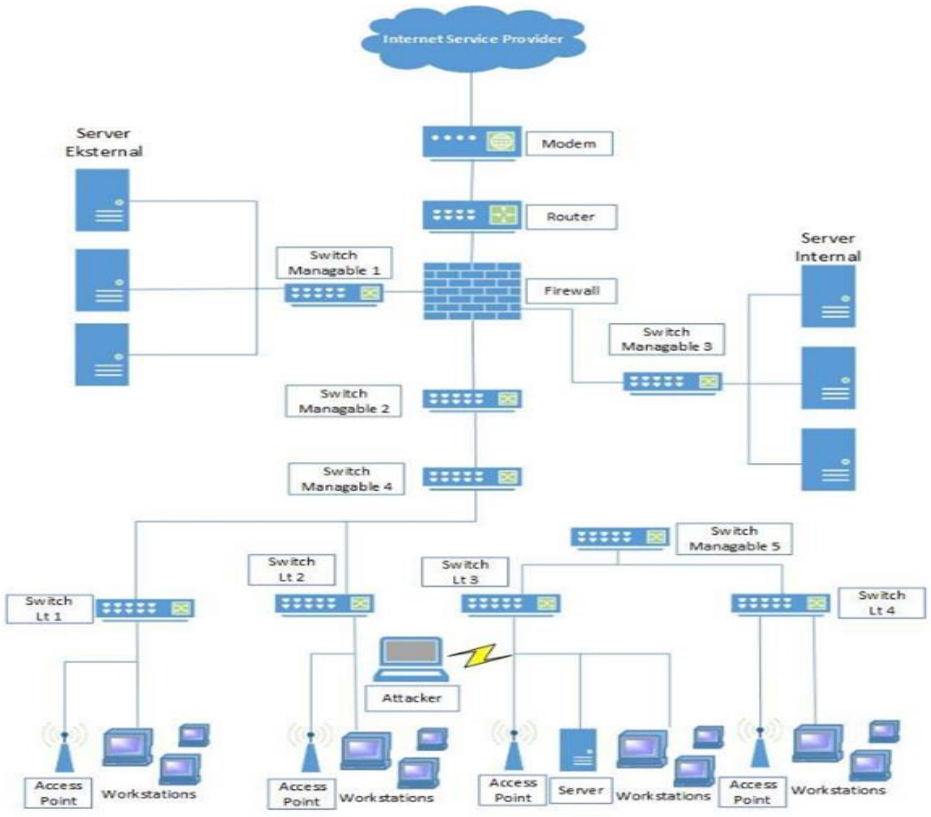

Figure 2. A honeypot on the internet to detect attack.

10. Leaking Ip Address

This paper tries to leak the IP address of the attacker in case the attacker is using TOR browser because TOR browser tries to hide the IP of the attacker using different nodes of the tor foundation which are present around the world. The request is intercepted by at least 3 nodes of the tor foundation and then forwarded to the server so that there are very less chances of the leakage of the origin of the attacker. There are some ways to detect the origin of the attacker. 1) First approach would be where the organisation adds all the nodes of the tor foundation so that it will become possible to figure out the end-to-end communication of the requests and response which will in turn leak the actual IP address of the attacker, but that approach would be very costly to be implemented in real life and exists just in theory. 2) Another approach would be to make the browser some DNS requests to its own DNS servers because TOR browsers do not make DNS requests through the tor nodes and the origin could be logged in the databases of the honeypot. 3) The last approach would be to send a client a attacker which should seem as a sensitive file to the attacker. Using that file a honeypot operator can try to leak the origin of the attacker by making a request to the operated hosted web server using some command line tool such as curl.

11. Results

If the attacker tries to attack the honeypot thinking that it is some system containing sensitive information then the application will log information about the attacker that it can retrieve from the request such as type of browser and origin of the attacker and push it to the API server which is hosted on cloud. This data can be retrieved from the cloud using the same API server using `/retrieve` endpoint. The backend will push the details about the attacker to the same API server which inturn will send the results to the data-base on the cloud. The system will also detect any kind of fuzzing and will report it to the administrator. Fuzzing will have a different area of effect and will be reported separately from the normal requests. The application will check for fuzzing by detecting common endpoints used by fuzzing tools. The tool used for fuzzing will also be reported. One prob when the user uses the TOR browser to check for the honeypot. In this scenario the origin of the attacker will not get leaked as the attacker will be masked behind the TOR architecture routers. To solve this problem the application ships a binary to the attacker which if the attacker runs, even in an isolated or containerized environment, will leak the IP address of the attacker and then the origin can be registered to the cloud which can be then fetched using API server for further process or enquiries. The application itself is a docker container and can be launched in any system with docker installed. The container includes all the dependencies so there is no problem installing dependencies and dealing with their versions. The application will take 15-20 secs to launch and reach a stage where it is fully functional for receiving requests. The application can be deployed on any cloud platform such as AWS/AZURE/GCP/Digital Ocean. The size of the database should be provisioned as per the expected requests for the honeypot. Similarly the compute capacity of the honeypot should be provisioned and scaled based on the expected incoming requests/connections. The entire analytics of all the types of requests can be seen on the dashboard. Below are the tables depicting the results from API server.

Table 1. Fuzzing Detection.

Time	User-Agent	Fuzzing Detected	Origin Leaked	Session_Id
15-07-2022 12:00:00	Mozilla Firefox	False	False	32234234
14-07-2022 11:23:00	Google Chrome	True	False	78432684
12-07-22 10:11:12	Curl	True	True	63832952
06-05-22 09:14:11	Gobuster	True	False	82020838
02-03-21 03:28:53	Dirb	True	True	56230482

Table 2. Origin and Session Id.

Time	Session_Id	Origin
12-07-22 10:11:12	32234234	118.151.210.81
14-07-2022 11:23:00	78432684	128.23.28.21
15-07-2022 12:00:00	56230482	73.26.123.72
02-03-21 03:28:53	63832952	12.21.33.34

12. Future Work

The future work of this project includes updating the dashboard which will enable the user to perform and visualise different kinds of analytics.

- Strengthening the fuzzing detection system to detect the fuzzing with more accuracy.
- Add alerts to the honeypot if the database is getting exhausted.
- Add alerts to the honeypot if the number of requests are more than what the current configuration can handle.
- Work on temporary local temporary storage in case the internet is not working at some point of time so that we can store it on the cloud later without losing data.
- Revise the APIs in order to improve the efficiency of the system.

13. Conclusion and Discussion

The fundamental container characteristics are high efficiency, small footprint and high flexibility, which makes them appropriate for creating an infrastructure that can be set up or disassembled quickly.

Although Docker containers are deemed secure, the simple implementation which comes with previously prepared Dockerfile files or GitHub images does not fulfill the security needs for building a Honeypot infrastructure, e.g. Cowrie Dockerfile or other existing images. An extra problem is that Honeypot's aim is to appeal to malicious users, who generally have a very good understanding of systems penetration, so it is to be likely that they will use all the potential weaknesses of the basic container installations. Docker actually supports powerful security mechanisms, if properly configured: Capabilities, Cgroups, MAC (AppArmor or SeLinux) are very powerful mechanisms, but they all need knowledge for their proper configuration.

Some vulnerabilities can be resolved by using the best practices:

- Application of verified images
- Application of minimal OSs (Atomic, CoreOS, Vmware Photon)
- Isolation at the hypervisor level
- Immunisation of capabilities
- Decrease of resources
- Startup in userspace
- Startup of applications in userspace within Docker
- Using NIST best practices
- Updating the host operating system

Alarming is the amount of information uncovered by Docker containers to the user, so it is easy for an attacker with the correct knowledge of the host operating system to deal with these vulnerabilities and abuse them, even in an computerized manner motivated by VM detection approaches used by malware. Having in mind the recognised deficiencies of the Honeypot applications using basic Docker, future research should suggest a model which would remove all known inadequacies addressed in this article, organised with a safe manner of orchestration of containers for the purpose of making a Honeypot infrastructure built on containers.

References

[1] S. A. Budiman, C. Iswahyudi, and M. Sholeh, “Implementasi Intrusion Detection System (IDS) Menggunakan Jejaring Sosial Sebagai Media Notifikasi,” Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST) , ISSN: 1979-911X Yogyakarta, 15 November 2014.

[2] Research infrastructures action, Sixth framework program, D1.1: Honeypot Node Architecture, page 7-24.

[3] C. Kaewkasi., K. Chun Mun Wong, ―Improvement of container scheduling for Docker using Ant Colony Optimiza-tion. ‖ Proc. Int. Conf. Knowledge and Smart Technology (KST).

[4] Provos N. and Holz T. 2007, Virtual Honeypots : From Botnets Tracking to Intrusion Detection, Addision Wesley Professional.

[5] The Honeypot Project, Know Your Enemy: Revealing the Security tools, tactic, and motives of Blackhats communi-ty.2002.
[6] V. Haldar, D. Chandra, and M. Franz, “Dynamic taint propagation for java,” in ACSAC ’05: Proceedings of the 21st Annual Computer Security Applications Conference, 2005, pp. 303–311.
[7] J. Mirkovic, S. Dietrich, P. Reiher “Internet denial of service: attack and defense mechanisms”, Prentice Hall Profes-sional Technical Reference, 2005.
[8] Savita Paliwal, “Honeypot: A Trap for Attackers:,IJARCCE, Vol. 6, Issue 3,2017.