Faculty of Mathematics and Computer Science

Heidelberg University

Master thesis

in Computer Science

submitted by

Stefan Machmeier

born in Heidelberg

2021

Honeypot Implementation

in a

Cloud Environment

This Master thesis has been carried out by Stefan Machmeier

at the

Engineering Mathematics and Computing Lab

under the supervision of

Herrn Prof. Dr. Vincent Heuveline

(Titel der Masterarbeit - deutsch):

(Title of Master thesis - english):

Acknowledgements

Contents

Acronyms

ADB	Android Debug Bridge
ADC	Application Delivery Controller
ASA	Adaptive Security Appliance
AWS	Amazon Web Services
CERT	Computer Emergency Response Team
DaaS	Data-as-a-Service
DDoS	Distributed Denial of Service
DICOM	Digital Imaging and Communications in Medicine
DTK	Deception Toolkit
EU	European Union
FHIR	Fast Healthcare Interoperability Resources
GCP	Google Cloud Platform
HaaS	Hardware-as-a-Service
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure-as-a-Service
ICS	Industrial control systems
IDS	intrusion detection system
IOCTA	Internet Organised Crime Threat Assessment
IPD	intrusion prevention system
IPP	Internet Printing Protocol
NIST	National Institute of Standards and Technology
NSM	network security monitoring
OS	operating system
PaaS	Platform-as-a-Service

RDP	Remote Desktop Protocol
SaaS	Software-as-a-Service
SCADA	Supervisory Control and Data Acquisition
VM	virtual machine
VMM	virtual machine monitors

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem description

Due to the pandamic, the last two years kept us at home, and we were tempted to use the Internet more often. Recent stastics of the monthly in-home data usage in the United States from January to March 2020 showed a drastic increase compared to the years before [61]. Even Europol (an agency that fights against terrorism, cybercrime, and other threats [27]) rose awareness of new cyber threats related to an increase of misinformation. As stated in their yearly Internet Organised Crime Threat Assessment (IOCTA), citizens and businesses are looking for any kinf of information that is desperately needed. Both contributes to cybercriminal acts. [26]

The increase of cyberattacks is merely connected to the pandamic, also fast growing technology comes along with new security concerns. Especially in cloud computing due to access to large pools of data and computational resources, controlling access to services is becoming a tougher challenge nowadays. Besides traditional security leverages such as firewalls or intrusion detection systems, one known methodology to strengthen infrastructures is to learn from those who attacks it. Honeypots are a security resource whose value lies in being probed, attacked, or compromised [60]. By getting attacked from others, zero-day-exploits, worm activity, or bots can be detected. In retrospect, this helps to adapt, or fix infrastructures before more damage occurs. As a cloud provider, it is a crucial point if and how attacks on production server could have been prevented. Considering the Global Security Report by Trustwave, the amount of attacks doubled in 2019, and increased by 20% in 2020 [13]. It puts cloud providers to the third most targeted environments for cyber attacks. Corparate and internal networks are on top of this ranking. However, this rise of attacks on cloud infrastructures has shown the importance why more security will be necessary in the future. Thus, making it a thrilling challenge for the upcoming years.

1.2 Justification, motivation and benefits

1.3 Research questions

The following research questions have been answered in this thesis:

1. How can honeypots contribute to a more secure cloud environment including baiting adversaries to our honeypots, and controlling their requests?
2. What is a preferable way to handle data management and visualization?
3. How can we analyze our data to get more information?

1.4 Limitations

Heidelberg offers a cloud service, called "HeiCloud". Since we tailor our implementation for this service. Moreover, it ecists a vast variety of different honeypots which bound us to a very few of them.

Chapter 2

Background

Using honeypots in a cloud environment merge two varying principals together. This chapter concludes the fundamental knowledge that is needed to comprehend the upcoming experiments. If the reader has a profoundly understanding of cloud computing, honeypots, and virtualization he can skip this chapter.

2.1 Virtualization

Virtualization, often refered to virtual machine (VM), is defined by Kreuter [41] as "an abstraction layer or environment between hardware component and the end-user". A VM runs on top of an operating system (OS) core components. By an abstraction layer, the virtual machine is connected with the real machine by hypervisors or virtual machine monitors (VMM). Hypervisors can use real machine hardware components, but also support virtual machine operating systems and configurations. Both are similar to emulators, that are defined by HA [31] as "process whereby one computer is set up to permit the execution of programs written for another computer". This allows to management multiple VM with real machine resources. There are three different types of virtualization, (i) software virtual machines, (ii) hardware virtual machines, and (iii) virtual OS/containers. Software virtual machine's manage interactions between the host operating system and guest operating system. Hardware virtual machine's offers direct and fast access to the underlying resources. It uses hypervisors, modified code, or APIs. Virtual OS/container partitions the host operating system into containers or zones. [23]

2.2 Cloud Computing

Nowadays it is one of the well-known keywords and has been used by vary large companies such as Google, or Amazon, however, the term "cloud computing" dates back to the late 1996, when a small group of technology executives of Compaq Computer framed new business ideas around the Internet.[51] Starting from 2007 cloud computing evolved into a serious competitor and outnumbered the keywords “virtualization”, and “grid computing” reported by Google trends [68]. Shortly, various cloud provider become publicly available, each with their own strengths and weaknesses. For example IBM’s Cloud[1], Amazon Web Services[2], and Google Cloud[3]. Why are clouds so attractive in practice?

- It offers major advantages in terms of cost and reliability. When demand is needed, consumers do not have to invest in hardware when launching new services. Pay-as-you-go allows flexibility.
- Consumer can easily scale with demand. When more computational resources are required due to more requests, scaling up instances in conjunction with a suited price model are straightforward.
- Geographically distributed capabilities supply the need for world-wide scattered services.

2.2.1 Definition of Cloud Computing

Considering the definition of Brian Hayes, cloud computing is “a shift in the geography of computation” [32]. Thus, computational workload is moved away from local instances towards services and datacenters that provide the need of users [16].

Considering the definition of the National Institute of Standards and Technology (NIST), cloud computing “is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [42]. NIST not only reflects the geographical shift of resources such as datacenters, but also mentions on-demand usage that contributes to a flexible resource management. Morever, NIST composes the term in five essential characteristics, three service models (see subsection 2.2.2), and four deployment models (see subsection 2.2.3) [42]

On-demand-self-service refers to the unilaterally provision computing capabilities. Consumers can acquire server time and network storage on demand without a human interaction.

Broad network access characterizes the access of capabilities of the network through standard protocols such as Hypertext Transfer Protocol (HTTP). Heterogeneous thin and thick client platforms should be supported.

Resource pooling allows the provider’s computing resources to be pooled across several consumers. A multi-tanent model with different physical and virtual resources

[1] `https://www.ibm.com/cloud`
[2] `https://aws.amazon.com/`
[3] `https://cloud.google.com/`

are assigned on demand. Other aspects such as location are independent and cannot be controlled on a low-level by consumers. Moreover, high-level access to specify continent, state, or datacenter can be available.

Rapid elasticity offers consumers to extend and release capabilities easily. Further automization to quickly increase resources when demand skyrockets significantly can be supported regardless limit and quantity at any time.

Measured service handles resources in an automated and optimized manner. It uses additional metering capabilities to trace storage, processing, bandwith, and active user accounts. This helps to monitor, and control resource usage. Thus, contributing to transparency between provider and consumer.

2.2.2 Service models

Service models are categorized by NIST into three basic models based on usage and abstraction level. Figure 2.1 shows the connection between each model whereas cloud resource are defined in subsection 2.2.3. Due to vast range of functionalities, Infrastructure-as-a-Service (IaaS) builds the foundation of service models. Each model on top represents a user-friendly abstraction with derated capabilities. Table 2.1 shows examples of such service models.

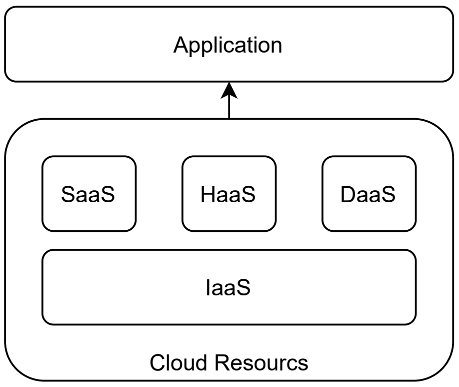


Figure 2.1: Abstract visualization of service models. The lowest level within the container “cloud resources” represents the depth of functionalities. Therefore, Infrastructure-as-a-Service (IaaS) offers the most functionalities whereas the others have a user-friendly abstraction.

Software-as-a-Service (SaaS) is a high-level abstraction to consumers. Controlling the underlying infrastructure is not supported. Often provider uses a multi-tenancy system architecture to organize each consumer’s application in a separate environment. It helps to employ scaling with respect to speed, security availability, disaster recovery, and maintenance. Main objective of SaaS is to host consumer’s software or application that can be accessed over the Internet using either a thin or rich client. [24] “Limited user-specific application configuration settings” can be made [42].

Platform-as-a-Service (PaaS) pivots on the full “Software Lifecycle” of an application whereas SaaS distincts on hosting complete applications. PaaS offers ongoing development and includes programming environment, tools, configuration management, and other services. In addition, the underlying infrastructure is not managed by the consumer. [42]

Infrastructure-as-a-Service (IaaS) offers a low-level abstraction to consumers with the ability to run arbitrary software regardless of operating system or application. In contrast to SaaS, IT infrastructures capabilities (such as storage, networks) can be used. It strongly depends on virtualization due to integration, or decomposition of physical resources. [42]

Data-as-a-Service (DaaS) serves as a virtualized data storage service on demand. Motivations behind such services could be upfront costs of on-premise enterprise database systems. [24] Mostly they require “dedicated server, software license, post-delivery services, and in-house IT maintenance” [24] Whereas DaaS costs solely what consumer’s need.When dealing with a tremendous amount of data, file systems and RDBMS often lack in performance. DaaS outruns such weak links by employing a table-style abstraction that can be scaled.[24]

Hardware-as-a-Service (HaaS) offers IT hardware, or datacenters to buy as a pay-as-you-go subscription service. The term dates back to 2006 during a time when hardware virtualization became more powerful. It is flexible, scalable and manageable.[68]

Table 2.1: Providers of cloud service models

SaaS	**PaaS**	**IaaS**	**Daas**	**HaaS**
Google Mail	Google App Engine	HeiCloud	Adobe Buzzword	Amazon EC2
Google Docs	Windows Azure	Amazon EC2	ElasticDrive	Nimbus
Microsoft Drive	AWS Elastic Beanstalk		Google Big Table	Enomalism
			Amazon S3	Eucalyptus
			Apache HBase	

2.2.3 Deployment models

Deployment models are categorized by NIST into four basic models. Each differs in data privacy, location, and manageablility [42]. Table 2.2 shows examples of such deployment models.

Table 2.2: Examples for cloud deployment models

Private Cloud	**Community Cloud**	**Hybrid Cloud**	**Public Cloud**
Seafile	Seafile	IBM Cloud	Amazon EC2
Nextcloud	Nextcloud	Amazon EC2	Google AppEngine

Private clouds offer the highest level of control in regard of data privacy, and utilization. Mostly, such clouds are deployed within in a single organization, either managed by in-house teams or third party suppliers. In addition, it can be on or off premise. Within private clouds consumers have full control of their data. Especially for European data privacy laws, it is not negligible when data is stored abroad and thus under law of foreign countries. However, the popularity has not been withdrawn due to immense costs when moving towards public clouds. [24, 42]

Community clouds can be seen as a conglomerate of mutliple organizations that merge their infrastructure with respect to a commonly defined policy, terms, and condition beforehand. [42]

Public clouds represents the most used deployment models. In contradiction to private one, public clouds are fully owned by the service provider such as business, academics, or government organization. Consumers do not know where their data is distributed. In addition, contracts underlie custom policies. [42]

Hybrid cloud is a mixure of two or more cloud infrastructures, such as private and public cloud. However, each entity keeps its core element. Hybrid clouds defines "standardized or propriertary technology to enables data and application portability"[42].

2.3 Honeypots

The term "honeypot" exists since more than a decade. 1997 was the first time that a free honeypot solution became public. Deception Toolkit (DTK), developed by Fred Cohen, released the first honeypot solution. However, the earliest drafts of honeypots are from 1990/91, and built the foundation for Fred Cohen's DTK. Clifford Stoll's book "The Cuckoo's Egg"[62], and Bill Cheswick's whitepaper "An Evening With Berferd"[20] describe concepts that are consider nowadays as honeypots.[60]

A honeypot itself is a security instrument that collects information on buzzing attacks. It disguises itself as a system, or application with weaklinks, so that it gets exploited and gathers knowledge about the adversary. In 2002 a Solaris honeypot helped to detect an unknown dtspcd exploit. Interestingly, a year before in 2001 the Coordination Center of Computer Emergency Response Team (CERT), "an expert group that handles computer security incidents"[29], shared their concerns regarding the dtspcd. Communities were aware that the service could be exploited to get access and remotely compromise any Unix system. However, during this time such an exploit was not known, and experts did not expect any in the near future. Gladly, early instances based on honeypot technologies could detect new exploits and avoid further incidents. Such events lay emphasis on the importance of honeypots.

2.3.1 Definition of a Honeypot

Dozen of defintions for honeypots circulate through the web that causes confusion, and misunderstandings. In general, the objective of a honeypot is to gather information about attacks, or attack patterns [45]. Thus, contributing as an additional source of security measure. See subsection 2.3.3 for a detailed view regarding honeypots in the security concept. As Spitzner [60] has listed, most misleading defintions are: honeypot is a tool for deception, it is a weapon to lure adversaries, or a part of an intrusion detection system. In order to get a basic understanding, we want to exhibit some of the key definitions. Spitzner [60] defines honeypots as a "security resource whose value lies in being probed, attacked, or compromised". Independent of its source (e.g. server, application, or router), we expect that our instance is getting probed, attacked, and eventually exploited. If a honeypot does not match this behaviour, it will not provide any value. It is important to mention that honeypots do not have any production value, thus, any communication that is acquired is suspicious by nature [60]. In addition, Spitzner Spitzner [60] points out that honeypots are not bounded to solve a single problem, hence, they function as a generic perimeter, and fit into different situation. Such functions are attack detection, capturing automated attacks, or alert/warning generator. Figure 2.2 show an example how honeypots could be used in an IT infrastructure.

In general, we differentiate two types of honeypots (i) Production honeypots (ii) Research honeypots. This categorization has their origin from Mark Rosch developer of Snort during his work at GTE Internetworking.

Production honeypots are the common type of honeypots everyone would think of it. The objective is to protect production environments, and to mitigate the risk of attacks. Normally, production honeypots are easy to deploy within an organization. Mostly, low-interaction honeypots are chosen due to a significant reduce in risk. Thus, adversaries might not be able to exploit honeypots to attack other systems. Downside is a lack of information. Standard information like the origin of attacks, or what exploits are used can be collected, whereas insides about communication of attackers, or deployment of such attacks are unlikely to obtain. In contrast, research honeypots do fulfill this objective.[60]

Research honeypots are used to learn more in detail about attacks. The objective is to collect information about the clandestine organizations, new tools for attacks, or the origin of attacks. Research honeypots are unlikely for production environments due to a higher increase of risk. Facing an increase in deployment complexity, and maintenance does not attract a production usage.[60]

It is worth to mention that there is no exact line between research or production honeypots. A possible cases are honeypots that could function as either an production or an research honeypot. Due to their dynamic range in which they are applicable it makes it hard to distinguish.

Provos [50] adds an additional differentiation for the virtual honeypot framework and splits it into the following types:

- Physical honeypots are "real machines on the network with its own IP address" [50]
- Virtual honeypots are "simulated by another machine that responds to network traffix sent to the virtual honeypot" [50]

2.3.2 Level of Interaction

When building and deploying a honeypot, the depth of information has to be defined beforehand. Should it gather unauthorized activities, such as an NMAP scan? Do you want to learn about buzzing tools and tactics? Each depth brings a different level of interaction because some information depends on more actions of adversaries. Therefore, honeypots differ in level of interaction.

Low-interaction honeypots provide the lowest level of interaction between an attacker and a system. Only a small set of services like SSH, Telnet, or FTP are supported which contributes to the deployment time. In terms of risk, a low-interaction honeypot does not give access to the underlying OS which makes it safe to use in a production environment. For example using an SSH honeypot, such services are emulated, thus, attackers can attempt to login by brute force or by guessing, and execute commands. However, the adversary will never gain more access because it is not a real OS. However, safety comes with the downside of less information. Collected is limited for statistical purpose such as (i) Time and data of attack (ii) Source IP address and source port of the attack (iii) Destination IP address and destination port of the attack. Transactional information can not be collected. [60]

A medium-interaction honeypot offers more sophisticated serivces with higher level of interaction. It is capable to respond to certain activities. For example a Microsoft IIS Web server honeypot could be able to respond in a way that a worm is expecting it. The worm would get emulated answers, and could be able to interact with it more in detail. Thus, gathering more severe information about the attack, including privilege assessment, toolkit capturing, and command execution. In contrast, medium-interaction honeypots allocate more time to install and configure. In addition, more security checks have to be perform due to a higher interaction level than low-interaction honeypots. [60]

High-interaction honeypots are the highest level interation. Mostly, they represent a real OS to provide a full set of of interactions to attackers. They are so powerful because other production servers do not differ much to high-interaction honeypots. They represent real systems in a controlled environment. Obviously, the amount of information is tremendous. It helps to learn about (i) new tools (ii) finding new bugs in the OS (iii) the blackhat community. However, the risk of such a honeypot is extremely high. It needs severe deployment and maintenance processes. Therefore, it is time consuming.

2.3.3 Security concepts

Security concepts are classified by Schneier [58] in prevention, detection, and reaction. Prevention includes any process that (i) discourages intruders and (ii) hardens systems to avoid any kind of breaches. Detection scrutinize the identification of attacks that threatens the systems (i) confidentiality (ii) integrity and (iii) availability. Reaction treats the active part of the security concept. When attacks are detected, in conducts reactive meassures to remove the threat. Each part is designed to be sophisticated so that all of them contribute to a secure environment. [45]

Honeypots contribute to the security concept like firewalls, or intrusion detection system (IDS). Regarding prevention, honeypots add only a small value because security breaches cannot be identified. Moreover, attackers would avoid wasting time on honeypots and go straight for production systems instead.

However, detection is one of the strengths of honeypots. Attacks often vanish in the sheer quantity of production activities. If any connection is obtain to a honeypot it is suspicious by nature. In conjunction with an alerting tool, attacks can be detected.

Honeypots strongly supply reaction tools due to their clear data. In production environments, finding attacks for further data analysis are not easy to grasp. Often data submerge with other activities which complicates the process of reaction. [45] Nawrocki et al. [45] distinct honeypots from other objectives such as firewall, or log-monitoring.

Table 2.3: Distinction between security concepts based on areas of operations (derived from [45]).

Objective	**Prevention**	**Detection**	**Reaction**
Honeypot	+	++	+++
Firewall	+++	++	+
Intrusion Detection Sys.	+	+++	+
Intrusion Prevention Sys.	++	+++	++
Anti-Virus	++	++	++
Log-Monitoring	+	++	+
Cyber Security Standard	+++	+	+

2.3.4 Value of Honeypots

To assess the value of honeypots we want to take a closer look to thier advantages and disadvantes.[44, 37, 60]

Advantages

- Data Value: Collected data is often immacluate and does not contain noise from other activities. Thus, reducing the total size of data, and speed up the analyzation.
- Resources: Firewalls, and IDS are often overwhelmed by the gigbits of traffic, thus, dropping network packets for analyzation. This results in a far less effictive detection for malicious network activities. However, honeypots are independent of resources because they only capture their activities at itself. Due to resource limitation, expensive hardware is not needed.
- Simplicity: A honeypot do not require any complex algorithms, or databases. It should be able to quickly deploy it somewhere. Research honeypots might come with a certain increase of complexity. However, if a honeypot is complex, it will lead to misconfigurations, breakdowns, and failures.
- Return on Investment: Capturing attacks immediately informs users that attacks occur on the infrastructure. This helps to demonstrate their value, and contributes to new investment in other security measurements.

In addition, Nawrocki et al. [45] listed four more advantages of honeypots:

- Independent from Workload: Honeypots only process traffic that is direct to them.
- Zero-Day-Exploit Detection: It helps to detect unknown strategies and zero-day-exploits.

- Flexibility: Well-adjusted honeypots for a variety of specific tasks are available.
- Reduced False Positives and Negatives: Any traffic or connection to a honeypot is suspicious. Client-honeypots verify such attacks based on system state changes. This results in either false positive, or false positive.

Disadvantages

- Narrow Field of View: Only direct attacks on honeypots can be investigated whereas attacks on production system are not detect by it.
- Fingerprinting: A honeypot often has a certain fingerprint that can be identified by attackers. Especially commerical ones can be detected by their responses or behaviours.
- Risk to the Environment: Using honeypots in an environment always increase the risk. However, it depends on the level of interaction.

2.3.5 Honeynets

Instead of having single honeypots that can be attacked, a honeynet offers a complete network of standard production systems such as you would find in an organization. Those systems are high-interaction honeypots, thus, allowing to fully interact with the OS, and applications. Key idea is that an adversary can probe, attack, and exploit these systems so that we can derive interaction within this network. It should be noticed that a honeynet have to be protected by firewall. Figure 2.3 represents such a honeynet within an organization.

In comparison to regular honeypot, the greatest value of honeynets is the usage of true production systems. Black hats often do not know that they attact a honeynet, thus, adding value to prevention. However, the downsides are high complexity and maintenance that is needed to keep a honeynet running. [60]

2.3.6 Legal Issues

Considering questions related to legal issues of honeypots can easily exceed this thesis. In this regard, we restrict us to the country we reside in, so that, we are only concerned about the European Union (EU) regulations, EU directives, and international agreements. Honeypots collect (i) content data that is used for communication, and (ii) transcational data that is used to establish the connection. Sokol et al. [59] studied the legal conditions for data collection and data retention. They come to the conclusion that administrators of honeypots have a legal ground of legitimate interest to store and process personal data, such as IP addresses. Moreover, for production honeypots the legitimate interest is to secure services. Regarding the length of data retention, the principle of data minimization has to be considered which means there is no clear answer for it. Any published data of research honeypots needs to be anonymized.

2.4 Related Work

The Snort [19] inline extension *Bait'n'Switch* [14] redirects attackers from production systems to honeypots. Usually Snort drops malicious packets when they are detected, and creates a specific rule to block malicious activity from the source. In contrast, *Bait'n'Switch* does not create a rule to block more request, instead it redirects them to honeypots. The attacker does not realize that all packets a rerouted, and that the original target has changed. The IDS is bounded to its signature database and can only redirect know attakcks.[47]

Whereas *Bait'n'Switch* had to drop the first attack which leads to a suspicious behaviour for attackers, the *Intrusion Trap System* [63] is capable of forwarding the first request. Therefore, the first recognized attack by the IDS can be forwarded and answered by honeypots. [47]

The *honeyd* [50] extension *Honeycomb* [40] enables automatically generates *Snort* and *Bro* [48] signatures for all incoming traffic. In addition, new signatures are created for similar patterns if they do not exist already, and updating of existing ones to improve their quality. This is based on the incoming traffic, and the corresponding attack session. Even mutations of attacks are considered. It generates a more generic description for signatures in order to match the original attack, and the mutation. It helps to keep the size of signatures small. However, the downside of this extension is the missing verification of attacks. Wrongly redirected traffic will not be proofen if an attack was successful eventhough a signature has been created. [47]

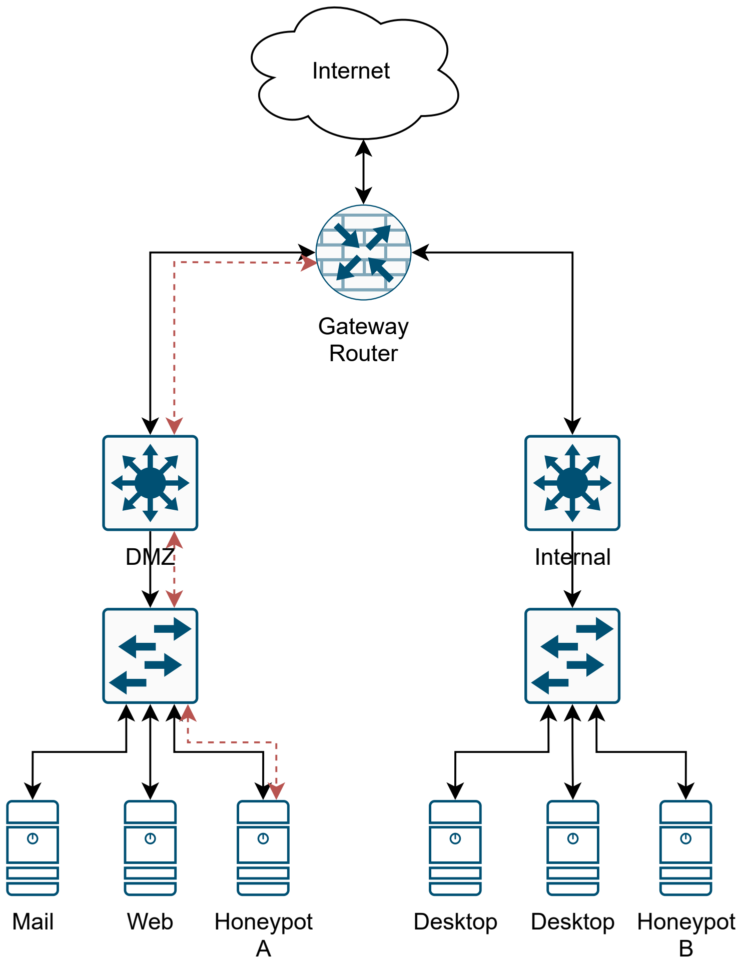


Figure 2.2: Example of honeypots in a simplified network (derived from [60]). Each of the demilitarized zone, and internal network are separated by a router and a Layer-3 switch. As dervied from above, in each network a honeypot is available (honeypot A, B). The read path symbolises the path of an attacker coming from the gateway router.

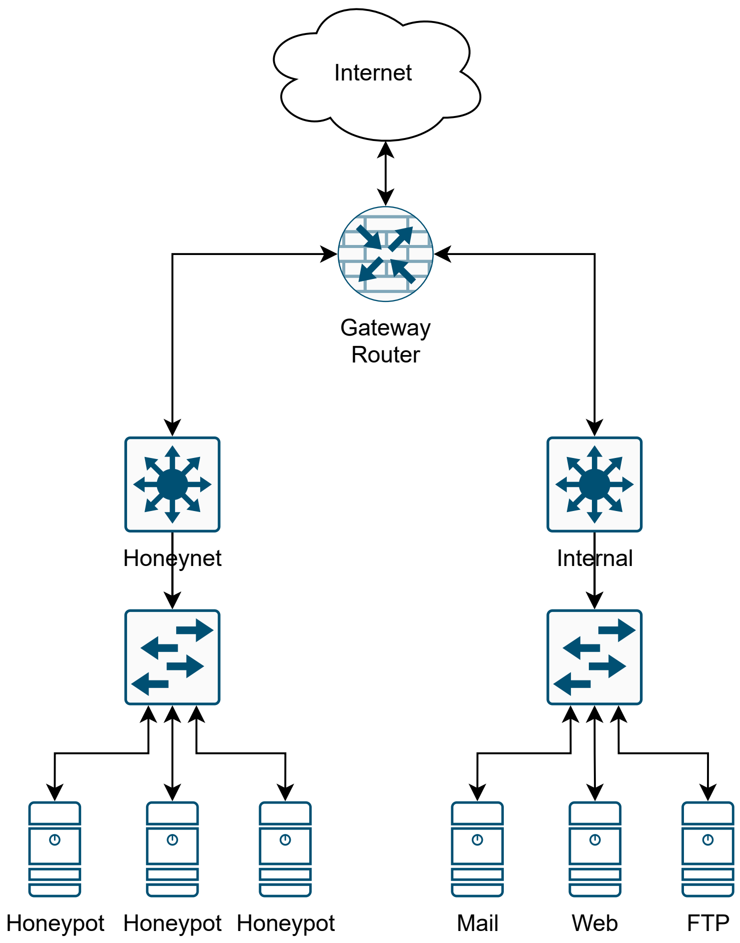


Figure 2.3: Example of honeynets in a simplified network (derived from [60]). On the right, this network presents the honeynet consisting of several other honeypots. On the right the network presents a ordinary subnet consisting of mail, web, and FTP server.

Chapter 3

Cloud Security with Honeypots

In this chapter we investigate... tbc.

3.1 Introduction

As previously mention in section 2.2, Cloud Computing, using cloud resources are becoming the go-to option for new services, and applications. Kelly et al. [38] thoroughly investigated honeypots on cloud providers such as Azure, Amazon Web Services (AWS), or Google Cloud Platform (GCP). Followingly, we present their results that we want to compare with the results of heiCLOUD later on. The results are collected by T-Pot version 20.06.0. over a duration of 3 weeks. In addition, Kelly et al. [38] considered different server geographical locations. They have collected data from East US, West Europe, and Southeast Asia. Table 3.1 shows the results presented by Kelly et al. [38]. Dionaea (a honeypot to capture malicious payload), Cowire (SSH and Telnet honeypot), and Conpot (industrial honeypot for Industrial control systems (ICS), and Supervisory Control and Data Acquisition (SCADA)) are the most attacked honeypots in comparison to the others. Regarding AWS, Dionaea account 91% of the total attacks, Glutton and Cowire are minor with 5%, and 2%. Interestingly, Cowire reported several attacks related to the COVID-19 pandemic to enable social engineering methods. In contrast to AWS, Cowire logged the majority of attacks with 51% on GCP. Beside several automated attacks trying to login with default credentials, adversaries tried to gather information about CPU architecture, scheduled tasks, and privilege escalation. Microsoft Azure reflects nearly the same results as the other two cloud providers beforehand.

The overall results show an average ratio of 55.000 attacks per day, summing up to roughly 1.6 mio in total. Similar results for different regions could have been reprocuded. Their results clearly show the disparity of the regions Europe, US, and Asia. An important question that Kelly et al. [38] answered is if attackers target services on cloud providers based on the cloud providers' market share. They could not confirm this assumption based on the fact that Google Cloud with the smallest market share received most of the attacks. In total, most of the attacks are originated from Vietnam, Russia, United States, and China. Due to technologies

Table 3.1: Overview of attacks on cloud providers. For a better overview, only the three most attacked honeypots are listed. The others combine several honeypots.

Provider	Honeypot				In Total
	Dionaea	**Cowire**	**Glutton**	**others**	
Amazon Web Services	228.075	4.503	11.878	3.688	248.144
Google Cloud Platform	162.570	297.818	84.375	36.403	581.116
Microsoft Azure	308.102	9.012	17.256	6.365	340.735

such as VPN, or Tor, the geolocation only indicates the last node, so location data might be distorted. Across all providers roughly 80% of the source IP addresses had a bad reputation and could been filtered by the organization. The operating devices used for attacking the services are mostly Windows 7 or 8, and different Linux kernels and distributions. Windows devices target vulnerabilities in remote desktop sharing software. Such vulnerabilities are (i) CVE-2006-2369[3] (RealVNC) in the US region, (ii) CVE-2001-0540[1] (RDP) in EU and Asia regions, (iii) CVE-2012-0152[4] (RDP) in the Asia region, and (iv) CVE-2005-4050[2] (VoIP) in EU region. In addition, attackers were also capable of disguising any fingerprinting activity of p0f.

In this chapter, we want to compare the findings Kelly et al. [38] claimed in the paper "A Comparative Analysis of Honeypots on Different Cloud Platforms" with ours using the University Heidelberg's cloud solution. First, a short introduction of heiCLOUD is held, followed by a closer lookup of T-Pot that is used to acquire data. Lastly, we present the results and do a thorough comparison closing up with a discussion based on a technical report of the Campridge University.

3.2 Methodology

Our foremost goal is to track as most attacks as possible. To gather various attacks from the Internet Figure 3.1 sketches our concept we plan to apply. Honeypots should be deployed on a single instance, and store their data, or log files in a database. By the help of data visualization tools we analyze the attacks. For security reasons, honeypots should run in virtualized environment to avoid any harm to our host system. We use Debian as a base linux distribution. Our instance runs on heiCLOUD, a cloud service provided by Heidelberg University. It is capable of 16 GB of RAM, 8 VCPUs, and a volatile memory of 30 GB. In addition, we mount a 125 GB permanent memory to securily store our data. In its very early stages, we compared different approaches to achieve this goal. As an example we compared native implementation approaches, using additional frameworks, and ready-to-use solutions. However, the T-Pot , developed by Telekom, offers a profoundly ready-to-use solution with major advantages. It combines several honeypots in conjunction with various analytic tools to trace the newest attacks. In addition, it helps to compare our findings with the ones Kelly et al. [38] claims.

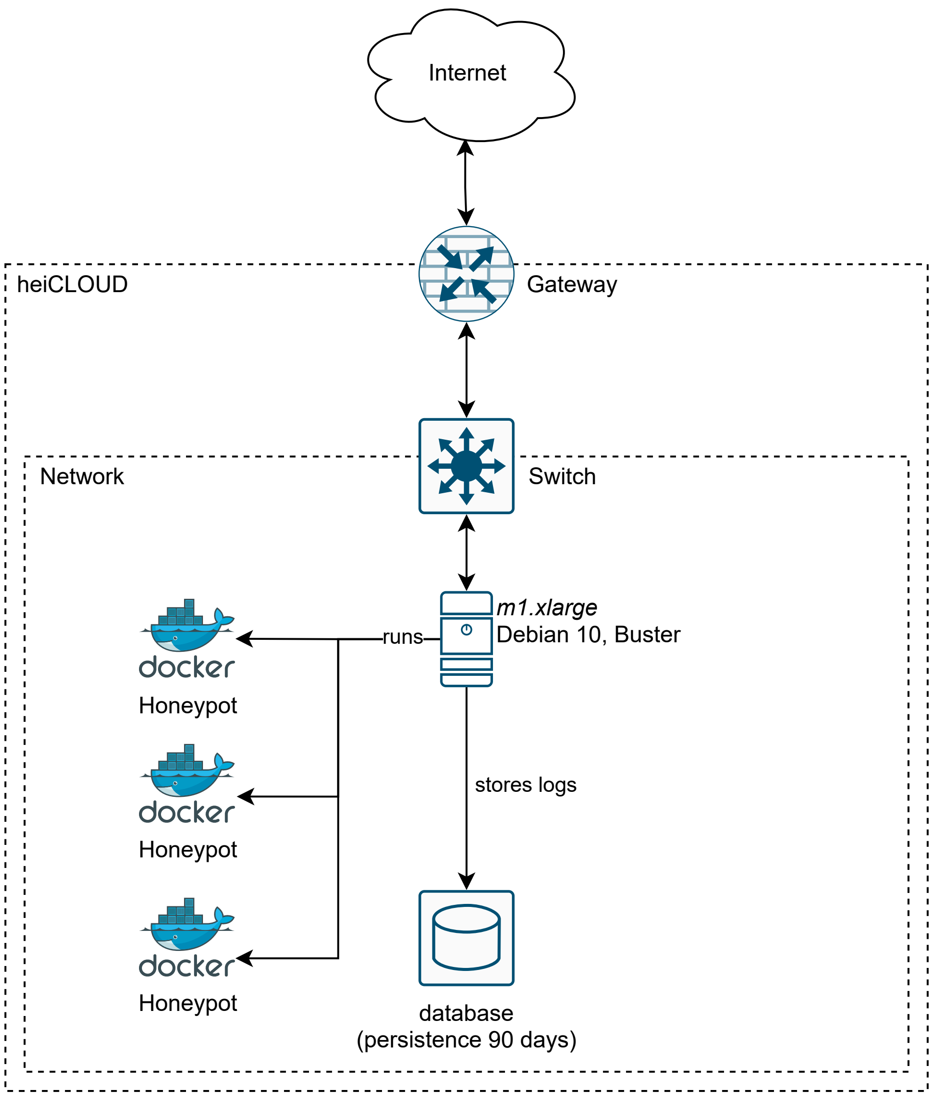


Figure 3.1: Concept to collect honeypot attacks

3.2.1 heiCLOUD

University Computing Center Heidelberg [65] offers a "IaaS specially tailored for higher education and research institutions" called heiCLOUD. It supplies multiple institutes at University Heidelberg with storage, virtual machines, or network components. In addition, heiCLOUD is a DFN[1] member, and offers others to use their

[1]German National Research and Education Network, the communications network for Science and research in Germany

services. As stated on their information website[64], it is (i) capable of freely manageable IT resources, (ii) beholds a stable and fast connection, (iii) ensures high availability and scalability, (iv) has freely selectable VM operating systems, and (v) has a transparent payment model [64]. Based on the open source application OpenStack, users can easily create own network areas, and manage their space individually. Unlike well-known cloud providers, heiCLOUD servers are located withing Germany, thus, abide by the European data privacy law. HeiCLOUD have never considered honeypots for additional cyber security measurements.

3.2.2 T-Pot

To be able to compare our results with Kelly et al. [38], we use the same approach to capture recent cyber attacks. The T-Pot solution, a mixture of Telekom and Honeypot, stands out with their sheer quantity of various honeypots. It requires 8 GB RAM and a minimum of 128 GB hard drive storage. Based on a Debian 10 Buster distribution, it relies on Docker to run their services [25]. T-Pot has to be deployed in a reachable network where intruders are expected. Either TCP and UDP traffic are forwarded without filtering to the network interface, or it runs behind a firewall with forwarding rules. Specified ports for attackers are 1-64000, higher ports are reserved for trusted IPs, thus, a reverse proxy asks for basic authentication. All daemons and tools run on the same network interface whereas some of them are encapsulated in their own Docker network. Docker is a lightweight virtualization technology that uses containers to run on the host system [22]. Unlike virtual machines, Docker reduces overhead with the downside of a greater attack surface. To mitigate attacks, Docker wraps containers in an isolated environment. This is achieved by restricting the kernel namespace and control groups (cgroups). Figure 3.2 visualizes the technical concept of T-Pot. Each service has dedicated ports or port ranges that are exposed. Attackers can communicate either with TCP or UDP. All honeypots and tools create log files that are used to get any knowledge about attackers. In order to view and trace current attacks, T-Pot uses the ELK stack. ELK is the acronym of Elasticsearch, Logstash and Kibana [10]. Elasticsearch is a search engine based on Lucene. It is multitenat-capable and offers full-text search via HTTP. Logstash is used to feed elasticsearch. In general, it offers a open server-side data processing pipeline that helps to send data from multiple sources to an elasticsearch node. Kibana is the main data visualization tool. It offers users to create plots and dashboards, crawl elasticsearch, and trace the system health. All logs of the honeypots and tools are forwarded to the search engine Elasticsearch by Logstash. The ELK stack is not directly exposed to the Internet, thus, an authentication is not needed. Users can monitor all logfiles with Kibana by pre-defined dashboards, or custom search queries. In addition, T-Pot features different services types, namely (i) standard, (ii) sensor, (iii) industrial, (iv) collector, (v) next generation, and (vi) medical. Each service type has a different set of honeypots and tools tailored to their core idea. T-Pot feeds their data to an external Telekom service, however, this data submission can be turned off. For this chapter we restrict ourself to the latest version 20.06.0. Newer versions might be available by time and could differ from ours.

Honeypots

Followingly, all available components will be explained, albeit the sheer quantity of it. In addition, Table 3.2 gives a quick overview of all available honeypots in conjunction with (i) the port they are running on, (ii) thier interation level, and (iii) a short decription.

ADBHoney [21] is a low interaction Android Debug Bridge (ADB) honeypot over TCP/IP. The importance of it lies in the ADB protocol that is used to debug and push content to the device. However, unlike USB it does not support any kind of ample mechanisms of authentication and protection. By exposing the ADB service over any port, an adversary could connect and exploit it. ADBHoney is designed to catch malware that has been pushed onto devices.

Cisco Adaptive Security Appliance (ASA) [52] is a low interaction honeypot that detects CVE-2018-0101[5]. It is a vulnerability that could allow an unauthenticated, remote attacker to cause a reload of the affected system or to remotely execute code. This can be achieved by flooding a webvpn-configured interface with crafted XML packets. Consequently, the attacker obtain full control by executing arbitrary code.

Citrix Application Delivery Controller (ADC) honeypot [33] detects and logs CVE-2019-19781[6] scans and exploitation attempts. This vulnerability allows adversaries to perform directory traversal attacks. Files are accessible by path strings to denote the file or directory. In addition, some file systems include special character to easily traverse the hierarchy. Attackers take advantage of it by combining special characters in order to get access to restricted areas. [28]

Conpot [53] is a low interaction industrial honeypot for ICS, and SCADA. It provides a variety of different common industrial control protocols. An adversary should be tricked by the complex infrastructure, and lure him into attacks. In addition, a custom human machine interface can be conntected to increase the attack surface. By randomly delaying the response time, conpot tries to emulate a real machine handling a certain amount of load.

Cowrie [46] is a medium to high interaction SSH and Telnet honeypot. It offers to log brute-force attacks and shell interactions with attackers. In medium interaction mode cowire emulates a UNIX shell in Python, whereas in high interaction mode it proxies all commands to another system.

DDoSPot [8] is a low interaction honeypot to log and detect UDP-based Distributed Denial of Service (DDoS) attacks. It is used as a platform to support various plugins

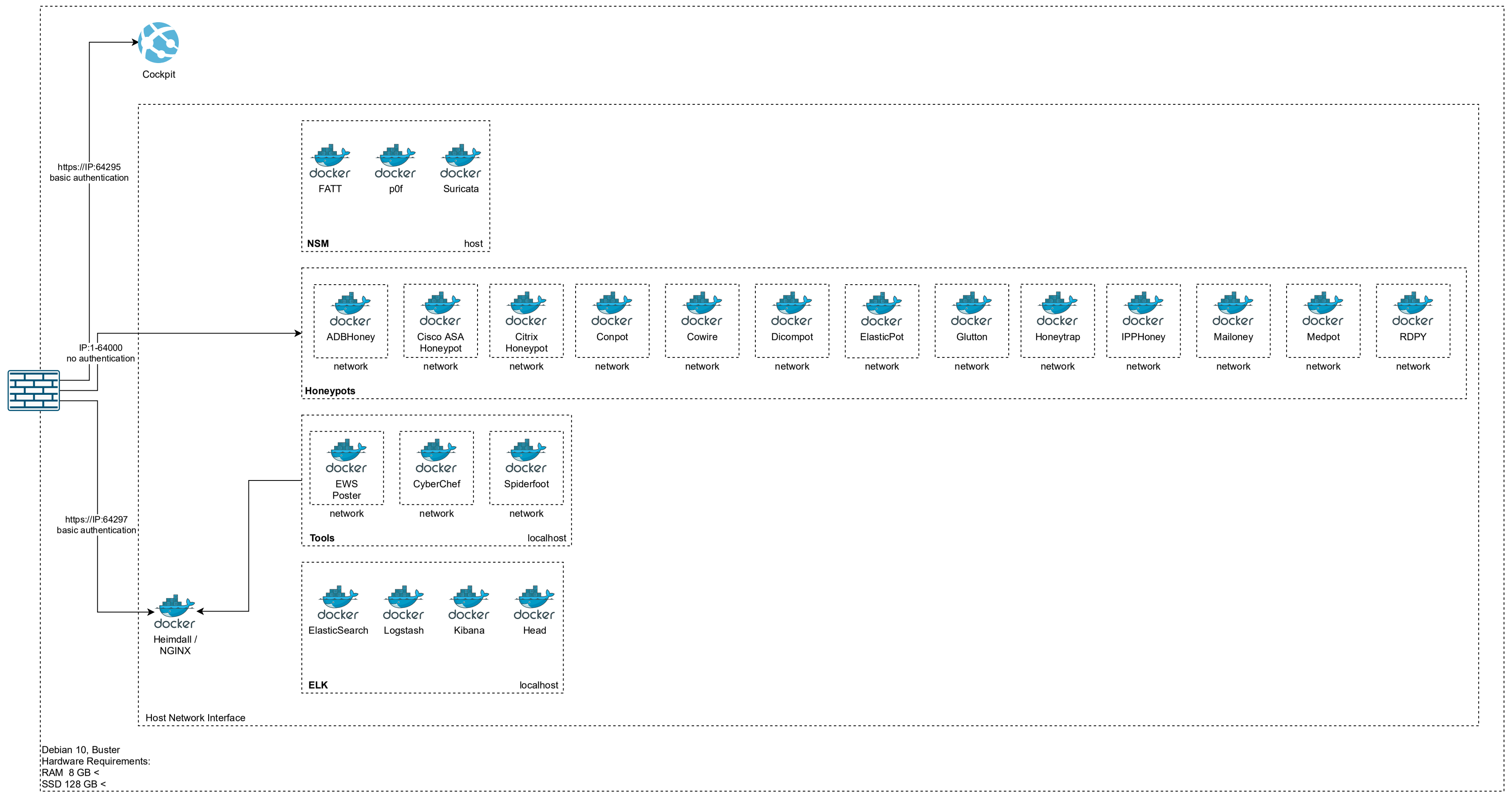


Figure 3.2: T-Pot architecture derived from [?]. Honeypots are encapsulated in their own network. NSM runs on the host network, and thus, receives every packet. ELK and tools run on localhost, and are accessible through NGINX.

for different honeypot services, and servers. Currently, it supports DNS, NTP, SSDP, CHARGEN, and random/mock UDP server.

Dicompot [39] is a low interaction honeypot for the Digital Imaging and Communications in Medicine (DICOM) protocol. As other honeypots before, it mocks a DICOM server in Go to collect logs and detect attacks.

Dionaea [9] is a medium interaction honeypot that tries to capture malware copies by exposing services. It supports a vast variety of protocols such as FTP, SMB, and HTTP. Several modules can be integrated to work with Dionaea such as VirusTotal for further malware results.

Elasticpot [17] is a low interaction honeypot for elasticsearch, a search engine based on the Lucene library.

Glutton [54] is a generic low interaction honeypot that works as a MitM for SSH and TCP. However, lacking documentation does not provide a deeper inside of this honeypot.

Heralding [66] is a credential catching honeypot for protocols like FTP, Telnet, SSH, HTTP, or IMAP.

HoneyPy [30] is a low to medium interaction honeypot that supports several protocols such as UDP, or TCP. New protocols can be added by writing a custom plugin for it. HoneyPy should give the freedom of easily deploying and extending honeypots.

HoneySAP [30] is a low interaction honeypot tailored for SAP services.

HoneyTrap [70] is a low interaction honeypot network security tool. As stated by Werner [70], HoneyTrap is vulnerable to buffer overflow attacks.

IPPHoney [18] is a low interaction Internet Printing Protocol (IPP) honeypot.

Mailoney [11] is a low interaction SMTP honeypot written in Python.

MEDpot [57] is a low interaction honeypot focused on Fast Healthcare Interoperability Resources (FHIR). It is a standard decription data format to transfer and exchange medical health records.

RDPY [49] is a low interaction honeypot of the Microsoft Remote Desktop Protocol (RDP) written in Python. It features client and server side, and it based on the event driven network engine Twisted. It supports authentication over SSL and NLA.

SNARE and TANNER [55, 56] is a honeypot project. SNARE is an abbreviation for Super Next generation Advanced Reactive honEypot. It is a successor of Glastopf, a web application sensor. In addition, it supports the feature of converting existing webpages into attack surfaces. TANNER [56] can be seen as SNARES's brain. Whenever a request has been sent to SNARE, TANNER decides how the response should like.

Tools

T-Pot integrates tools to screen network traffic.

FATT [35] is used to extract metadata and fingerprints such as JA3 [15] and HASSH [36] from captured packets. JA3 is a method for "creating SSL/TLS client fingerprints" whereas HASSH is a "network fingerprinting standard which can be used to identify specific client and server SSH implementations". In addition, it features live network traffic. As noted by the author, FATT is based on a python wrapper for tshark, namely pyshark, and thus having performance downturns. T-Pot applies FATT on every request made on the host network.

Spiderfoot [43] is an open source intelligence automation tool that helps to screen targets to get information about what is exposed over the Internet. It can target different entities such as IP address, domain, hostname or network subnet. In addition, it features more than 200 modules that can be integrated as an extension. T-Pot uses it to scan defensively and thus not include any other module.

Suricata [12] is a "a high performance IDS, intrusion prevention system (IPD) and network security monitoring (NSM) engine". T-Pot lets suricata analyze and assess any request made on the host network.

p0f [71] is a fingerprinting tool that uses passive traffic fingerprinting mechanisms to check TCP/IP communications. T-Pot lets p0f passively check any request made on the host network.

Endlessh [69] is a SSH server that sends an endless, random SSH banner. The key idea is to lock up SSH clients that try to connect to the SSH server. It lowers the transcation speed by intentionally inserting delays. Due to connection establishing before cryptographic exchange, this module does not require any cryptographic libraries.

HellPot [34] is a "endless honeypot". Connecting to this honeypot results in a memory overflow. Its key idea is to send an endless stream of data to the attacker until its memory, or storage runs out.

3.3 Results

Our T-Pot has been deployed for 3 weeks and collected in total 825539 attacks. Overall, Cowire, Heralding, and RDPY received most of the attacks with a total amount of 540398 attacks. Figure 3.3 shows the distribution of honeypot attacks. The total numbers are based on Table 3.3. Noticeable is the large disparity of the previously mentioned attacks on AWS, GCP, and Azure. Why heiCLOUD has

Suricata

Table 3.2: Overview of all available honeypots and tools of T-Pot with interaction level, port, and a short description. Ports are marked with either TCP or UDP, if a port misses any definition, both TCP and UDP are allowed.

Honeypots	Port	Interaction-level	Description
adbhoney [21]	5555/tcp	low	ADB protocol honeypot
ciscoasa [52]	5000/udp, 8443/tcp	low	honeypot for CVE-2018-0101[5] detection
citrixhoneypot [33]	443/tcp	low	detects and logs CVE-2019-19781[6] scans and exploitation attempts
conpot [53]	80, 102, 161, 502, 623, 1025, 2404, 10001, 44818, 47808, 50100	low	industrial honeypot for ICS, and SCADA
cowrie [46]	2222, 23	high	SSH and Telnet honeypot
ddospot [8]	1112/tcp	low	log and detect UDP-based DDoS attacks
dicompot [39]	1112/tcp	medium	honeypot for the DICOM protocol
dionaea [9]	21, 42, 69/udp, 8081, 135, 443, 445, 1433, 1723, 1883, 1900/udp, 3306, 5060/udp, 5061/udp	low	capture malware copies
elasticpot [17]	9200	low	honeypot for elasticsearch
glutton [54]	NFQ	medium	MitM proxy for SSH and TCP
heralding [66]	21, 22, 23, 25, 80, 110, 143, 443, 993, 995, 1080, 5432, 5900	low	credential catching honeypot
honeypy [30]	7, 8, 2048, 2323, 2324, 4096, 9200	low	extendable honeypot
honeysap [30]	3299/tcp	low	honeypot for SAP services
honeytrap [70]	NFQ	medium	captures attacks via unknown protocols
ipphoney [18]	631	low	IPP honeypot
mailoney [11]	25	low	SMTP honeypot
medpot [57]	2575	low	FHIR honeypot
rdpy [49]	3389	low	Microsoft RDP honeypot
snare/tanner [55]	80	low	web application honeypot

Table 3.3: Overview of attacks on heiCLOUD, AWS, GC and Azure. Only the top 10 most attacked honeypots are considered. "-" entails that a honeypot is not part of the top 10.

Honeypots	BASIS		COMPARISION					
	heiCLOUD		AWS		GC		Azure	
	Number	**Pct.**	**Number**	**Pct.**	**Number**	**Pct.**	**Number**	**Pct.**
adbhoney [21]	13586							
ciscoasa [52]	931							
citrixhoneypot [33]	1480							
conpot [53]	807							
cowrie [46]	98813							
ddospot [8]	0							
dicompot [39]	27							
dionaea [9]	3293							
elasticpot [17]	531							
glutton [54]	0							
heralding [66]	35843							
honeypy [30]	0							
honeysap [30]	18							
honeytrap [70]	264264							
ipphoney [18]	0							
mailoney [11]	0							
medpot [57]	3							
rdpy [49]	405742							
snare/tanner [55]	201							
In total	825539							

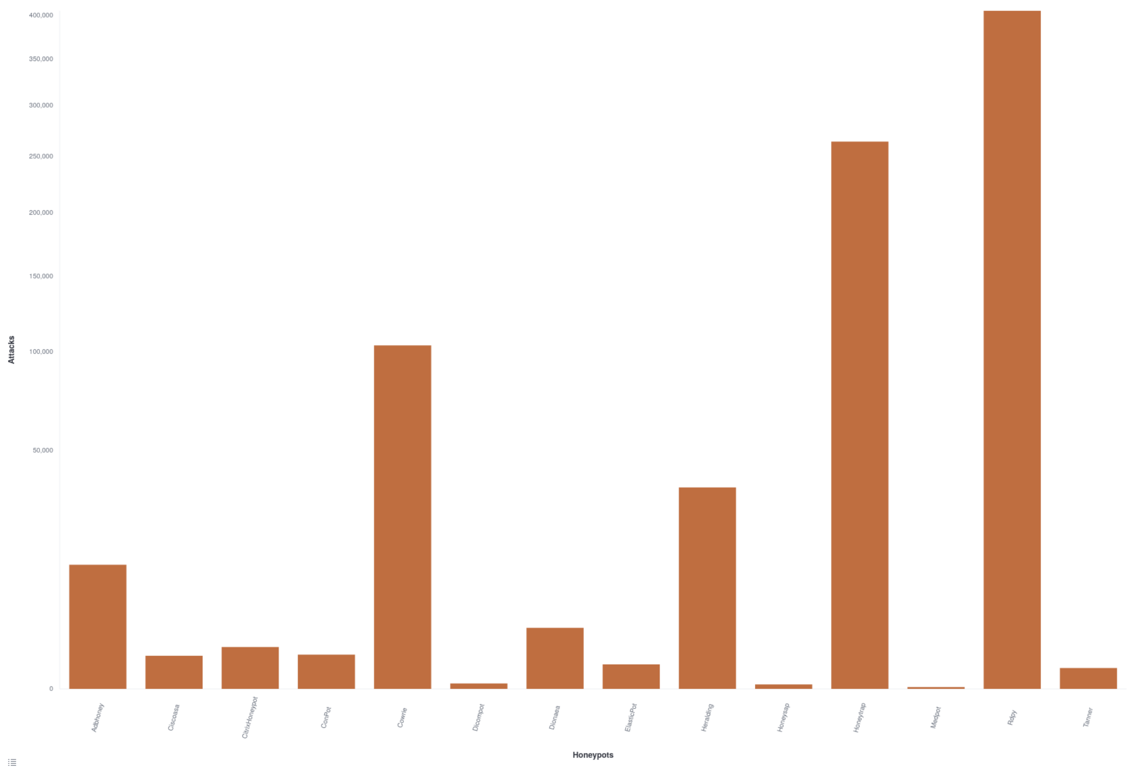


Figure 3.3: Distribution of honeypot attacks

Honeytrap

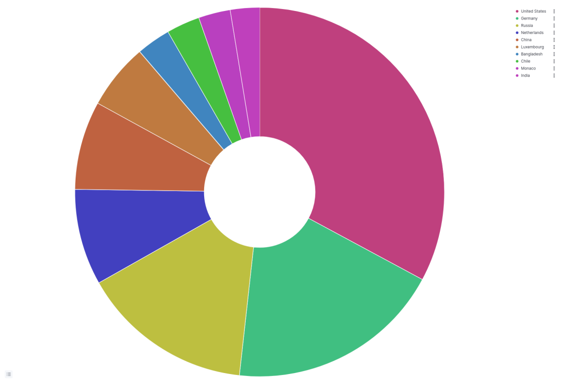
RDPY

Cowire

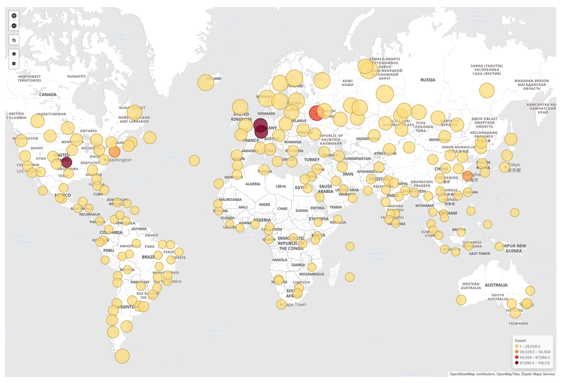
Interestingly, zero-day exploits like the Apache vulnerability [7] that came with version 2.49.0 got registered in CVE on the 6th of October, and immediately used by attackers on the 15th of October. Attackers could perform a remote code execution using path traversal attack when the CGI scripts of Apache are enabled. This shows how fast attackers, most likely bots, adapt to new vulnerabilities in order to compromise more systems.

3.4 Discussion

As recently investigated by Vetterl [67], fingerprinting honeypots is becoming easier due to a fatal flaw in the underlying protocol implementation. Vetterl [67] states that attackers always try to prevent their methods, exploits and tools to be disclosed. Therefore detecting honeypots before attack them to avoid disclosing potential skills is a strong motivation for black hats.

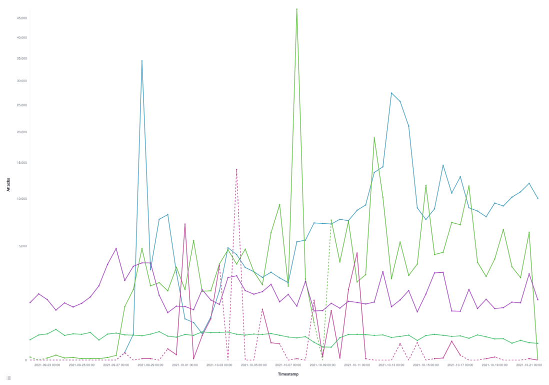


(a)

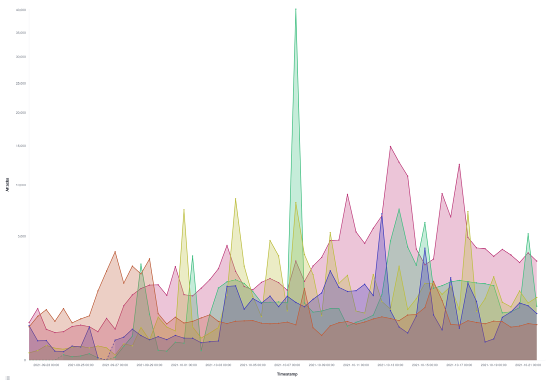


(b)

Figure 3.4: Attack distribution

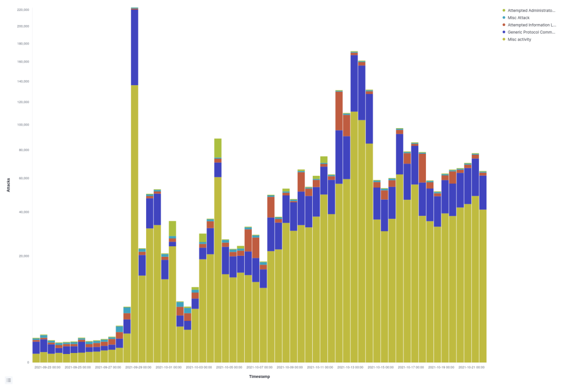


(a)

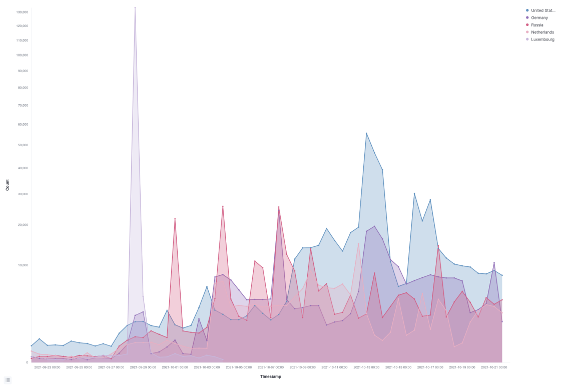


(b)

Figure 3.5: Attack histogram

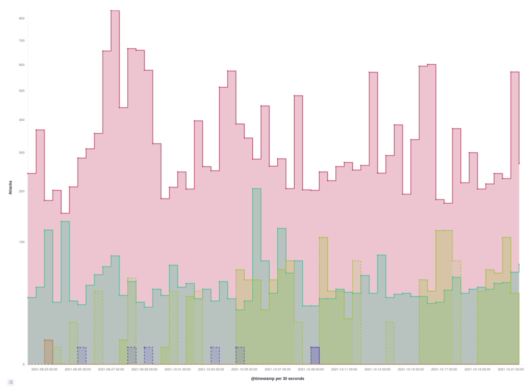


(a)

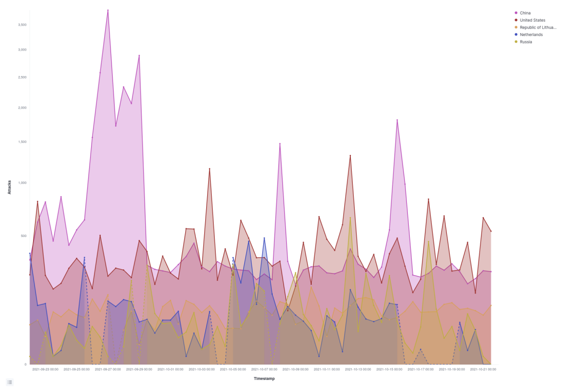


(b)

Figure 3.6: Suricata results

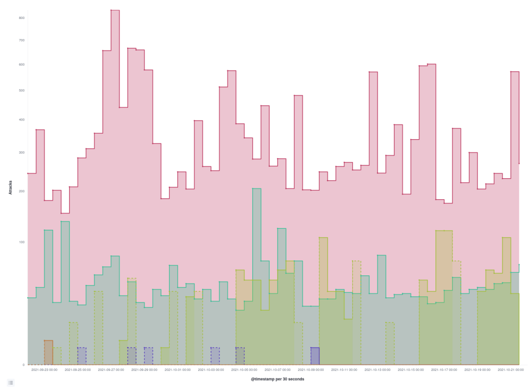


(a)

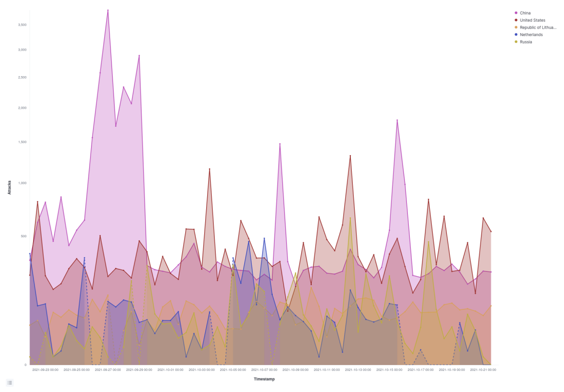


(b)

Figure 3.7: Honeytrap results

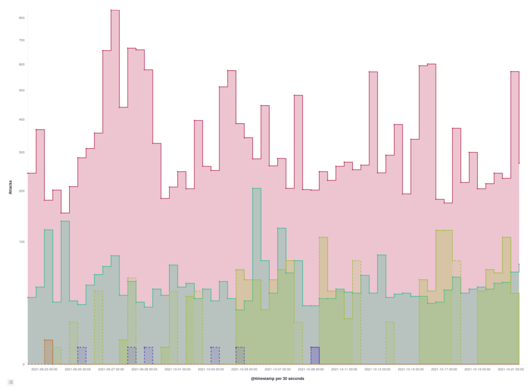


(a)

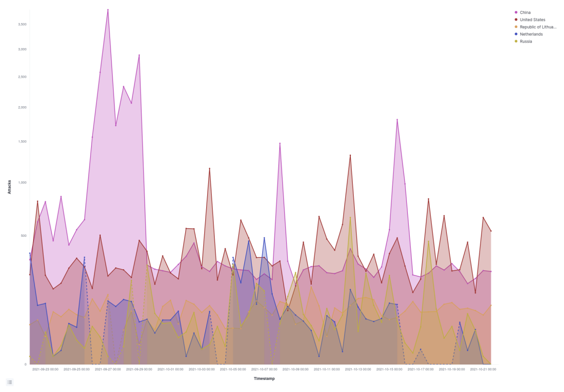


(b)

Figure 3.8: RDPY results



(a)



(b)

Figure 3.9: Cowire results

3.5 Summary

In this chapter we have

Chapter 4

Concept

4.1 Introduction

4.2 Summary

Chapter 5

Experimental Results & Evaluation

5.1 Summary

Chapter 6

Conclusion

6.1 Future work

Bibliography

[1] CVE-2001-0540. Available from MITRE, CVE-ID CVE-2001-0540., March 09 2002. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0540.

[2] CVE-2005-4050. Available from MITRE, CVE-ID CVE-2005-4050., December 07 2005. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-4050.

[3] CVE-2006-2369. Available from MITRE, CVE-ID CVE-2006-2369., May 15 2006. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-2369.

[4] CVE-2012-0152. Available from MITRE, CVE-ID CVE-2012-0152., December 13 2011. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0152.

[5] CVE-2018-0101. Available from MITRE, CVE-ID CVE-2018-0101., November 27 2017. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-0101.

[6] CVE-2019-19781. Available from MITRE, CVE-ID CVE-2019-19781., December 13 2019. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-19781.

[7] CVE-2021-42013. Available from MITRE, CVE-ID CVE-2021-42013., October 06 2021. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-42013.

[8] DDoSPot. https://github.com/aelth/ddospot, 2021. Accessed: 2021-09-26.

[9] dionaea - catches bugs. https://github.com/DinoTools/dionaea, 2021. Accessed: 2021-09-26.

[10] The Elastic Stack. https://www.elastic.co/elastic-stack/, 2021. Accessed: 2021-09-26.

[11] Mailoney - an SMTP honeypot. https://github.com/phin3has/mailoney, 2021. Accessed: 2021-09-26.

[12] Suricata. https://github.com/OISF/suricata, 2021. Accessed: 2021-09-26.

[13] Fahim Abbasi. 2020 trustwave global security report. *Trustwave*, 2020.

[14] Jack Whitsitt Alberto Gonzalez. The bait and switch honeypot: An active and aggressive part of your network security infrastructure. http://baitnswitch.sourceforge.net/, 2002. Accessed: 2021-09-06.

[15] John B. Althouse, Jeff Atkinson, and Josh Atkins. JA3 - a method for profiling ssl/tls clients. https://github.com/salesforce/ja3, 2021. Accessed: 2021-09-26.

[16] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, April 2010. doi: 10.1145/1721654.1721672. URL https://doi.org/10.1145/1721654.1721672.

[17] Vesselin Bontchev. Elasticpot: an elasticsearch honeypot. https://gitlab.com/bontchev/elasticpot, 2021. Accessed: 2021-09-26.

[18] Vesselin Bontchev. Ipphoney: an internet printing protocol honeypot. https://gitlab.com/bontchev/ipphoney, 2021. Accessed: 2021-09-26.

[19] Brian Caswell, James C. Foster, Ryan Russell, Jay Beale, and Jeffrey Posluns. *Snort 2.0 Intrusion Detection*. Syngress Publishing, 2003. ISBN 1931836744.

[20] Bill Cheswick. An evening with berferd in which a cracker is lured, endured, and studied. In *In Proc. Winter USENIX Conference*, pages 163–174, 1992.

[21] Gabriel Cirlig. ADBHoney. https://github.com/huuck/ADBHoney, 2021. Accessed: 2021-09-26.

[22] Theo Combe, Antony Martin, and Roberto Di Pietro. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 3(5):54–62, 2016. doi: 10.1109/MCC.2016.100.

[23] Jeff Daniels. Server virtualization architecture and implementation. *XRDS*, 16(1):8–12, September 2009. ISSN 1528-4972. doi: 10.1145/1618588.1618592. URL https://doi.org/10.1145/1618588.1618592.

[24] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: Issues and challenges. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010. doi: 10.1109/aina.2010.187. URL https://doi.org/10.1109/aina.2010.187.

[25] Docker. Docker overview. https://docs.docker.com/get-started/overview/, 2021. Accessed: 2021-09-21.

[26] Europol. Internet organised crime threat assessment (iocta). *European Union Agency for Law Enforcement Cooperation*, 9(1), 2020.

[27] Europol. About europol. https://www.europol.europa.eu/about-europol, 2021. Accessed: 2021-09-04.

[28] Michael Flanders. A simple and intuitive algorithm for preventing directory traversal attacks, 2019.

[29] Federal Office for Information Security. Cert-bund. `https://www.bsi.bund.de/EN/Topics/IT-Crisis-Management/CERT-Bund/cert-bund_node.html`, 2021. Accessed: 2021-09-12.

[30] Martin Gallo. Honeysap: Sap low-interaction honeypot. `https://github.com/SecureAuthCorp/HoneySAP`, 2021. Accessed: 2021-09-26.

[31] Lichstein. HA. When should you emulate. *Datamation*, 15(11):205, 1969.

[32] Brian Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, July 2008. ISSN 0001-0782. doi: 10.1145/1364782.1364786. URL `https://doi.org/10.1145/1364782.1364786`.

[33] Marcus Hutchins. Honepot for cve-2019-19781 (citrix adc). `https://github.com/MalwareTech/CitrixHoneypot`, 2020. Accessed: 2021-09-26.

[34] Yung Innanet. Hellpot. `https://github.com/yunginnanet/HellPot`, 2021. Accessed: 2021-09-26.

[35] Adel Karimi. FATT /fingerprintAllTheThings - a pyshark based script for extracting network metadata and fingerprints from pcap files and live network traffic. `https://github.com/0x4D31/fatt`, 2021. Accessed: 2021-09-26.

[36] Adel Karimi, Ben Reardson, John Althouse, Jeff Atkinson, and Josh Atkins. HASSH - a profiling method for ssh clients and servers. `https://github.com/salesforce/hassh`, 2021. Accessed: 2021-09-26.

[37] Tejvir Kaur, Vimmi Malhotra, and Dheerendra Singh. Comparison of network security tools- firewall, intrusion detection system and honeypot. 2014.

[38] Christopher Kelly, Nikolaos Pitropakis, Alexios Mylonas, Sean McKeown, and William J. Buchanan. A comparative analysis of honeypots on different cloud platforms. *Sensors*, 21(7):2433, April 2021. doi: 10.3390/s21072433. URL `https://doi.org/10.3390/s21072433`.

[39] Mikael Keri. Dicompot - A Digital Imaging and Communications in Medicine (DICOM) Honeypot. `https://github.com/nsmfoo/dicompot`, 2021. Accessed: 2021-09-26.

[40] Christian Kreibich and Jon Crowcroft. Honeycomb: Creating intrusion detection signatures using honeypots. *SIGCOMM Comput. Commun. Rev.*, 34 (1):51–56, January 2004. ISSN 0146-4833. doi: 10.1145/972374.972384. URL `https://doi.org/10.1145/972374.972384`.

[41] D Kreuter. Where server virtualization was born. *Virtual Strategy Magazine*, 2004.

[42] P M Mell and T Grance. The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology, 2011. URL https://doi.org/10.6028/nist.sp.800-145.

[43] Steve Micallef. Spiderfoot automates osint for threat intelligence and mapping your attack surface. https://github.com/smicallef/spiderfoot, 2021. Accessed: 2021-09-26.

[44] Iyatiti Mokube and Michele Adams. Honeypots: Concepts, approaches, and challenges. In *Proceedings of the 45th Annual Southeast Regional Conference*, ACM-SE 45, page 321–326, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936295. doi: 10.1145/1233341.1233399. URL https://doi.org/10.1145/1233341.1233399.

[45] Marcin Nawrocki, Matthias Wählisch, Thomas C. Schmidt, Christian Keil, and Jochen Schönfelder. A survey on honeypot software and data analysis. *CoRR*, abs/1608.06249, 2016. URL http://arxiv.org/abs/1608.06249.

[46] Michel Oosterhof. Cowrie SSH/Telnet Honeypot. https://github.com/cowrie/cowrie, 2021. Accessed: 2021-09-26.

[47] G. Schaefer P. Diebold, A. Hess. A honeypot architecture for detecting and analyzing unknown network attacks. Telecommunication Networks Group, Technische Universität Berlin, Germany,, Feburary 2005.

[48] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Comput. Netw.*, 31(23–24):2435–2463, December 1999. ISSN 1389-1286. doi: 10.1016/S1389-1286(99)00112-7. URL https://doi.org/10.1016/S1389-1286(99)00112-7.

[49] Sylvain Peyrefitte. Rdpy: Remote desktop protocol in twisted python. https://github.com/citronneur/rdpy, 2021. Accessed: 2021-09-26.

[50] Niels Provos. Honeyd - a virtual honeypot daemon. Washington, D.C., August 2003. USENIX Association.

[51] Antonio Regalado. Who coined 'cloud computing'?, Feb 2020. URL https://www.technologyreview.com/2011/10/31/257406/who-coined-cloud-computing/.

[52] Cymmetria Research. Cisco ASA honeypot. https://github.com/Cymmetria/ciscoasa_honeypot, 2018. Accessed: 2021-09-26.

[53] Lukas Rist, Johnny Vestergaard, Daniel Haslinger, Andrea Pasquale, and John Smith. Conpot ics scada honeypot. http://conpot.org/, 2021. Accessed: 2021-09-26.

[54] Lukas Rist, Johnny Vestergaard, Daniel Haslinger, Andrea Pasquale, and John Smith. Glutton: low interaction honeypot. https://github.com/mushorg/glutton, 2021. Accessed: 2021-09-26.

[55] Lukas Rist, Johnny Vestergaard, Daniel Haslinger, Andrea Pasquale, and John Smith. Snare: Super next generation advanced reactive honeypot. `https://github.com/mushorg/snare`, 2021. Accessed: 2021-09-26.

[56] Lukas Rist, Johnny Vestergaard, Daniel Haslinger, Andrea Pasquale, and John Smith. Tanner: He who flays the hide. `https://github.com/mushorg/tanner`, 2021. Accessed: 2021-09-26.

[57] Markus Schmall. Medpot: HL7 / FHIR honeypot. `https://github.com/schmalle/medpot`, 2021. Accessed: 2021-09-26.

[58] Bruce Schneier. *Secrets & lies - IT-Sicherheit in einer vernetzten Welt.* Dpunkt-Verlag, Köln, 2004. ISBN 978-3-898-64302-3.

[59] Pavol Sokol, Jakub Míšek, and Martin Husák. Honeypots and honeynets: issues of privacy. *EURASIP Journal on Information Security*, 2017, 02 2017. doi: 10.1186/s13635-017-0057-4.

[60] Lance Spitzner. *Honeypots - Tracking Hackers.* Addison-Wesley, Amsterdam, 2003. ISBN 978-0-321-10895-1.

[61] Statista. Year-over-year change in average monthly in-home data usage by device in the united states from january to march 2020. `https://www.statista.com/statistics/1106821/covid-19-change-in-in-home-data-usage-in-us-2020/`, 2021. Accessed: 2021-09-04.

[62] Clifford Stoll. *The Cuckoo's Egg: Tracking a Spy through the Maze of Computer Espionage.* Pocket Books, 2000. ISBN 0743411463.

[63] K. Takemori, K. Rikitake, Y. Miyake, and K. Nakao. Intrusion trap system: an efficient platform for gathering intrusion-related information. In *10th International Conference on Telecommunications, 2003. ICT 2003.*, volume 1, pages 614–619 vol.1, 2003. doi: 10.1109/ICTEL.2003.1191480.

[64] University Computing Center Heidelberg. heicloud - the heidelberg university cloud infrastructure. `https://heicloud.uni-heidelberg.de/heiCLOUD`, 2021. Accessed: 2021-09-02.

[65] University Computing Center Heidelberg. Heicloud. `https://www.urz.uni-heidelberg.de/en/service-catalogue/cloud/heicloud`, 2021. Accessed: 2021-09-02.

[66] Johnny Vestergaard. Heralding: Credentials catching honeypot. `https://github.com/johnnykv/heralding`, 2021. Accessed: 2021-09-26.

[67] Alexander Vetterl. Honeypots in the age of universal attacks and the Internet of Things. Technical Report UCAM-CL-TR-944, University of Cambridge, Computer Laboratory, February 2020. URL `https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-944.pdf`.

[68] Lizhe Wang, Gregor von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing: a perspective study. *New Generation Computing*, 28(2):137–146, April 2010. doi: 10.1007/s00354-008-0081-5. URL `https://doi.org/10.1007/s00354-008-0081-5`.

[69] Christopher Wellons. Endlessh: an ssh tarpit. `https://github.com/skeeto/endlessh`, 2021. Accessed: 2021-09-26.

[70] Tillmann Werner. Honeytrap. `https://github.com/armedpot/honeytrap/`, 2021. Accessed: 2021-09-26.

[71] Michal Zalewski. p0f v3: passive fingerprinter. `https://github.com/p0f/p0f`, 2021. Accessed: 2021-09-26.

Appendices

Appendix A

T-Pot

...pictures of view

Appendix B

Installation and Configuration