Faculty of Mathematics and Computer Science

Heidelberg University

Master thesis

in Computer Science

submitted by

Stefan Machmeier

born in Heidelberg

1996

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):

Contents

Acronyms

CERT Computer Emergency Response Team

DaaS Data-as-a-Service

DTK Deception Toolkit

HaaS Hardware-as-a-Service

HTTP Hypertext Transfer Protocol

IaaS Infrastructure-as-a-Service

IDS Intrusion Detection Systems

IOCTA Internet Organised Crime Threat Assessment

NIST National Institute of Standards and Technology

OS Operating System

PaaS Platform-as-a-Service

SaaS Software-as-a-Service

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 [19]. Even Europol (an agency that fights against terrorism, cybercrime, and other threats [6]) 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. [5]

Even unrelated to the pandamic, 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 [18]. 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.

1.2 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?

1.3 Limitations

Due to the fact that Heidelberg offers a cloud service, called “HeiCloud”, 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

2.1.1 Docker

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.[16] Starting from 2007 cloud computing evolved into a serious competitor and outnumbered the keywords "virtualization", and "grid computing" reported by Google trends [21]. 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.

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

2.2.1 Definition of Cloud Computing

Considering the definition of Brian Hayes, cloud computing is "a shift in the geography of computation" [7]. Thus, computational workload is moved away from local instances towards services and datacenters that provide the need of users [1].

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" [10]. 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) [10]:

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 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.

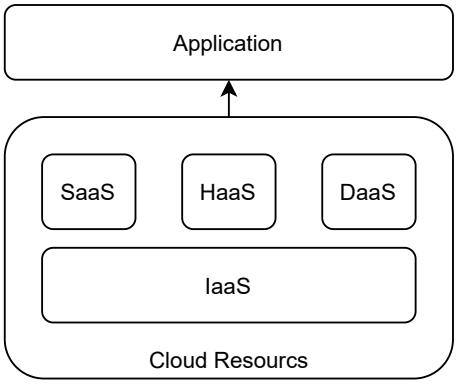

Figure 2.1: Abstract visualization of service models. The lowest level within the container "cloud resources" represents the depth of functionalities

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.[4] "Limited user-specific application configuration settings" can be made [10].

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. [10]

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. [10]

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.[4] Mostly they require "dedicated server, software license, post-delivery services, and in-house IT maintenance" [4]. 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.[4]

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.[21]

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 [10]. 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		Amazon EC2
	Nextcloud		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. [4, 10]

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. [10]

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. [10]

Hybrid cloud is a mixure of two or more cloud infrastructures, such as private and public cloud.However, each entity keeps its core element. However, hybrid clouds defines "standardized or propriertary technology to enables data and application portability"[10].

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"[20], and Bill Cheswick's whitepaper "An Evening With Berferd"[2] describe concepts that are consider nowadays as honeypots.[18] 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", 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.

citation

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 [12]. 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 et al. [18] 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 et al. [18] 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 [18]. In addition, Spitzner et al. 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. An example

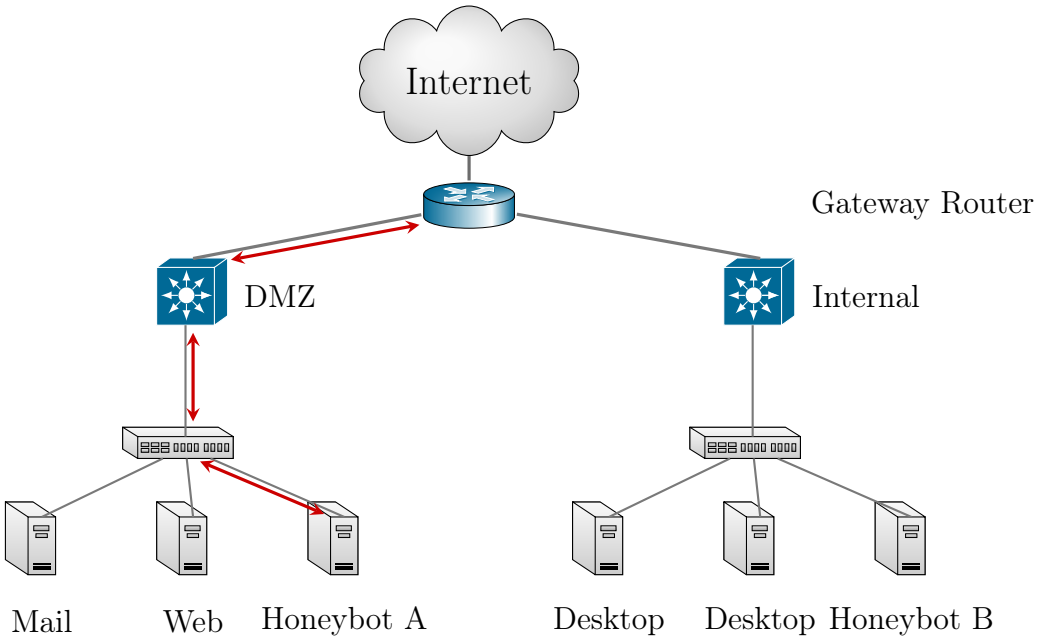

Figure 2.2: Example of honeypots in a simplified network (derived from [18])

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.[18]

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 environ-

ments due to a higher increase of risk. Facing an increase in deployment complexity, and maintenance does not attract a production usage.[18]

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 et al. adds an additional differentiation for the virtual honeypot framework [15] and splits it into the following types:

- Physical honeypots are "real machines on the network with its own IP address" [15]
- Virtual honeypots are "simulated by another machine that responds to network traffix sent to the virtual honeypot" [15]

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 Operating System (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. [18]

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. [18]

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 Schneider et al. [17] 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. [12]

Honeypots contribute to the security concept like firewalls, or Intrusion Detection Systems (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. [12] Nawrocki, Wählisch, Schmidt, Keil, and Schönfelder et al. [12] distinct honeypots from other objectives such as firewall, or log-monitoring.

2.3.4 Value of Honeypots

To assess the value of honeypots we want to take a closer look to thier advantages and disadvantes.[11, 8, 18]

Table 2.3: Distinction between security concepts based on areas of operations (derived from [12])

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

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, Wählisch, Schmidt, Keil, and Schönfelder et al. [12] 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

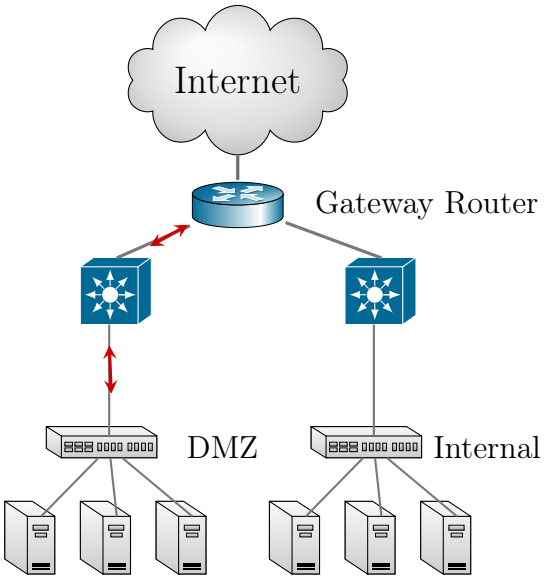

Figure 2.3: Example of honeynets in a simplified network (derived from [18])

[18]

2.3.6 Legal Issues

[18]

2.4 Related Work

In this chapter we investigate previous work that have been done

2.4.1 Honeypot technologies

The Bait'n'Switch Honeypot

[14]

Intrusion Trap System

[14]

Honeycomb

[14]

Honeypots in a cloud environment

[9]

T-Pot

Chapter 3

Cloud Security with Honeypots

[13]
[9]

3.1 Summary

Chapter 4

Concept

4.1 Introduction

4.2 HeiCloud

IaaS [3]

Get information or paper about that

4.3 Methods Used

4.3.1 Requirements and specification

4.3.2 Proposed Honeypots

Cowire

Dionaea

Honeyd

4.3.3 Frameworks and technologies

HoneyTrap

4.4 Architecture

Maybe rename section

4.5 Data Management

4.6 Data Analysis

4.7 Summary

Chapter 5

Data Management

Chapter 6

Experimental Results & Evaluation

6.1 Attack vectors

6.1.1 Primer

6.2 Concept

6.3 Implementation

6.4 Summary

Chapter 7

Conclusion

7.1 Future work

Bibliography

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, Apr. 2010.

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

[3] R. der Universität Heidelberg. heicloud - die heidelberger cloud-infrastruktur. `https://heicloud.uni-heidelberg.de/heiCLOUD`, 2021. Accessed: 2021-09-02.

[4] T. Dillon, C. Wu, and E. Chang. Cloud computing: Issues and challenges. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010.

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

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

[7] B. Hayes. Cloud computing, 2008.

[8] T. Kaur, V. Malhotra, and D. Singh. Comparison of network security tools-firewall, intrusion detection system and honeypot. 2014.

[9] C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, and W. J. Buchanan. A comparative analysis of honeypots on different cloud platforms. *Sensors*, 21(7):2433, Apr. 2021.

[10] P. M. Mell and T. Grance. The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology, 2011.

[11] I. Mokube and M. 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.

[12] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder. A survey on honeypot software and data analysis. *CoRR*, abs/1608.06249, 2016.

[13] S. Nithin Chandra and T. Madhuri. Cloud security using honeypot systems. *International Journal of Scientific & Engineering Research*, 3(3):1, 2012.

[14] G. S. P. Diebold, A. Hess. A honeypot architecture for detecting and analyzing unknown network attacks, Feburary 2005.

[15] N. Provos. Honeyd: A virtual honeypot daemon (extended abstract). 01 2003.

[16] A. Regalado. Who coined 'cloud computing'?, Feb 2020.

[17] B. Schneier. *Secrets & lies - IT-Sicherheit in einer vernetzten Welt*. Dpunkt-Verlag, Köln, 2004.

[18] L. Spitzner. *Honeypots - Tracking Hackers*. Addison-Wesley, Amsterdam, 2003.

[19] 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.

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

[21] L. Wang, G. von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu. Cloud computing: a perspective study. *New Generation Computing*, 28(2):137–146, Apr. 2010.

Appendices

Appendix A

Installation and Configuration