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

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

1 Introduction

1.1 Problem description

1.2 Justification, motivation and benefits

1.3 Research questions

1.4 Limitations

2 Background

This chapter concludes the fundamental knowledge that is needed to comprehend the upcoming practical work. Firstly, an introduction to cloud computing will be held. Next, a throrough understanding of honeypots is given. Lastly, we introduce some concepts of intrusion detection systens.

2.1 Virtualization

2.1.1 Docker

2.2 Cloud Computing

Check cites

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.[Reg20] Starting from 2007 cloud computing evolved into a serious competitor and outnumbered the keywords "virtualization", and "grid computing" reported by Google trends [WvLY+10]. 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.

In this section, we want to give basic unterstandings of cloud computing. First, we exhibit the history and draw some motivation. Following, we look at some the definitions, and point out characteristics. Lastly, we give an short introduction to HeiCloud, a cloud service that is offered by Heidelberg University.

[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" [Hay08]. Thus, computational workload is moved away from local instances towards services and datacenters that provide the need of users [AFG+10].

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" [MG11]. 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) [MG11]:

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,

Fix image

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.

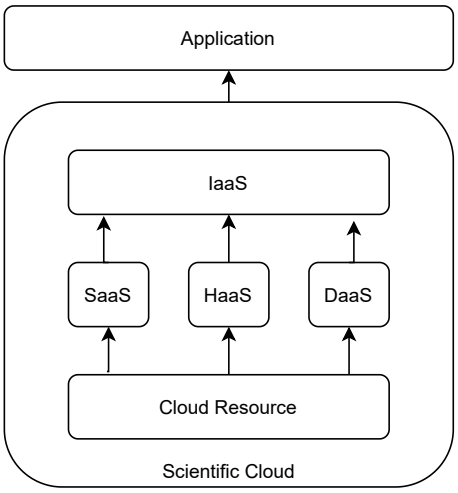

Figure 2.1: Cloud functionalities (derived from [WvLY+10])

Software-as-a-Service (SaaS) is a high-level abstraction to consumers. Controlling the underlying infrastructure is not supported. Often provider uses a multitenancy 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.[DWC10] "Limited user-specific application configuration settings" can be made [MG11].

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.

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.

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.[DWC10] Mostly they require "dedicated server, software license,

post-delivery services, and in-house IT maintenance" [DWC10]. 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.[DWC10]

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.[WvLY+10]

Table 2.1: Examples for 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 [MG11]. 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. [DWC10] [MG11]

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.

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.

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"[MG11].

2.2.4 HeiCloud

IaaS

Get information or paper about that

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"[Sto00], and Bill Cheswick's whitepaper "An Evening With Berferd"[Che92] describe concepts that are consider nowadays as honeypots.[Spi03] 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 [NWS+16]. 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. [Spi03] 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. [Spi03] 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 [Spi03]. 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.

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

- Production honeypots
- Research honeypots

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.[Spi03]

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

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 [Pro03] and splits it into the following types:

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

2.3.2 Level of Interaction

When building and deploying a honeypot, the goal has to be defines. Should it gather unauthorized activities, such as an NMAP scan? Do you want to learn about buzzing tools and tactics? Honeypots differ in level of interaction.

Low-interaction honeypots are the easiest level of interaction. Only a small set of services like SSH, Telnet, or FTP are supported. In terms of risk, low-interaction honeypots do not give access to the underlying Operating System (OS) which make them .

Medium-interaction honeypots

High-interaction honeypots

Pure honeypots

2.3.3 Security concepts

Check honeypots in terms of security concepts

[Sch04]

1. Prevention
2. Detection
3. Reaction

[NWS+16]

2.3.4 Value of Honeypots

Advantages

- Data Value
- Resources
- Simplicity
- Return on Investment

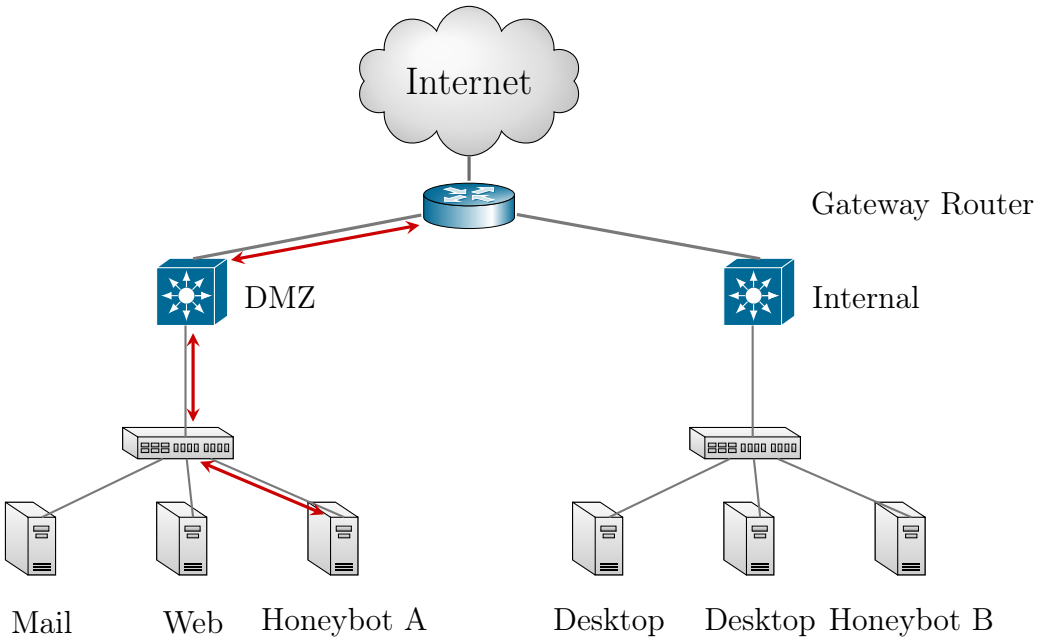

Figure 2.2: Example of honeypots in a simplified network (derived from [Spi03])

- Independent from Workload
- Zero-Day-Exploit Detection
- Flexibility
- Reduced False Positives and Negatives

Disadvantages

- Narrow Field of View
- Being Fingerprinted
- Risk to the Environment

2.3.5 Honeynets

[Spi03]

2.3.6 Legal Issues

[Spi03]

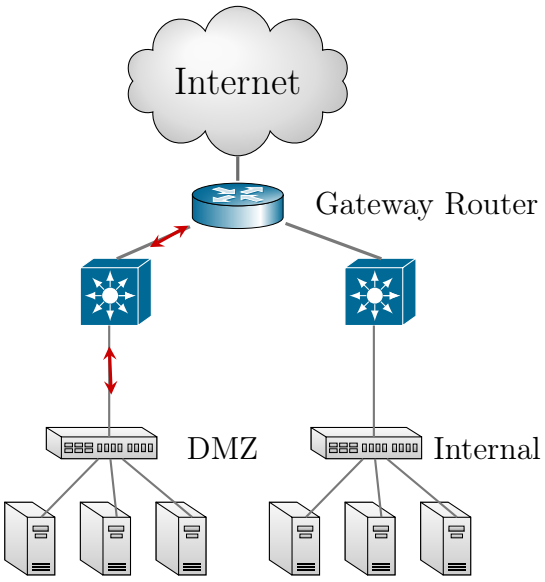

Figure 2.3: Example of honeynets in a simplified network (derived from [Spi03])

2.4 Summary

3 Previous Work

3.1 The Bait'n'Switch Honeypot

[PD05]

3.2 Intrusion Trap System

[PD05]

3.3 Honeycomb

[PD05]

3.4 Honeypots in a cloud environment

[KPM+21]

3.5 T-Pot

3.6 Summary

4 Cloud Security with Honeypots

[NCM12]
[KPM+21]

4.1 Summary

5 Concept

5.1 Introduction

5.2 Summary

6 Methods Used

6.1 Requirements and specification

6.2 Proposed Honeypots

6.2.1 Cowire

6.2.2 Dionaea

6.2.3 Honeyd

6.3 Frameworks and technologies

6.3.1 HoneyTrap

7 Data Management

8 Data Analysis

9 Practical Work

9.1 Attack vectors

9.1.1 Primer

9.2 Concept

9.3 Implementation

9.4 Summary

10 Evaluation

10.1 Data Analyzation

10.2 Summary

11 Conclusion

11.1 Future work

Bibliography

[AFG+10] 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.

[Che92] 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.

[DWC10] 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.

[Hay08] Brian Hayes. Cloud computing, 2008.

[KPM+21] 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.

[MG11] P M Mell and T Grance. The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology, 2011.

[NCM12] SR Nithin Chandra and TM Madhuri. Cloud security using honeypot systems. *International Journal of Scientific & Engineering Research*, 3(3):1, 2012.

[NWS+16] 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.

[PD05] G. Schaefer P. Diebold, A. Hess. A honeypot architecture for detecting and analyzing unknown network attacks, Feburary 2005.

[Pro03] Niels Provos. Honeyd: A virtual honeypot daemon (extended abstract). 01 2003.

[Reg20] Antonio Regalado. Who coined 'cloud computing'?, Feb 2020.

[Sch04] Bruce Schneier. *Secrets & lies - IT-Sicherheit in einer vernetzten Welt*. Dpunkt-Verlag, Köln, 2004.

[Spi03] Lance Spitzner. *Honeypots - Tracking Hackers*. Addison-Wesley, Amsterdam, 2003.

[Sto00] Clifford Stoll. *The Cuckoo's Egg: Tracking a Spy through the Maze of Computer Espionage*. Pocket Books, 2000.

[WvLY+10] 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.

Appendices

A Installation and Configuration