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

DaaS Data-as-a-Service

HaaS Hardware-as-a-Service

HTTP Hypertext Transfer Protocol

IaaS Infrastructure-as-a-Service

NIST National Institute of Standards and Technology

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 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.[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. To do so, . Lastly, we give an short introduction to HeiCloud, an offered by Heidelberg University.

Fix introduction

2.1.1 History

maybe leave out?

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

2.1.2 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 2.1.3), and four deployment models (see subsection 2.1.4) [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.

2.1.3 Service models

Service models Examples for such service models can be derived from Figure 2.1.3. Figure 2.1

more introduction

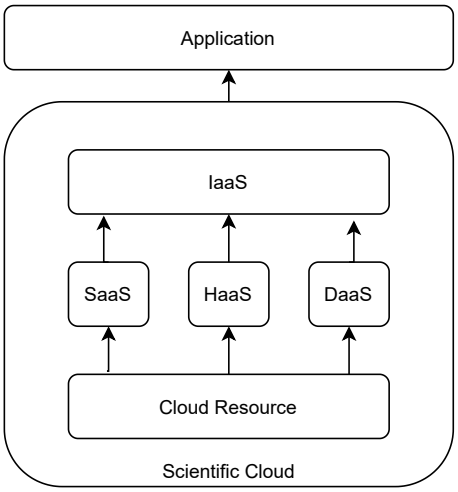

Figure 2.1: Cloud functionalities (derived from [WvLY+10])

Software-as-a-Service (SaaS) is a high-level abstraction to consumers. Controlling the underlying infrastructure is not supported. Often provider uses a 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.[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.

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 data, file systems and RDBMS often lack in performance. DaaS outruns such weak links by employing a table-style abstraction that can be scaled.

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 manage-

able. [WvLY+10]

Table 2.1: Examples for cloud service models

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

2.1.4 Deployment models

Private Cloud

Community Cloud

Public Cloud

Hybrid Cloud

Examples for such deployment models are:

Table 2.2: Examples for cloud deployment models

	Name
Private Cloud	
Community Cloud	
Public Cloud	
Hybrid Cloud	

2.1.5 Cloud Security

[NCM12]

2.1.6 HeiCloud

IaaS

Get information or paper about that

2.2 Honeypots

The first public honeypot [Spi03]

Add some history of honeypots

2.2.1 Definition of a Honeypot

On the Internet there are a dozen of defintions for honeypots. Thus, to cope with all the subtle differences, we want to take a closer look at some of the definitions and narrow down our own one.

Spitzner defines honeypots as a "security resource whose value lies in being probed, attacked, or compromised."[Spi03]

High-interaction honeypots

Low-interaction honeypots

Pure honeypots

2.2.2 Honeynets

[Spi03]

2.2.3 Legal Issues

[Spi03]

2.3 Intrusion Detection System

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]

4 Practical Work

4.1 Attack vectors

4.1.1 Primer

4.2 Proposed Honeypots

4.2.1 Cowire

4.2.2 Dionaea

4.2.3 Honeyd

4.3 Concept

4.3.1 HoneyTrap

4.4 Implementation

5 Experimental Work

Connect results of Honeypots with NIDS/IDS to update rules.

5.1 SNORT

6 Evaluation

6.1 T-Pot

6.2 Analyzation

7 Conclusion

7.1 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.

[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.

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

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

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

[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.