

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor

Technische Hochschule Wildau

Fachbereich Wirtschaft, Informatik, Recht

Studiengang Verwaltungsinformatik Brandenburg (B.Sc.)

Thema (deutsch): Malware-Analyse - Wie Forensik-Tools helfen die Funktionsweise von Cyberangriffen zu verstehen

Thema (englisch): Malware-Analysis - How to enhance the understanding of cyberattacks using forensic-tools

Autor/in: Patrick Lucas Büdke

Seminargruppe: VIB1/19

Betreuer/in: Dr. Frank Seeliger

Zweitgutachter/in: Herr Falko Benthin

Inhaltsverzeichnis

Gendererklärung	III
Zusammenfassung	IV
Abstract	V
Abbildungsverzeichnis	VI
Tabellenverzeichnis	VII
Abkürzungsverzeichnis	VIII
Glossar	IX
1 Einleitung	1
1.1 Thema der Arbeit und Motivation	1
1.1.1 Relevanz	1
1.2 Forschungsstand	2
1.3 Aufbau der Arbeit	2
1.4 Ziel der Arbeit	2
2 Theoretische Grundlagen	4
2.1 Terraform	4
2.1.1 Funktionsweise	4
2.1.2 Vorteile	6
2.2 Cloud Computing	7
2.2.1 Die Arten von Cloud-Diensten	7
2.3 Virtuelle Maschinen	9
2.3.1 Was ist eine virtuelle Maschine?	9
2.3.2 Erstellen einer virtuellen Maschine	9
2.3.3 Absicherung einer virtuellen Maschine	10
2.4 Malware	11
2.4.1 Was ist Malware und wie funktioniert sie?	11
2.4.2 Arten von Malware	11
2.4.2.1 Viren	12
2.4.2.2 Trojaner	12
2.4.2.3 Würmer	13
2.4.2.4 Spyware/Adware	13
2.4.2.5 Scareware	14
2.4.2.6 Ransomware	14
2.4.2.7 Root kit	15

2.4.2.8	Backdoors	16
2.5	Malware-Analyse	17
2.5.1	Statische Analyse	17
2.5.2	Dynamische Analyse	18
3	Methodik	19
3.1	Virtuelle Umgebung	19
3.1.1	Konfiguration der Umgebung	19
3.2	Terraform	20
3.2.1	Konfiguration der Server	20
3.2.2	Konfiguration der Firewall	21
3.3	Tools	23
3.4	Malware	23
4	Ergebnisse	24
5	Diskussion	25
5.1	Analyse der Ergebnisse	25
5.2	Bewertung der Ergebnisse	25
5.3	Ausblick	25
6	Fazit	26
6.1	Zusammenfassung der Ergebnisse	26
7	Literatur- und Quellenverzeichnis	27
	Literatur- und Quellenverzeichnis	27
8	Anhang	28
A	Anhang A	28

Gendererklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Bachelorarbeit auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Formulierungen gelten gleichermaßen für alle Geschlechter.

Zusammenfassung

Diese Arbeit untersucht die automatisierte Bereitstellung einer virtuellen und abgesicherten Umgebung, mit welcher schädliche Software analysiert werden kann. Diese Umgebung läuft auf virtuellen Maschinen (VMs), die in kurzer Zeit mit dem Infrastructure-as-Code-Tool (IaC-Tool) Terraform von HashiCorp erzeugt werden. In dieser Umgebung kann Malware beobachtet und analysiert werden, ohne dass es Folgen nach sich zieht. Hier sind zudem, für die Malware-Analyse nützliche Tools installiert.

Abstract

Abbildungsverzeichnis

1	Terraform - Planungsphase	5
2	Aufbau der Cloud-Umgebung	19

Tabellenverzeichnis

Abkürzungsverzeichnis

VM Virtuelle Maschine

SSH Secure Shell

DNS Domain Name System

IaaS Infrastructure as a Service

PaaS Platform as a Service

SaaS Software as a Service

OS-Image Operating System-Image

SSD Solid State Drive

HCL HashiCorp Configuration Language

IT Informationstechnik

vgl. vergleiche

Glossar

Admin(istrator) - ist ursprünglich der Verwalter eines Netzwerkes. Er hat die meisten Rechte und darf auf alle Einstellungen zugreifen

App - ist eine Anwendung

Backdoor - bezeichnet einen Teil einer Software, der es dem User ermöglicht, unter Umgehung der normalen Zugriffssicherung Zugang zum Computer oder einer sonst geschützten Funktion eines Programms zu bekommen

Browser Plugin - sind kleine Zusatzprogramme, die innerhalb von Browsern die Inhalte und Funktionen erweitern.

Cloud - ist in der IT die Abkürzung für Cloud Computing und wird auch als Rechnerwolke oder Datenwolke bezeichnet

Cloud Computing - beinhaltet Technologien und Geschäftsmodelle um IT-Ressourcen dynamisch zur Verfügung zu stellen und ihre Nutzung nach flexiblen Bezahlmodellen abzurechnen

Cloud Provider - Anbieter für Cloud Computing

Cloud Server - ist ein virtueller Server. Dieser bezieht seine Hardwarekomponenten von externen Dienstleistern und wird über das Internet angeboten.

Download - das Herunterladen von Software

Dropper

Firewall - ist ein Sicherungssystem um Netzwerke und Computer vor unerwünschten Zugriffen zu schützen

Hardware

HashiCorp

Hashwert

Host-System

Infrastructure-as-Code-Tool

Key-Logger

Linker

Malware

Netzwerkressourcen

Open-Source-Software - ist Software deren Quellcode der Öffentlichkeit zugänglich ist. Die Benutzung des Codes steht dem Benutzer selbst offen.

Oracle

OS-Images - ist eine komprimierte Sammlung von Referenzdateien und Ordnern, die zum Installieren und Konfigurieren eines neuen Betriebssystems auf einem Computer verwendet werden

Provisioning

Rechenleistung

Remote-Access

Rootkits

Schadprogramm (Code)

Shell

Software

Speicherplatz

System Calls

Ubuntu

Wiki - ist eine Sammlung von Informationen und Beiträgen im Internet zu einem bestimmten Thema, die von den Nutzern selbst bearbeitet werden können

Workflow - die Abwicklung arbeitsteiliger Vorgänge

1 Einleitung

1.1 Thema der Arbeit und Motivation

In dieser Arbeit geht es um die Nutzung virtueller Umgebungen für die Analyse von schädlicher Software in einem automatisierten Verfahren. Dabei entstehen im Bereich der virtuellen Umgebung besonders komplexe Probleme, wie die Auswahl der Tools, des Betriebssystems oder des Cloud-Anbieters. Außerdem wird die Frage aufgeworfen, wie eine virtuelle Umgebung automatisiert und flexibel erstellt werden kann. Dabei ist die Unabhängigkeit vom Anbieter besonders wichtig, da das Malware-Analyse-Labor so flexibel sein soll, dass es bei vielen verschiedenen Anbietern mit wenig Aufwand konfiguriert und verwendet werden kann.

Wenn diese Umgebung erstellt wurde, ist es wichtig, dass diese sicher ist, denn Malware sollte nicht auf Rechnern mit sensiblen Daten Zugriff haben. Das heißt, dass die Art, wie die virtuelle Umgebung nach außen abgesichert ist, eine grundlegende Rolle für die Funktion dieser spielt. Allgemein soll durch diese Arbeit die persönliche IT-Sicherheit vereinfacht und verbessert werden. Daher ist die Absicherung einer der wichtigsten Punkte. Zu diesen wichtigen Punkten zählt unter anderem auch die Auswahl der Tools. Diese legt fest, wie die Umgebung später genutzt wird.

Weitere Punkte, die außerdem aufgegriffen werden, sind die Konfiguration der Kommunikationsmöglichkeiten, die der Analyse-Umgebung zur Verfügung stehen. Um hier feste Aussagen treffen zu können, muss die allgemeine Funktionsweise verschiedener Malwarearten durchblickt werden.

1.1.1 Relevanz

Malware-Analyse gibt Auskunft über das Verhalten von der untersuchten Schadsoftware. Die Informationen, die dabei gewonnen werden, können für die Sicherheitsmaßnahmen des Zielsystems wichtige Aufschlüsse darüber geben, wie das Eindringen dieser Malware-Art in das System in Zukunft unterbunden werden kann.

Außerdem kann die Funktionalität der Malware unterbunden werden, wenn bekannt ist, mit welchen Ressourcen die Malware arbeiten muss, um zu funktionieren. Alles in Allem bietet Malware-Analyse eine große Bereicherung für die IT-Sicherheit. Das Problem dabei ist oft, dass die notwendige Umgebung für die Analyse von Malware nicht überall vorhanden ist. Es besteht die Möglichkeit des Mietens solcher Umgebungen bei externen Firmen, was aber teilweise sehr kostspielig ist. Die Grundlage für eine Alternative dazu soll in dieser Arbeit geschaffen werden.

1.2 Forschungsstand

Malware-Analyse und die Erstellung dieser ist ein Thema mit hohem Stellenwert in der IT. Der Forschungsstand zu einzelnen Malwarearten, sowie zu der Malware-Analyse, ist ein dementsprechend gut erforschtes Themengebiet mit vielen wissenschaftlichen Publikationen. Auch "Terraform" als Provisioning-Tool für Cloud-Umgebungen ist nicht unbekannt. Es finden sich daher auch hierzu wissenschaftliche Artikel und Publikationen, die den Umgang und die Möglichkeiten des Tools betrachten.

In dieser Arbeit sollen diese beiden Parts zusammengefügt werden, denn bei der Recherche ist auffällig geworden, dass die beiden Themen nur separat voneinander untersucht wurden. Daher sollen sie hier verbunden werden, um die Möglichkeiten aufzuzeigen, die Terraform in der Welt der Malware-Analyse bietet. Dazu liegen bis zum jetzigen Zeitpunkt wenig bis keine Arbeiten und Publikationen vor.

1.3 Aufbau der Arbeit

Zunächst wird im zweiten Kapitel eine theoretische Grundlage geschaffen. Diese ist untergliedert in Grundlagen zu "Terraform", zu Cloud Computing, zu virtuellen Maschinen und zu Malware, sowie ihrer Analyse.

Anschließend befasst sich Kapitel drei mit dem praktischen Teil dieser Arbeit. Dafür wird das Vorgehen mit den einzelnen Werkzeugen, um das Ziel zu erreichen, beschrieben. Hierbei zeigt der Teil Terraform 3.1 auf, wie die Konfigurationsdateien aussehen. Im Anschluss dazu geht es um die virtuelle Umgebung, die aus den entsprechenden Konfigurationsdateien hervorgeht. Die Tools, die hier installiert sind, werden im nächsten Punkt genauer erläutert. Hier wird einerseits auf die einzelnen Tools eingegangen, andererseits ist hier auch der Zweck dieser Tools erklärt.

Im vierten Kapitel werden die Ergebnisse dargestellt. Dazu wird die tatsächliche Vorgehensweise dargelegt und die Verwendung, der in Kapitel drei genannten Methoden, aufgezeigt. Außerdem werden hier die Ergebnisse und deren Aussagekraft über die Forschung eingeordnet.

Das fünfte Kapitel ist die Diskussion, in welcher die Ergebnisse genauer interpretiert und bewertet werden. Auch mögliche Verbesserungsvorschläge sind hier aufzugreifen. Als Abschluss dieser Arbeit steht das Fazit im sechsten Kapitel. Dort werden die wichtigsten Erkenntnisse, aus der Forschung, im Rahmen der Arbeit dargelegt. Anschließend folgt das Literaturverzeichnis, sowie der Anhang.

1.4 Ziel der Arbeit

Das konkrete Ziel dieser Arbeit ist es, die Möglichkeit zu bieten, mithilfe bestimmter Tools eine virtuelle Umgebung für die Analyse von Malware zu erstellen. Der Fokus liegt darauf, dass die Flexibilität dieser Umgebung geboten wird. Außerdem sollen für

jede Analysetechnik Tools zur Verfügung stehen. Ein weiterer Faktor, der beachtet wird, ist, dass die Kosten für diese Umgebung sehr günstig im Vergleich zu den kommerziellen Lösungen sind. So ist das Ziel, jedem, der Malware analysieren möchte oder muss, eine kostengünstige, erste Grundlage bieten zu können, um die Analyse durchführen zu können.

2 Theoretische Grundlagen

2.1 Terraform

2.1.1 Funktionsweise

Terraform ist eine *Open-Source-Software*, welche die Vorbereitung von *Cloud Servern* einfacher macht. Sie wurde von HashiCorp dazu entwickelt, Infrastrukturen vorzubereiten und zu verwalten.¹ Mit Infrastrukturen sind hier *virtuelle Maschinen* verschiedenster Anbieter gemeint.² Terraform hat lizenzierte Provider für die seine Dienste anwendbar sind.³ Durch die hohe Konnektivität dieser Software sind die Einsatzorte sehr vielseitig. Das macht die Software in der Entwicklung und im Betrieb von Unternehmen sehr attraktiv.

Terraform ist ein Infrastructure-as-Code-Tool und wird für die Bereitstellung von physischen als auch virtuellen Servern verwendet. Das Tool arbeitet mit Konfigurationsdateien, die in der HashiCorp Configuration Language(HCL) geschrieben werden. Die genannten Dateien sind von Menschen lesbar. Zudem ist HCL eine deklarative Sprache. Das heißt, dass der Nutzer den Wunschaufbau der Cloud-Umgebung in der Terraform-Datei beschreibt. Die einzelnen Schritte um den gewünschten Zustand zu erreichen, werden von Terraform übernommen.

Die Prozesse in die die Arbeit mit Terraform unterschieden werden kann, sind in drei Schritte aufgeteilt. Der erste Schritt ist `write`. In dieser Phase wird der Code definiert. Dafür werden die benötigten Ressourcen für die jeweiligen Provider definiert und Terraform-Konfigurations-Dateien mit der Endung `.tf` verwendet. Aus diesen Dateien entnimmt Terraform beispielsweise welcher Cloud-Anbieter verwendet wird. Als Beispiel für einen Anbieter mit dem Terraform funktioniert kann die "Hetzner-Cloud" genannt werden. Zu den Providern findet sich auf der Seite von Terraform eine jeweilige Dokumentation darüber, wie die Ressourcen einzubinden sind. Der Code der Konfigurations-Dateien entscheidet darüber, wie die virtuelle Umgebung konfiguriert wird, wenn das Skript ausgeführt wird.

¹Wang, 2022.

²Brikman, 2019

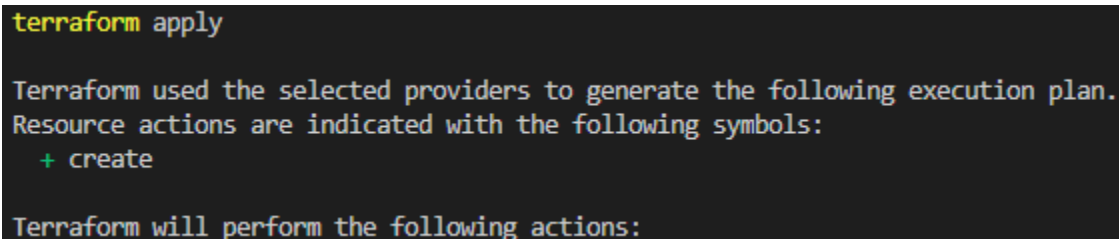
³HashiCorp, 2022.

Code Listing 1: Block in HCL - Syntax

```
1  #Create a server
2  resource "hcloud_server" "malware" {
3      name          = "forensics"
4      image          = "ubuntu-22.04"
5      server_type    = "cx11"
6      firewall_ids  = [hcloud_firewall.saferfw.id]
7      ssh_keys       = [hcloud_ssh_key.default.id]
8      user_data      = file("user_data.yml")
9  }
```

Im obigen Code ist der allgemeine Aufbau eines Blocks in Terraform zu erkennen. Ein Block beginnt mit dem `type`. Er legt auch fest, welche und wie viele sogenannte Bezeichner folgen müssen. Dieser Typ ist hier in Zeile zwei als `"resource"` festgelegt. Als Bezeichner folgen `"hcloud_server"` und `"malware"`. Wir definieren hier also einen Server in der Hetzner-Cloud mit dem Namen `"malware"`. Der Inhalt des Blocks befindet sich in geschweiften Klammern. In diesen Klammern sind zusätzliche Argumente zu finden. Diese bestimmen die Abhängigkeiten und die Zusammenhänge mit anderen Blocks und Argumenten. Die Abhängigkeiten von anderen Netzwerkkomponenten sind in Zeile sechs zu erkennen. Hier wird die Firewall dem Server zugewiesen. Auch der SSH-Key ist in Zeile acht festgelegt.

Wenn der Code verfasst wurde, folgt der nächste Schritt im Arbeitsablauf von Terraform. Hier sollten die Konfigurationsdateien, die verfasst wurden, initialisiert werden. Dies geschieht mit dem Befehl `terraform init`. Anschließend kann für das weitere Vorgehen ein `plan` erstellt werden. Durch `terraform apply` wird dieser Plan erstellt und ausgegeben. Um Fortzufahren muss der Plan, indem jeder Schritt aufgelistet ist, den Terraform durchführen wird, bestätigt werden. Diese einzelnen Schritte können Änderungen, Löschungen oder das Hinzufügen von Dateien sein.



```
terraform apply

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:
```

Abbildung 1: Terraform - Planungsphase

Die Dritte und letzte Phase im Terraform-*Workflow* ist `terraform apply`. Sie beginnt mit der Bestätigung des Plans. Ab diesem Zeitpunkt beginnt Terraform mit der Bereit-

stellung der Cloud-Umgebung. Woraus diese besteht, ist von den Konfigurationsdateien abhängig.

2.1.2 Vorteile

Terraform birgt einige nützliche Vorteile. Einer dieser ist die Flexibilität. Durch die Vielzahl an Providern kann diese gewährleistet werden. Diese Provider lassen das Provisioning einer Cloud-Umgebung mittels Terraform zu. So kann beispielsweise eine Konfiguration für einen Hetzner-Server mit wenigen Änderungen verwendet werden, um eine virtuelle Umgebung bei "Oracle " zu erstellen. Diese Flexibilität ist besonders für dieses Projekt interessant, denn so kann die virtuelle Umgebung bei verschiedenen Anbietern konfiguriert und verwendet werden.

Ein weiterer Vorteil, der sich daraus ergibt, ist, dass Terraform an sich sehr schnell ist. Mit den Befehlen `init` und `apply` wird eine virtuelle Umgebung mit den gewünschten Konfigurationen erstellt. Das ist gegenüber dem manuellen Einrichten einer virtuellen Umgebung sehr zeitsparend.

Hinzu kommt, dass Terraform gut zu verstehen ist. Die Sprache in der die Konfigurationsdateien geschrieben ist, ist sehr intuitiv und gut verständlich. Wenn die Syntax einmal begriffen wurde, lässt sich aus den Konfigurationsdateien gut auf das Endprodukt schließen. Natürlich sind virtuelle Umgebungen, die von Grund auf komplex sind, auch in Terraform komplex. Dennoch ist die Verständlichkeit der Sprache eine positive Anmerkung wert.

Weiterhin ist für IT-Teams die Planungsphase ein gutes Tool, um Veränderungen bereits im Vorweg zu sehen und Folgen abschätzen zu können. Dabei können Fehler, die zu einem unerwünschten Ergebnis führen, frühzeitig erkannt und vermieden werden.

2.2 Cloud Computing

Cloud Computing (*deutsch Rechnerwolke oder Datenwolke*) ist das Bereitstellen von einzelnen Anwendungen bis hin zu Rechenzentren. Die Ressourcen hierfür werden aus der *Cloud* genommen, welche auch als *Internet* bekannt ist. Durch flexible Einstellungsoptionen ist es durch Cloud Computing möglich IT-Systeme bedarfsspezifisch und flexibel anzubieten. Dabei ist auch zu beachten, dass Cloud Computing viele Vorteile mit sich bringt. Darunter fallen beispielsweise die Kosten und die Skalierbarkeit virtueller Systeme.

Cloud Computing wird in drei Arten aufgeteilt: Public (öffentlich), Privat (privat) und Hybrid (Mischung aus öffentlich und privat). Bei der öffentlichen Form besitzt ein Dienstleister die benötigten Ressourcen, welchen man *Cloud Provider* nennt. Dabei werden die, den Kosten entsprechenden Komponenten, dem Kunden über das Internet zur Verfügung gestellt. Hier spielen die Kosten, als Vorteil, eine große Rolle, da die Hardware nicht selbst angeschafft werden muss. Zudem sind Hardware-Wartungen nicht nötig, da diese vom *Cloud Provider* übernommen werden. Durch die Vielzahl an Komponenten, die große Anbieter bieten können, steigt die Skalierbarkeit sehr stark. Die Servermenge in den Rechenzentren der Provider ist außerdem ausschlaggebend dafür, dass das Risiko eines Ausfalls minimiert wird.

Die private Form des Cloud Computings unterscheidet sich von der öffentlichen Form. Besonders im Aspekt des Zugriffs auf die Hardware treten Unterschiede auf. Während bei der öffentlichen Form die selbe Hardware von verschiedenen Kunden genutzt wird, wird die Hardware bei privaten Cloud Computing-Lösungen nur einem Kunden zugewiesen. Diese kann hierbei im Besitz des Kunden sein und sich in seiner Organisation befinden. Auch kann sich die Hardware in Räumen von Drittanbietern befinden. Dabei ist sie dennoch nur für einen Käufer der Systemkomponenten bestimmt. So können Unternehmen ihre Netzwerkrichtlinien souverän bestimmen. Zudem gewährleistet die private Nutzung von Hardware mehr Sicherheit.

In der hybriden Form sind die beiden vorangegangenen Modelle vereint. Ziel dabei ist es die Vorteile beider Formen in einer Form zu verschmelzen. Beispielsweise kann die große Flexibilität von öffentlichen Clouds benutzt werden. Währenddessen stellt die private Cloud, durch ihre höhere Sicherheit, eine Zone für Komponenten des Netzwerkes da. Aus der Zusammenführung der beiden Modelle ergibt sich eine Form, die die Vorteile vereint. Die Netzwerkkontrolle ist geboten, die Kosteneffizienz ist hoch und es wird ein hohes Maß an Flexibilität und Skalierbarkeit geboten.

2.2.1 Die Arten von Cloud-Diensten

Infrastructure as a Service (IaaS) oder zu deutsch: *Infrastruktur als Dienstleistung* stellt die notwendigen Systemkomponenten bereit. Dazu zählen beispielsweise der Arbeitsspeicher, die Festplatte oder der Prozessor. Diese Komponenten können von Drittanbietern oder von der Organisation selbst bereitgestellt werden. In dieser Kategorie sind *Ama-*

von *Web Services*, *Microsoft Azure* oder auch die *Hetzner Cloud* einzuordnen. Diese bieten das digitale bereitstellen von Cloud-Umgebungen an. Diese Umgebungen können an die Anforderungen angepasst und flexibel erweitert werden. So zahlt der Nutzer nur für das, was er benutzt. Zudem muss er so keine Hardware für seine Cloud-Umgebung selber bereitstellen.

Platform as a service (PaaS) oder zu deutsch: *Plattform als Dienstleistung* stellt die Plattform für Entwicklungen und Anwendungen bereit. Diese wird von der folgenden Dienstleistung *Software as a service (SaaS)* benötigt. Nutzer von PaaS sind zum Beispiel Anwendungsentwickler die in großen Teams zusammen an einem Projekt arbeiten. Als Vertreter von *PaaS* -Produkten kann *Google App Engine* genannt werden. Hier stehen für den Nutzer verschiedene Entwicklungssprachen zur Verfügung. Der Vorteil ist, dass viele Aufgaben die Entwickler übernehmen müssten um eine skalierbare Software zu schreiben, von Google übernommen werden. Somit werden die Arbeitszeit und Kosten verringert.

Software as a service (SaaS) befasst sich mit der Software von Systemen. Eine Software ist eine Anwendung für einen Computer. Der Inhalt der Software legt fest was bei seiner Ausführung auf dem Computer passiert. Der ausführende Teil wird von der Hardware übernommen. Bei Software aus dem Modell *SaaS* werden Anwendungen, die cloud-basiert arbeiten können angeboten. Diese sind in der Regel skalierbar, sodass sie auf Wünsche einzelner Organisationen eingehen können. Darunter fällt beispielsweise das Tool des Softwarekonzerns *SAP Enterprise Resource Planning*. Dieses Tool ist für Unternehmen gedacht und kann Prozesse im Zusammenhang betrachten und beispielsweise analysieren. Hier wird klar, dass die Skalierbarkeit von *SaaS* große Vorteile bringt. So kann Software individuell auf Unternehmen angepasst werden, um den einzelnen Anforderungen gerecht zu werden.

2.3 Virtuelle Maschinen

2.3.1 Was ist eine virtuelle Maschine?

Virtuelle Maschinen bilden mit Infrastructure as a Service (IaaS) eines der drei Servicemodelle des Cloud Computings. Sie beruhen auf der Idee, Hardware, Rechenleistung, Speicherplatz und Netzwerkressourcen aus der Cloud bereitzustellen. So soll der Benutzer von virtuellen Maschinen das gleiche Erlebnis haben, wie er es auch bei physischen Geräten haben würde.

Der Vorteil daran, dass virtuelle Maschinen ein aus dem Internet bereitgestellter Dienst sind, ist die Flexibilität. Durch die Verwendung von virtuellen Maschinen können Kosten gespart werden. Das rührt daher, dass der Kunde sein System den Anforderungen entsprechend planen und umsetzen kann. Ist die VM einmal aufgesetzt und soll vergrößert werden, muss nicht die Hardware ausgetauscht werden. Stattdessen wird die Größe der virtuellen Umgebung individuell angepasst. Somit einher geht auch die Änderung der Kosten für das System.

Zudem ist die Unabhängigkeit vom Host-System ein Vorteil. In vielen Unternehmen wird Windows als graphisches Betriebssystem benutzt. Dennoch kann es sein, dass ein Rechner mit einem Linux-System für manche Prozesse besser geeignet ist. Alle Rechner auf Linux umzustellen nur, weil bestimmte Abläufe so besser funktionieren, ist aber auch keine Lösung. Auch die Anschaffung eines einzelnen Rechners, um Linux zu benutzen, ist nicht effizient und praktisch. Stellvertretend dafür kann Linux auf einer virtuellen Umgebung laufen. Die Kosten dafür sind je nach Gerätkonfiguration weit unter dem Hardwarepreis für ein Gerät mit gleicher Ausstattung. Zudem ist die Verfügbarkeit des Gerätes gewährleistet, da theoretisch von überall, wo es einen freien Internetzugang gibt, auf die Cloud zugegriffen werden kann.

2.3.2 Erstellen einer virtuellen Maschine

Eine virtuelle Maschine kann in der Regel bei einem *Cloud Provider* erstellt werden. Bevor jedoch eine VM erstellt wird, muss darüber nachgedacht werden welchem Zweck die VM dient. Daraus ergeben sich wichtige Konfigurationsmerkmale, wie das Betriebssystem oder die Sicherheitskonfigurationen. Nach dieser Planungsphase kommt es zum Erstellen der VM. Hierfür kann die grafische Umgebung, der jeweiligen Provider, genutzt werden. Das Erstellen eines einfachen Servers ist schnell und unkompliziert gemacht. Für die Verbildlichung einer solchen Weboberfläche, wird in dieser Arbeit die *Hetzner-Cloud* verwendet. Hierbei wird zuerst über das Feld "NEUES PROJEKT" ein Projekt erstellt. In diesem Projekt kann mithilfe des Buttons "SERVER HINZUFÜGEN" ein Server erstellt werden. Der Server hat die Konfigurationsoptionen:

Standort - bestimmt woher die Serverleistungen kommen sollen

Image - ist in die Reiter *OS-Images* und *Apps* aufgeteilt. Es bietet die Möglichkeit das Betriebssystem und Tools zu installieren.

Typ - hier kann die Rechenleistung, Festplattengröße und das maximale Datenaufkommen festgesetzt werden. Der Preis der jeweiligen Cloudlösung ist mit aufgelistet.

Volume - hier kann SSD-Speicher festgelegt werden

Networking - Festlegen über welche Netze der Server kommunizieren darf

Firewalls - können hinzugefügt werden, nachdem sie in einem anderen Teil der *Hetzner Cloud* erstellt wurden

Zusätzliche Features - Festlegung von Benutzerdaten, Backups und Platzierungsgruppen

SSH-Key - hier kann der SSH-Key hinzugefügt werden

Name - Festlegung des Servernamens

Die virtuelle Maschine kann auch mit Terraform aufgesetzt werden. Durch die Vielzahl an Partnern sind Konfigurationen in Terraform besser an andere Provider anpassbar und werden somit wiederverwendbar. Dafür müssen in den Konfigurationsdateien von Terraform nur wenige Ressourcen geändert werden. Wie eine VM mit Terraform aufgesetzt werden kann, wird in einem späteren Teil der Arbeit an einem Beispiel aufgegriffen(3.1).

2.3.3 Absicherung einer virtuellen Maschine

Die Sicherung der virtuellen Maschine kann je nach Einsatzgebiet eine große Bedeutung haben. Um Sicherheitskonfigurationen einzubinden, kann die Firewall bearbeitet werden. Auch dafür bietet Terraform eine Möglichkeit.

-Terraform

2.4 Malware

2.4.1 Was ist Malware und wie funktioniert sie?

Laut Definition ist Malware eine "böswärtige Software" (auf Englisch "malicious software").

Allgemein dient der Begriff Malware als Klassifizierung von Dateien oder Software, die Schäden verursachen, sobald sie sich im System des Benutzers befinden.

Die häufigsten Arten sind:

1. Viren
2. Trojaner
3. Würmer
4. Spyware/Adware
5. Scareware
6. Ransomware
7. Root kit und Backdoors

Dabei können sowohl persönliche Schäden als auch Sachschäden entstehen.

Um einen solchen Malware Angriff zu tätigen, benötigt man einen Schadcode. Mithilfe dessen verschaffen sich Angreifer Zugriff auf das System des Opfers und klauen Passwörter oder andere sensible Daten. Um den Datendiebstahl durchzuführen, muss ein Angreifer dafür sorgen, dass der Schadcode auf dem Zielsystem ausgeführt wird. Damit das klappt, betten diese den schädlichen Code in eine Datei ein, die das Opfer öffnen soll. Nach der Durchführung erfüllt der Schadcode seinen Zweck und installiert zum Beispiel eine Backdoor oder startet einen Key-Logger. Key Logger können unter anderem die Eingabe von Passwörtern protokollieren.

2.4.2 Arten von Malware

Neue Schadprogramm Varianten entstehen, wenn die Funktionsweise der neuen Malware grundlegend von bereits vorhandenen Schadprogrammen abweicht. Wichtig ist hierbei, dass der Hashwert einzigartig ist, aber ein neuer Hashwert nicht mit einer neuen Malware gleichzusetzen ist. Das Ergebnis einer Hashfunktion unterscheidet sich bereits bei geringfügigen Veränderungen im Programmcode. Das heißt, dass ein neuer Hashwert nicht eine neue Malware bedeutet, da die Funktionsweise des Schadcodes teilweise dieselbe bleibt.

Während für bekannte Schadprogramm-Varianten teilweise Detektionsmethoden existieren, sind neue Varianten unmittelbar nach ihrem Auftreten unter Umständen noch nicht als Schadprogramme erkennbar und daher besonders bedrohlich.

Zudem kann zwischen den Arten nicht sonderlich stark differenziert werden, da sie ineinander übergehen und ähnliche Methodiken und Ziele verfolgen.

2.4.2.1 Viren

Viren, die Programme befallen, bestehen, wie ausführbare Programme, aus Code. Der spezifische Code ist stark von der jeweiligen Hardwareplattform und auch dem verwendeten Betriebssystem abhängig. Das ergibt sich mitunter daraus, dass Programme grundsätzlich einige Funktionen des Rechners nicht direkt ansprechen und durch System Calls bewilligt werden müssen.

Da sich der Virencode in den Programmcode einbetten muss, ist ein Virus meistens auf eine bestimmte Plattform und ein bestimmtes Betriebssystem als Wirt festgelegt und kann Systeme mit anderer Plattform oder anderem Betriebssystem nicht infizieren.

Der ausführbare Code des Virus wird im Code des ursprünglichen Programms platziert und das Programm so modifiziert, dass beim Start zuerst der Virus aufgerufen wird.

Um länger unbemerkt zu bleiben, transferieren viele Viren danach die Kontrolle zurück an das ursprüngliche Programm, sodass dessen Funktion durch den Virus scheinbar nicht beeinträchtigt wird. Gelangt das Virus zur Ausführung, kann es weitere Programmdateien auf dem Rechner infizieren und gegebenenfalls weiteren Schadcode ausführen.

Viren müssen nicht unbedingt nur ausführbare Programme infizieren. Einige Viren nisten sich im Hauptspeicher des Rechners ein und bleiben dort durchgehend aktiv. Solche Viren werden als "speicherresistente" Viren bezeichnet.

In bestimmten Fällen können sich Viren auch über Dokumente, also eigentlich nicht ausführbare Daten, verbreiten. Dies ist unter anderem möglich, wenn die Anwendungsprogramme, welche die Dokumente aufrufen, Schwachstellen aufweisen, welche die Ausführung des injizierten Codes ermöglicht.

2.4.2.2 Trojaner

Trojaner sind Schadsoftware-Varianten, über die meist destruktive oder datenstehlende Malware auf ein System geschleust wird. Anders als Viren und Würmer, sind Trojaner nicht in der Lage, sich selbstständig zu replizieren oder Dateien zu infizieren. Sie tarnen ihre Malware als nützliches Programm und hoffen darauf, dass arglose Nutzerinnen und Nutzer sie eigenhändig installieren. Täuschung ist hierbei ihre Verbreitungsstrategie. Häufig kommen Trojaner in fingierter Software vor, die von Angreifern manipuliert wurde. Diese Software ist zumeist als Download in unseriösen Quellen verfügbar.

In den meisten Fällen bestehen Trojaner aus zwei eigenständigen Programmen, die auf verschiedene Weise miteinander verknüpft sein können. Sogenannte Linker heften das Schadprogramm an eine ausführbare Wirtssoftware. Wird das vermeintlich nützliche Programm ausgeführt, startet gleichzeitig auch der Schadcode im Hintergrund.

Eine zweite Möglichkeit ist der Einsatz eines Droppers, der beim Start des Wirtsprogramms heimlich die Schadsoftware auf dem System ablegt. Während die Ausführung des schädlichen Programms im ersten Fall vom Wirt abhängig ist, kann es bei Einsatz des Droppers völlig unabhängig vom Trojaner agieren.

Die dritte Möglichkeit ist die Integration des geheimen Codes in eine Wirtssoftware, wie es zum Beispiel bei vielen Browser-Plugins der Fall ist. Auch hier ist die Ausführung des schädlichen Programms an die Wirtssoftware gebunden. Wird diese beendet oder gelöscht, stehen auch die geheimen Funktionen nicht mehr zur Verfügung.

Weil der Trojaner in der Regel durch den Anwender selbst gestartet wird, hat er die gleichen Rechte wie der angemeldete Benutzer. Folglich kann er alle Aktionen ausführen, die auch der Nutzer ausführen könnte.

2.4.2.3 Würmer

Bei einem Computerwurm handelt es sich um eine Malware, die sich selbstständig reproduziert und sich über Netzwerkverbindungen verbreitet. Der Computerwurm infiziert dabei normalerweise keine Computerdateien, sondern einen anderen Computer im Netzwerk. Dies geschieht, indem sich der Wurm repliziert. Diese Fähigkeit gibt der Wurm seinem Replikat weiter, wodurch auch dieser auf die gleiche Art und Weise andere Systeme infizieren kann.

An der Stelle zeigt sich auch der Unterschied zwischen Computerwürmern und -viren. Computerwürmer sind eigenständige Programme, die sich selbst replizieren und im Hintergrund laufen, während Viren eine Host-Datei benötigen, die sie infizieren können. Aus diesem Grund kommt es häufig vor, dass ein Computerwurm erst bemerkt wird, wenn das Programm Systemressourcen verbraucht, wodurch andere Aufgaben verlangsamt oder angehalten werden.

2.4.2.4 Spyware/Adware

Bei Spyware handelt es sich um eine Software, die ohne Wissen des Anwenders Aktivitäten auf dem Rechner oder im Internet ausspioniert und aufzeichnet. Dabei werden Informationen weitergeleitet und durch verschiedene Methoden die Daten gesichert. Häufig wird hierbei das Keylogging verwendet um Benutzernamen, Passwörter und Bankdaten herauszufinden. Aufzeichnungen von Audio- und Videodaten sind ebenso wie das Erfassen von Inhalten aus E-Mail-, Messaging- und sozialen Apps keine Seltenheit. Jede Tätigkeit, die auf dem Rechner ausgeführt wird, ist nachvollziehbar. Spyware kann nicht nur gefährlich sondern auch aufdringlich werden. Adware installiert sich selbst ebenso heimlich auf dem Rechner und spioniert den Browserverlauf aus, um mit passenden Anzeigen den User zu belästigen. Auch in der Gaming-Szene spielt Spyware eine immer größer einhergehende Rolle. Viele Programmierer lassen in der Installationkonsole eine Red-Shell-Spyware mitinstallieren, mit der die Online-Aktivität des Spielers verfolgt werden soll, um in Zukunft bessere Spiele zu veröffentlichen, die angepasster an den Verbraucher sind. Der Verbraucher wurde weder in Kenntnis davon gesetzt, noch hat dieser der Installation aktiv zugestimmt. Es handelt sich also demnach um Spyware.

2.4.2.5 Scareware

Scareware ist eine Art von Malware, die einen Virus oder ein anderes Problem auf einem Gerät zu erkennen vorgibt. Die Benutzer sollen zur Behebung des Problems schädliche Software downloaden oder kaufen. Üblicherweise ist Scareware die Vorstufe für einen Cyberangriff und nicht ein Angriff an sich.

Scareware-Angriffe beginnen häufig mit einer Pop-up-Werbung, die den Eindruck erweckt, sie stamme von einer Sicherheitssoftware oder vom Betriebssystem des Rechners. Klicken die Benutzer darauf, werden sie auf eine infizierte Webseite geleitet. Auf dieser sollen Sie weitere Anweisungen zur Behebung ihres angeblichen Problems erhalten. Das umfasst beispielsweise die Installation eines neuen Tools oder Programms, einen Scan des Rechners, die Eingabe von Anmeldedaten, um mehr Informationen zu erhalten, oder das Hochladen von Kreditkartendaten für den Wiederherstellungsprozess. Häufig führt das dazu, dass Benutzer unwissentlich schädliche Programme wie bereits erwähnte Malware-Arten auf das Gerät herunterladen.

Scareware-Angriffe können auch über E-Mail erfolgen. Bei dieser Angriffsart verschicken die Angreifer E-Mails mit hoher Priorität oder Dringlichkeit, die Benutzer zum sofortigen Handeln auffordern. Die Links in der E-Mail täuschen den Benutzern vor, mit ihnen würden die Bedrohung behoben oder das System gescannt. Ein Klick darauf führt zum Download und zur Installation von infizierten Dateien, schädlichem Code oder Schadprogrammen.

Ebenso wie viele andere Arten von Malware sind Scareware-Angriffe sehr problematisch, weil die Betrüger Zugang zu den Konto- oder Kreditkartendaten der Benutzer erlangen können, was Identitätsdiebstahl oder andere Betrugsarten ermöglicht.

2.4.2.6 Ransomware

Ransomware ist eine Art von Schadsoftware. "ransom " bedeutet übersetzt Lösegeld, was schon eine Aussage über die Funktionalität der Malware trifft. Beispielsweise spricht man von Ransomware, wenn der Schadcode darauf programmiert ist, Daten eines vermeindlichen Opfers so zu codieren, dass diese nicht wieder hergestellt werden können. Somit hat der Angreifer ein großes Druckmittel in der Hand, mit dem das Opfer zur Zahlung von Lösegeld aufgefordert werden kann. Bemerkbar macht sich Ransomware durch Erpresserbriefe oder einen blockierten Bildschirm, welcher erst wieder freigegeben wird, wenn die eingeforderte Summe gezahlt wird. So können mitunter große Unternehmen zu sehr hohen Zahlungen aufgefordert werden, da hier eine Codierung der Daten sehr schädigend für das Unternehmen werden kann. Die Zahlung garantiert jedoch nicht, dass der Computer und die Daten wieder freigegeben werden. Als Schutz vor Datenverlust gelten regelmäßige Backups der Daten und Updates der Sicherheitssysteme.

2.4.2.7 Root kit

Bei einem Rootkit handelt es sich nicht um eine einzelne Malware sondern um eine Sammlung verschiedener Schadprogramme, die sich über eine Sicherheitslücke in einen Computer einnistet und Angreifern den dauerhaften ferngesteuerten Zugriff (Remote-Access) auf diesen erlaubt. Wesentliches Merkmal von Rootkits ist, dass sie sich vor Virenscoannern und Sicherheitslösungen verstecken können, sodass der Benutzer nichts von ihrer Existenz mitbekommt.

Je nachdem, auf welcher Berechtigungsstufe sich das Rootkit ausgebreitet hat, kann es dem Angreifer sogar umfassende Administrationsrechte verschaffen (in diesem Fall spricht man von einem Kernel-Mode-Rootkit), wodurch er die uneingeschränkte Kontrolle über den Rechner erhält.

Sie bestanden anfangs zumeist aus modifizierten Versionen standardmäßiger Programme wie „ps“ (ein Unix command, der eine Liste aller aktiven Prozesse aufruft) und „passwd“ (zum Ändern des Benutzerpassworts). Daraus ergibt sich die Bezeichnung "rootkit": Mit „Root“ wird bei Unix der Administrator bezeichnet, der Wortteil „kit“ bedeutet so viel wie „Ausrüstung“ oder „Werkzeugkasten“. Der zusammengesetzte Begriff Rootkit umschreibt somit ein Set von Software-Werkzeugen, das einen Angreifer dazu ermächtigt, Root-Rechte über einen Computer zu erlangen (gemeint sind Kernel-Mode-Rootkits).

Inzwischen existieren allerdings für eine Vielzahl von Betriebssystemen Rootkits. Auch für Windows- und andere Betriebssysteme ergibt die Bezeichnung „Rootkit“ durchaus Sinn: Denn einige Rootkits dringen bis in den Kernel, also den innersten Kern und damit die „Wurzel“ (Englisch: „root“) des Systems vor und werden von dort aus aktiv.

Die Infiltration eines Systems durch ein Rootkit lässt sich allgemein in folgende Punkte gliedern:

1. Infektion des Systems
2. Tarnung (Stealth)
3. Einrichtung einer Hintertür (Backdoor)

Infektion des Systems

Zunächst wird das Rootkit durch eine Sicherheitslücke oder einen Drive-By Download in den Computer eingemischt. An Passwörter und Zugangsdaten kommen die Angreifer in der Regel durch Beeinflussung oder bewusste Täuschung. Solche Sicherheitslücken sind demnach meist menschlicher Komponente. Je nach Berechtigungsstufe, kann das Rootkit verschiedene Ziele verfolgen, die eher recht oberflächlich sind oder aber auch im direkten Kern agieren (Kernel-Mode-Rootkit).

Tarnung (Stealth)

Sobald das Rootkit im System eingedrungen ist, verschleiert es seine Existenz. Dafür beginnt das Rootkit jene Prozesse zu manipulieren, über die Programme und Systemfunktionen Daten untereinander austauschen. Auf die Art erhält das Virenprogramm beispielsweise gefälschte Informationen, aus denen sämtliche Hinweise auf das Rootkit

herausgefiltert wurden.

Einrichtung einer Hintertür (Backdoor)

Die Backdoor wird im folgenden dann als Mittel zum Zweck genutzt, damit der Angreifer mittels Remote-Access, ausgespähtem Passwort oder Shell in das System gelangt. Das Rootkit verschleiert dabei jegliche Hinweise und ermöglicht dadurch weitere Installationen wie beispielsweise Keylogger.

2.4.2.8 Backdoors

Eine Backdoor (deutsch "Hintertüren") bezeichnet ein Teil einer Software, welche es dem Benutzer ermöglicht die Zugriffssicherung eines Systems zu umgehen und sich so Zugriff darauf zu verschaffen. Diese können bewusst vom Entwickler einer Software eingebaut sein. In diesem Fall können sie beispielsweise für die Reparatur von Systemen oder das Zurücksetzen von Adminpasswörtern verwendet werden. Jedoch kann eine Backdoor auch ungewollt auf ein System gelangen. Dabei wird über eine andere Malwareart eine Backdoor installiert. Dafür werden häufig Trojaner 2.4.2.2 verwendet. Eine Backdoor ist immer nur ein Teil einer Malware und keine eigene Malware. Um die Funktionalität der Backdoor zu gewährleisten, werden also weitere Malwarekomponenten benötigt. Als Beispiel dafür sind Widgets oder kleinere Programme zu benennen, die auf dem Zielsystem installiert sind und vermeintlich harmlos wirken. Dennoch können genau diese den ferngesteuerten Zugriff über eine Backdoor ermöglichen. Werden diese Programme beendet oder deinstalliert, kann auch die Backdoor nicht aufrecht erhalten werden und somit kann auch kein weiterer Fernzugriff stattfinden. Die Nutzung von Backdoors ist unterschiedlich. Häufig wird eine Backdoor zum Datenklau oder für die Installation weiterer Malware verwendet. Auch die Ausspähung des Zielsystems oder auch die Verschlüsselung einzelner auf dem Ziel befindlicher Daten kann hier als Ziel genannt werden.

2.5 Malware-Analyse

Malware kann sowohl statisch als auch dynamisch analysiert werden.

2.5.1 Statische Analyse

Bei der statischen Analyse wird das *Sample* der Malware analysiert ohne es auszuführen. Oft ist das auch der erste Schritt bei der Analyse, um eine Vorstellung darüber zu haben, wie sich die Schadsoftware verhalten könnte. Es gibt hierfür verschiedene Tools und Vorgehensweisen.

Für diese Art der Analyse werden beispielsweise Antivirus-Programme verwendet. Diese können in der Regel bestätigen, ob es sich bei dem vorliegenden Programm um Malware handelt. Um aussagekräftige Ergebnisse zu bekommen, müssen hier mehrere Antivirus-Programme benutzt werden. Virusscanner können eine Aussage über die Malware treffen, indem sie den "Fingerabdruck" der Datei mit den Fingerabdrücken in ihrer Datenbank vergleichen. Ein solcher "Fingerabdruck" ist einzigartig für jedes Programm. Er lässt sich in einem Hashwert abbilden. Um einen Hashwert zu ermitteln, stehen verschiedene Hashfunktionen zur Verfügung. Darunter zählen beispielsweise der "Message-Digest-Algorithm 5"(MD5) oder der "Secure Hash Algorithm" (SHA) in verschiedenen Ausführungen. Das Problem an Hashfunktionen ist, dass je nach Inhalt, der zu verschlüsselnden Datei, ein neuer Wert ermittelt wird. Das heißt, wenn der Programmierer des Schadcodes einen Part des Codes abändert, ist der Hashwert ein anderer. So hat sich an der Malware nichts verändert außer der Hashwert. Dadurch kann es vorkommen, dass Virens Scanner Malware aufgrund minimaler Änderungen nicht mehr erkennen können. Somit erkennt die Antivirusprüfung Malware nicht mit völliger Sicherheit. Um Antivirus-Programme zu täuschen, verwenden Hacker beispielsweise *msfvenom*. Dieses Tool hilft bei der Verschlüsselung von Code. So kann dieselbe Malware einen anderen Hashwert haben. Unter Umständen ist dieser auch nicht in den Datenbanken der Virens Scanner enthalten, wodurch die schädliche Software unentdeckt bleiben kann.

Um in der Analyse weiter voranzukommen und herauszufinden wie sich die Malware verhält, gibt es die Methode Strings aus dem Code zu lesen und zu deuten. Strings sind in der IT Zeichenketten. Diese sind in bestimmten Fällen im Code von ausführbaren Dateien, beispielsweise als *URLs*, enthalten. Sie können Auskunft über die Funktionsweise der Malware geben und sind somit ein wichtiger Faktor in der Analyse dieser. Um Strings aus Dateien auslesen zu können, müssen die entsprechenden Tools installiert sein. Anhand der Menge der gefundenen Strings kann vermutet werden, ob ein Programm verschachtelt oder verpackt wurde. Das ist eine Methode, die Hacker verwenden, um in einer Antivirus-Prüfung nicht aufzufallen. Womit das Programm verpackt wurde, kann unter Linux anhand der Software "Detect-It-Easy" aufgedeckt werden. Solche Programme geben die Art des sogenannten "Packers" aus. So kann die Malware wieder entpackt werden, um mehr Strings finden und auszuwerten zu können.

Zur grundlegenden statischen Analyse von Malware gehört auch zu überprüfen, ob verlinkte Bibliotheken oder Funktionen vorliegen. Diese Verbindungen zu verschiedenen Bibliotheken können sowohl statisch als auch dynamisch sein. Statisch bedeutet hierbei, dass der gesamte Code der Bibliothek in den Quelltext kopiert wird. Diese Methode wird jedoch selten verwendet. Öfter kommt die dynamische Verlinkung zu Bibliotheken zum Einsatz. Der Begriff "dynamisch" kommt daher, dass die Verbindung erst im laufenden Prozess hergestellt wird. Diese Verlinkungen können weitere Aussagen darüber treffen, wie sich die Malware im laufenden Zustand verhält.

2.5.2 Dynamische Analyse

Die dynamische Analyse von Malware ist die Untersuchung des Schadprogrammes während es ausgeführt wird. Hierbei wird das Verhalten der Malware während der *Laufzeit* beobachtet und analysiert. Außerdem ist die Analyse des Systems nach der Laufzeit ein Teil dieser Methode. Das ist nützlich, um den Schadcode einer groben Art von Malware zuzuordnen. Wichtig hierbei ist vor allem die Umgebung auf der das Schadprogramm ausgeführt wird. Es ist wichtig eine Art *forensisches Labor* zu erstellen in der kein Schaden für das Produktionssystem entstehen kann.

Um das Verhalten der Malware betrachten zu können muss diese ausgeführt werden. Es ist jedoch nicht immer einfach die Malware zu *triggern*. Das liegt vor allem an der unterschiedlichen Funktionsweise der Malware. Es gibt Malware, die nur über die Kommandozeile im Terminal ausgeführt werden kann. Diese braucht häufig Argumente, die die Aktion der Malware beeinflussen. Da man nicht jedes mögliche Argument kennen kann, welches die Malware benötigt, kann hier nur sehr schwer eine vollständige Analyse stattfinden.

Die ausführbare Datei der Malware unter Linux hat die Endung "DLL". Anders als auf Betriebssystemen mit graphischer Oberfläche kann diese jedoch nicht mit einem Doppelklick gestartet werden, sondern über die Kommandozeile.

3 Methodik

3.1 Virtuelle Umgebung

Um eine sichere virtuelle Umgebung zu schaffen, werden zwei Server benötigt. Ein Server bearbeitet DNS-Anfragen und der andere Server ist mit Malwareanalyse-Tools ausgestattet. Zudem sind keine Verbindungen zugelassen, außer Port 22. Dieser ist der SSH-Port und bietet hier die Möglichkeit den Server von außen zu beobachten. Ansonsten besteht die Möglichkeit für den Analyseserver sich mit dem Server zu verbinden der ein Netzwerk simuliert. Das ist daher nötig, da Malware teilweise nur mit Internetverbindung funktioniert.

3.1.1 Konfiguration der Umgebung

Wie bereits beschrieben besteht die Cloud-Umgebung aus zwei Servern und einem Rechner, der Zugriff auf die Server hat.

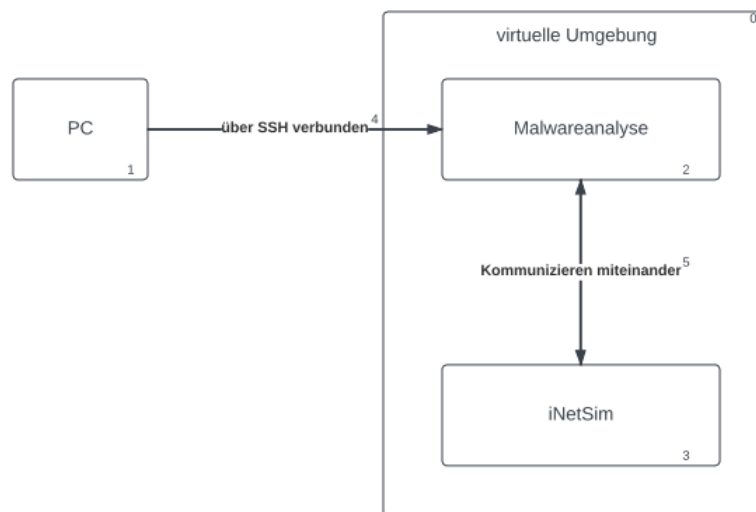


Abbildung 2: Aufbau der Cloud-Umgebung

In Abbildung zwei ist dieser Aufbau bildlich dargestellt. Auf der rechten Seite befindet sich die virtuelle Umgebung (mit null gekennzeichnet). Auf ihr laufen die beiden virtuellen Server. Einer der Server ist für die Malware-Analyse zuständig (mit zwei gekennzeichnet). Hier sind die Tools vorzufinden und die Malware wird hier eingeschleust. Außerdem gibt es den zweiten virtuellen Server, welcher ein Netzwerk simuliert (mit

drei gekennzeichnet). Dafür wird das Programm "iNetSim " verwendet. Dieses simuliert gängige Internetdienste und ist dafür gemacht bei der Malware-Analyse eingesetzt zu werden.

Diese beiden Server können über jeden Port miteinander kommunizieren, haben aber keine Verbindung nach außen, sondern bleiben in der geschlossenen Umgebung. Die Kommunikation der beiden Server ist in der Abbildung mit fünf gekennzeichnet. Die einzige weitere Verbindung die in diesem System hergestellt wird ist die Verbindung vom PC des Benutzers auf den Malware-Analyseserver über SSH (mit vier gekennzeichnet). Somit wird gewährleistet, dass man als Analyst der Malware das System steuern und beobachten kann.

Anhand des Aufbaus der Umgebung ist auch zu erkennen wie abgekapselt das System ist. Es bestehen keine Verbindungen nach außen, sondern nur untereinander, bzw von außen nach innen. Das ist auch wichtig um das Ausdringen der Malware zu verhindern. Diese Regeln für die Kommunikation sind in den Firewallregeln festgehalten.

3.2 Terraform

3.2.1 Konfiguration der Server

In Terraform können Server erstellt und entsprechend konfiguriert werden. Wie das gemacht wird, wird hier einmal aufgegriffen.

In HCL werden Blocks für die Erstellung von vielen Komponenten verwendet, darunter auch für die Servererstellung.

Code Listing 2: Konfiguration Malware-Server

```
1 # Create a server
2 resource "hcloud_server" "malware" {
3   name      = "malware"
4   image     = "ubuntu-22.04"
5   server_type = "cx11"
6   firewall_ids = [hcloud_firewall.saferfw.id]
7   ssh_keys = [hcloud_ssh_key.default.id]
8   user_data = file("user_data.yml")
9
10  public_net {
11    ipv4_enabled = true
12    ipv4 = hcloud_primary_ip.primary_ip_Malware.id
13    ipv6_enabled = false
14  }
15 }
```

Der abgebildete Code ist die Konfiguration für einen Server in der "Hetzner-Cloud".

Das ist bereits in Zeile zwei hinter "resource " zu erkennen. Der hier erstellte Server trägt den Namen "malware " und hat als Betriebssystem *Ubuntu* (Vgl. Zeile vier). Der Server-Typ (Zeile 5) legt fest, welche Cloud-Lösung des Providers verwendet werden soll. In der "Hetzner-Cloud " ist "cx11" die günstigste Version. Diese ist für die grundlegende Malware-Analyse dennoch ausreichend. Falls die Analyse durch stärkere Systemkomponenten beschleunigt werden soll, kann hier der Typ geändert werden.

Im Anschluss daran wird dem Server eine Firewall zugewiesen. Welche Einstellungen diese hat wird im nächsten Unterpunkt genauer betrachtet. Daraufhin bekommt der Server die Info woher der SSH-Key genommen wird (Zeile sieben). In Zeile zehn werden die öffentlichen Netzwerkeinstellungen vorgenommen. Dazu wird festgelegt, welche "Internet-Protokoll-Version " vorhanden ist. In diesem Fall ist "IPv4 " erlaubt und "IPv6 " nicht. Die "IPv4 "-Adresse wird dem Server fest zugewiesen. Das ist für die spätere Kommunikation mit dem anderen Server wichtig.

Code Listing 3: Konfiguration iNetSim-Server

```
1 #Create second Server
2 resource "hcloud_server" "iNetSim" {
3     name          = "iNetSim"
4     image          = "ubuntu-22.04"
5     server_type    = "cx11"
6     firewall_ids   = [hcloud_firewall.iNetSimFW.id]
7     ssh_keys       = [hcloud_ssh_key.default.id]
8
9     public_net {
10         ipv4_enabled = true
11         ipv4          = hcloud_primary_ip.primary_ip_iNetSim.id
12         ipv6_enabled = false
13     }
14 }
```

Die Konfiguration des zweiten Servers ist, wie hier zu sehen, sehr ähnlich zu der des ersten Servers. Der Unterschied hier ist nur die zugewiesene Firewall (Zeile sechs) und die IP-Adresse (Zeile elf).

3.2.2 Konfiguration der Firewall

Die Erstellung einer sicheren und durchdachten Firewall ist bei der Arbeit mit Malware sehr wichtig, um Schäden am eigenen oder fremden Systemen zu vermeiden.

Code Listing 4: Konfiguration der Firewall für den Analyseserver

```
1 #Create a firwall
2 resource "hcloud_firewall" "saferfw" {
3   name = "saferfw"
4   rule {
5     direction = "in"
6     protocol  = "tcp"
7     port      = "22"
8     source_ips = [
9       "0.0.0.0/0",
10      ":::/0"
11    ]
12  }
13
14  rule {
15    direction = "out"
16    protocol  = "udp"
17    port      = "1-65535"
18    source_ips = [
19      hcloud_primaryip.primary_ip_iNetSim.id
20    ]
21  }
22
23  rule {
24    direction = "out"
25    protocol  = "tcp"
26    port      = "1-65535"
27    source_ips = [
28      hcloud_primaryip.primary_ip_iNetSim.id
29    ]
30  }
31 }
```

Dieser Code stellt die Definition der Firewall für den Malware-Analyseserver dar. Der Name dieser Firewall ist "saferfw" und sie besteht aus drei Regeln. Die erste Regel (Zeile vier bis elf) legt die Kommunikation über den SSH-Port fest. Diese ist nur nach innen zugelassen. Wer diese Verbindung zum Server herstellt ist egal, solange der SSH-Key stimmt.

3.3 Tools

- ltrace
- Ghidra
- reverse engineering Tool
- binary analysis cookbook page 202
- EDB Debugger
- binary analysis cookbook

3.4 Malware

- kurze Erklärung, dass ich Malware hab

4 Ergebnisse

5 Diskussion

5.1 Analyse der Ergebnisse

5.2 Bewertung der Ergebnisse

5.3 Ausblick

Eine angemessene Benutzerfreundlichkeit ist ein wichtiger Punkt dieser Arbeit. Eine virtuelle Umgebung ist ohne Benutzerhandbuch teilweise schwer zu durchblicken und daher nicht wirklich benutzerfreundlich. Um dem Benutzer dabei etwas an die Hand zu geben ist die Erstellung eines *Wikis*. Dieses soll als Benutzerhandbuch dienen und die wichtigsten Bereiche und Hinweise zur Erstellung und Verwendung der virtuellen Umgebung abdecken. Zu jeder Funktionalität sollen hier Begründungen und Hinweise zu finden sein. Außerdem werden an manchen Punkten Alternativen oder Verbesserungsmöglichkeiten aufgezeigt werden.

6 Fazit

6.1 Zusammenfassung der Ergebnisse

7 Literatur- und Quellenverzeichnis

Literatur

Brikman, Yevgeniy (2019). *Terraform: Up Running: Writing Infrastructure as Code*. Ö'Reilly Media, Inc."

HashiCorp (2022). *Providers*. URL: <https://registry.terraform.io/browse/providers> (besucht am 07.09.2022).

Wang, Kevin (14. Apr. 2022). *What is Terraform?* URL: <https://www.terraform.io/intro> (besucht am 07.09.2022).

aufteilen in internetquellen publications books

8 Anhang

A Anhang A

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Berlin, den 25. September 2022

Patrick Lucas Büdke