INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Evaluierung der Dienst- und Versionserkennung von Linux-Diensten mit Open-Source und kommerziellen Netzwerkscanner

Dmitriy Elin

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Evaluierung der Dienst- und Versionserkennung von Linux-Diensten mit Open-Source und kommerziellen Netzwerkscanner

Dmitriy Elin

Aufgabensteller:	Prof. Dr. Helmut Reiser
Betreuer:	Tobias Appel
Abgabetermin:	15. Juli 2019

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

München, den 15. Juli 2019

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Das Wissen über das Betriebssystem und die Arten von Diensten, die auf einem Remote-Server ausgeführt werden, sind wertvolle Informationen für den Penetration-Tester und Angreifer. Diese Information lässt erkennen, welche Sicherheitslücken vorhanden sein könnten. In dieser Arbeit wird ein Überblick über die Methode für die aktive Port-Scanning gegeben. Es wird die bereits auf dem Markt vorhandenen Softwareprodukte für die Diensterkennung dargestellt und mit freie Open-Source Werkzeug NMAP verglichen.

Im Rahmen einer Evaluation müssen eine Linux-Umgebung mit bekannten Diensten entwürfen, um die Software anhand der Genauigkeit der Erkennung untersucht zu können. Außerdem, wird es ein Versuch untergenommen, die Werkzeuge durch die Änderung der Portnummer und der Antwort-Nachrichten umgehen.

Schließlich werden dann die Ergebnisse mehrerer Testläufe miteinander verglichen.

Inhaltsverzeichnis

1 Einführung

Das Leibniz-Rechenzentrum (LRZ) betreibt für die Münchner Hochschulen und andere wissenschaftsnahe Einrichtungen das Münchner Wissenschaftsnetz (MWN), in dem sich über 200.000 Endgeräte befinden. Solche komplexen dezentralen Systeme werden oft nicht zentral verwaltet. Die unkoordinierte Entwicklung führt dazu, dass die Information über die Hardware- und Softwarekonfiguration meist lückenhaft und unvollständig ist. Es kann der Fall sein, dass einige Endgeräte veraltete Softwarekomponenten einsetzen und somit potentiell kompromittierbar bzw. ausnutzbar für Schadsoftware sind. Die erhöhte Komplexität führt zu einer neuen Herausforderung für den Betreiber eines solches Netzes, und zwar den Überblick über die Software zu behalten.

Das Wissen über das Betriebssystem und die Arten von Diensten, die auf einem Remote-Server ausgeführt werden, sind wertvolle Informationen für den Penetration-Tester und Angreifer. Diese Information lässt erkennen, welche Sicherheitslücken vorhanden sein könnten. Unter Fingerprint versteht man den Prozess eines Beobachters oder Angreifers, der ein Gerät oder eine Anwendungsinstanz eindeutig identifiziert (mit ausreichend hoher Wahrscheinlichkeit), basierend auf mehreren Informationselementen, die dem Beobachter oder Angreifer mitgeteilt werden [Co13]. Im Folgenden wird mit dem Begriff Fingerprint die Betriebssystemerkennung (OS Fingerprint) bezeichnet. Ein ähnlicher Prozess, bei dem die laufenden Dienste ermittelt werden, wird als Diensterkennung bezeichnet.

Sowohl die Methoden vom OS Fingerprinting, als auch die von der Diensterkennung sind gut erlernt und werden schon seit langer Zeit in Schwachstellenscannern und Netzwerk-Dienstprogramm verwendet. Bei diesem Verfahren (auch als aktive Port-Scanning Technik bekannt) werden Datenpakete an einen zu wählenden Ports-Bereich von untersuchenden Rechner geschickt und seine Reaktionen ausgewertet, so dass durch einen Abgleich mit einer Datenbank auf die verwendeten Dienste geschlossen werden kann. Die Aktive Erkennung findet die verwendeten Dienste, ohne sich auf die Klientaktivität zu verlassen. In diesem Fall fehlen aber die Dienste, die zum Zeitpunkt der Prüfung nicht verfügbar waren, und solche, die die Prüfungen aktiv blockieren. Ein anderes Verfahren ist das passive Port-Scanning. Im Gegensatz zu aktiven Verfahren, werden nur die Daten benutzt, die im Kommunikationspfad gemessen werden können, z. B. an einem Router oder Switch. Passive Discovery findet schnell sehr beliebte Dienste, auch wenn diese Dienste durch Firewalls geschützt sind.

Die meistverbreiteten Open-Source Netzwerkscanner nmap wurden vor der Jahrtausendwende entwickelt und werden bis heute stetig aktualisiert. Nmap („Network Mapper“) ist ein Werkzeug für die aktive Netzwerkanalyse und Sicherheitsüberprüfung. Es wurde entworfen, um große Netzwerke schnell zu scannen, auch wenn es bei einzelnen Hosts gut funktioniert. Nmap benutzt rohe IP-Pakete um festzustellen, welche Hosts im Netzwerk verfügbar sind, welche Dienste (Anwendungsname und -version) diese Hosts bieten, welche Betriebssysteme (und Versionen davon) darauf laufen, welche Art von Paketfiltern bzw. Firewalls benutzt werden sowie andere Eigenschaften. Auch wenn Nmap üblicherweise für Sicherheitsüberprüfungen verwendet wird, wird es von vielen Systemen und Netzwerkadministratoren für Routineaufgaben benutzt, z.B. für die Netzwerkinventarisierung, die Verwaltung von Ab-

laufplänen, für Dienstaktualisierungen und die Überwachung von Betriebszeiten von Hosts oder Diensten.

Es gibt sowohl zahlreiche weitere Open-Source Tools, als auch kommerzielle Produkte, die Betriebssystem- und Diensterkennung anbieten.

1.1 Zielsetzung

Beim Betreiben einer komplexen IT-Infrastruktur müssen Endgerate regelmäßig auf Schwachstellen überwacht werden. Um veraltete Softwarekomponenten von Endgeräten zu identifizieren, ist es sinnvoll das Netz auf erreichbare Dienste mit einem Port-Scanner zu scannen und die gefundenen Ergebnisse mit einer CVE-Datenbank abzugleichen. Ziel dieser Arbeit ist herauszufinden, wie gut die Diensterkennung von NMAP im Vergleich zu anderen Open-Source Schwachstellenscannern und kommerziellen Schwachstellenscannern funktioniert.

1.2 Vorgehensweise

Nachfolgend wird kurz das Vorgehen zur Umsetzung dieser Arbeit beschrieben. Im Rahmen dieser Bachelorarbeit wurde eine Evaluierungsumgebung für Dienst- und Versionserkennung in einem einfachen lokalen Netz entwickelt, mit der man in der Lage ist, mehrere verschiedene Anwendungen zu vergleichen. Dabei ist die Suche nach aktiven Hosts im gegebenen Netzwerk nicht Teil dieser Arbeit.

1.3 Aufbau der Arbeit

Kapitel 1 gibt eine kurze Einleitung in das Thema und zeigt die Vorgehensweise zur Umsetzung der Arbeit auf. Im Kapitel 2 wird zuerst eine Literaturrecherche durchgeführt und danach werden die Grundlagen behandelt, die für das Verständnis der Arbeit notwendig sind. Der Aufbau und die besonderen Eigenschaften von Netzwerkprotokollen, Scantechniken und Diensterkennung eingangs erzählt. Im Kapitel 3 werden Programme im Vergleich aufgezeigt. Im nächsten Schritt wird Versuchsaufbau definiert. Der Fokus vom Kapitel 5 und 6 liegt auf Aufbau von Evaluierungsumgebung und die praktische Durchführung von Tests. Des Weiteren werden dann im Abschnitt 7 die Ergebnisse erläutert und nach bestimmten Kriterien bewertet. Das Ende bildet eine Zusammenfassung und ein Ausblick in Kapitel 8. Im Anhang dieser Arbeit sind die Dienstverfügbarkeit-Protokolle (Anhang A) und die Konfigurationsdateien für die Testdienste (Anhang B).

2 Grundlagen

2.1 Netzwerkgrundlagen, Schichtenmodelle und Protokolle

Die verschiedenen Aspekte einer Kommunikation werden in mehrere Schichten zerlegt und in einem Schichtenmodell dargestellt. Protokolle innerhalb einer Schicht erfüllen ähnliche Aufgaben und sind gegen Protokolle der gleichen Schicht austauschbar. Dieses Prinzip ermöglichte die Entwicklung des Internets, eines weltweiten Netzwerks verschiedenster Computersysteme mit verschiedensten Ubertragungsmedien. Als Schichtenmodell kommt dabei das TCP/IP-Referenzmodell zum Einsatz, welches aus den folgenden Schichten besteht:

Schicht 5 – Anwendungsschicht Die Anwendungsschicht enthält jene Netzwerkprotokolle, die von Anwendungen benutzt werden, mit denen der Endnutzer direkt in Berührung kommt, ob Browser, EMail-Kommunikation oder Dateitransfer.

Schicht 4 – Transportschicht Die Transportschicht erstellt eine Ende-zu-Ende-Verbindung zwischen beiden Hosts und abstrahiert auf diese Weise von den darunter liegenden Schichten. Weiterhin adressiert die Transportschicht den jeweiligen Prozess auf Quell- und Zielhost über die Portnummer. Je nach Implementierung enthält die Transportschicht auch Flusskontroll Mechanismen und Verfahren zur Segmentierung von Datenpaketen sowie Fehlerbehebungsverfahren, die den Erhalt sowie die korrekte Reihenfolge empfangener Daten sicherstellen.

Schicht 3 – Internetschicht Die Vermittlungsschicht dient der Navigation durch den Netzgraphen, für die Kommunikation zwischen Computern, die sich in verschiedenen Netzen befinden. Sind die zu versendenden Pakete zu groß für die Sicherungsschicht, müssen sie von der Internetschicht fragmentiert werden.

Schicht 2 – Netzzugangsschicht Die Netzzugangsschicht dient der Adressierung im lokalen Netzwerk, der Sicherstellung der Integrität übermittelter Daten sowie der Flusskontrolle, um Überlastungen des Netzmediums zu vermeiden. Außerdem spezifiziert die Netzzugangsschicht die physikalische Übertragung von Signalen zwischen zwei direkt verbundenen Computern über das verwendete Netzmedium.[Tan92]

2.1.1 Das Internet-Protokoll

Das Internet-Protokoll befindet sich im TCP/IP-Modell auf der Internetschicht. Es dient primär der Adressierung von Zielsystemen in lokalen TCP/IP-Netzen und im Internet. Für das Fingerprinting von Betriebssystemen spielt der Aufbau des IP-Headers eine besondere Rolle. Anhand verschiedener Implementierungen des TCP/IP-Stacks können Betriebssysteme voneinander unterschieden werden. Daher erfolgt ein Überblick über die Felder des IP-Headers:

2.1.2 Das Transmission Control Protocol

Das Transmission Control Protocol (TCP) ist unzweifelhaft einer der bedeutsamsten Erfindungen der Informationstechnik des 20. Jahrhunderts. TCP schafft es, auf einem fehler-

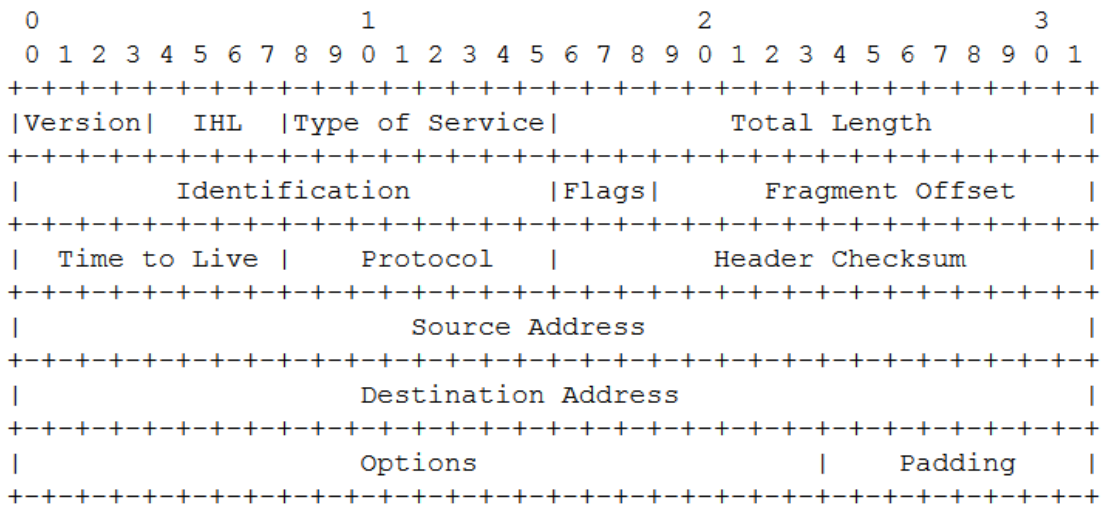

Abbildung 2.1: IPv4-Header

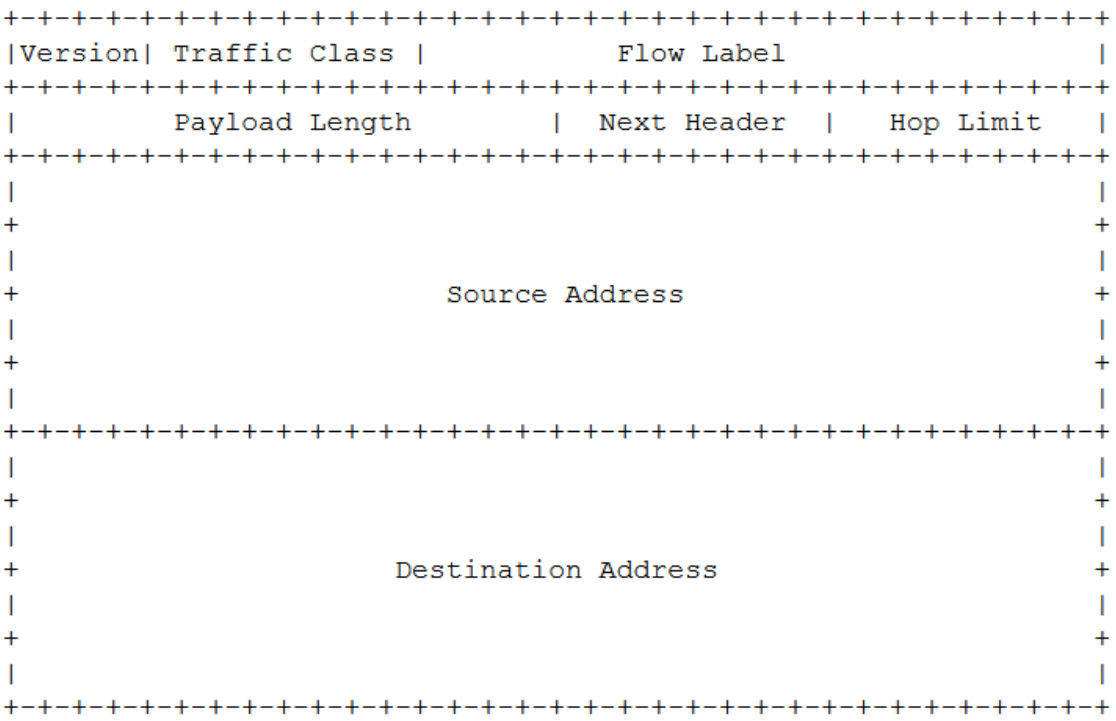

Abbildung 2.2: IPv6-Header

behafteten Medium auf Basis von unzuverlässigen Netzwerk-Protokollen eine zuverlässige Kommunikations-Verbindung zu etablieren, in der Nachrichten stets in der richtigen Reihenfolge ankommen und bei Störungen in den unteren Netzwerk-Schichten neu übertragen werden 2.3.

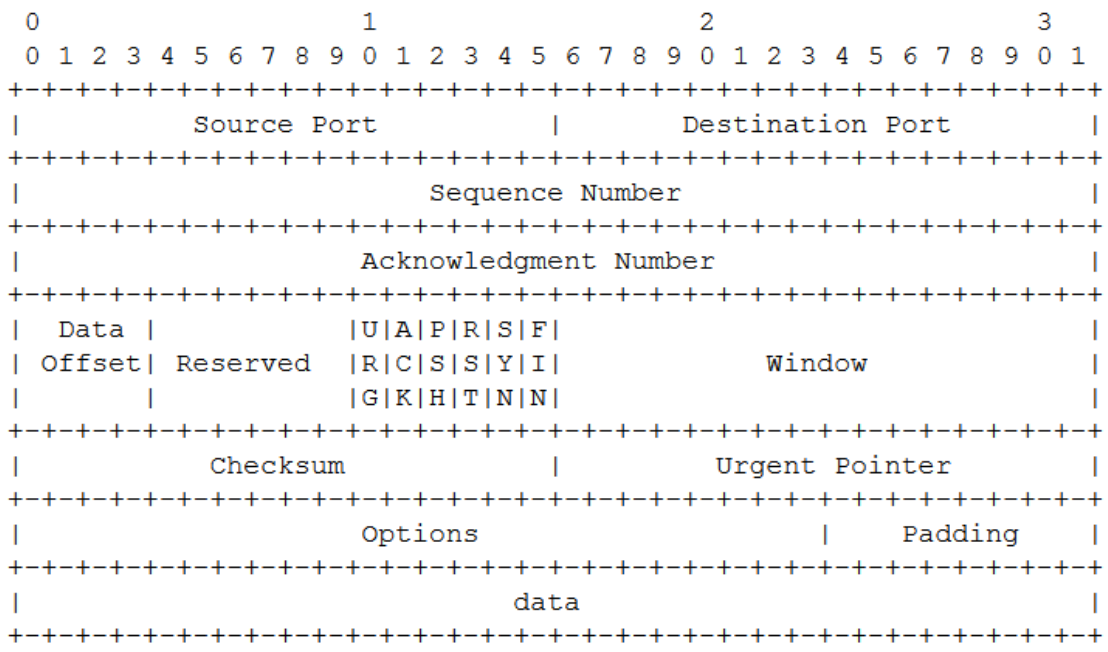

Abbildung 2.3: TCP-Header

2.2 OS Fingeprinting

Unter OS Fingerprint versteht man ein Prozess zur Erkennung vom Betriebssystem auf einem Rechner im Netzwerk. Dieser Prozess erfordert eine Menge von Tests und Inspektion der RFC-Richtlinien um die Punkte zu erkennen, die von verschiedenen Anbietern unterschiedlich interpretiert werden könnte. Beim OS Fingerprinting werden TCP/IP-Pakete mit bestimmten Flags an ein Ziel geschickt und die Antworten analysiert, so dass durch einen Abgleich mit einer Datenbank auf das verwendete Betriebssystem geschlossen werden kann. Die Datenbank enthält die Reaktionen von verschiedenen Betriebssystemen auf diese TCP/IP-Pakete, die versuchsweise ermittelt wurden. OS Fingerprint nutzt den TCP/IP-Stack aus, die charakteristische TCP/IP Implementierung, die sich von Betriebssystem zu Betriebssystem unterscheidet [PC04]. Anhand der Reaktion eines Systems auf eine Reihe von Anfragen kann ein „Fingerabdruck" erstellt werden. Die beim Fingerprint eingesetzten Anfragetypen lassen sich nach den benutzten Pakettypen gliedern[Fyo99]. Die Anwendbarkeit von diesen Methoden mit IPv6 Protokoll wurden bereits in der Arbeit [Ner06] untersucht.

2.2.1 IP-Anfragen

Bei IP-Anfragen werden Felder des IP-Headers und spezielle Eigenschaften der jeweiligen Implementierung im Bezug auf Fragmentierung untersucht.

- IPID Sampling. Hier wird das IP Identification Feld analysiert. Das Identification Feld im IP-Header enthält eine Nummer, die hilft, fragmentierte Teile eines Datagramms korrekt zu reassemblieren. Die meisten Betriebssysteme erhöhen die IP ID mit jedem versendeten Paket um 1, während andere die ID zufällig wählen oder nur in festen Schritten größer als 1 erhöhen.
- Don't Fragment Bit. Das Don't Fragment Bit im IP-Header verbietet das Fragmentieren des Datagramms. Einige Systeme setzen dieses Bit in bestimmten Fällen, während andere es gar nicht setzen, so dass das jeweilige Verhalten helfen kann, die Implementierung zu identifizieren.
- Fragmentation Handling. Diese Technik untersucht die Reassemblierung von fragmentierten IP-Paketen. Im speziellen wird hier die Reassemblierung von überlappenden Fragmenten beobachtet. Je nach Implementierung besitzt der ältere oder der neuere überlappende Teil Gültigkeit.

2.2.2 ICMP-Anfragen

ICMP-Anfragen sind Anfragen mittels des ICMP-Protokolls. Das ICMP-Protokoll wird zum Austausch von Fehler- und Kontrollnachrichten eingesetzt.

- ICMP Error Message Quenching. Einige Betriebssysteme limitieren die Senderate für Fehlerbenachrichtigungen. Getestet wird dies, indem mehrere Pakete an einen zufälligen hohen UDP Port gesendet werden, der geschlossen ist. Dies provoziert ICMP Destination Unreachable Fehlerbenachrichtigungen. Anhand der Eingangsrate der Fehlerbenachrichtigungen des gescannten Systems kann die Implementierung dann identifiziert werden.
- ICMP Message Quoting. Bei einer ICMP Fehlerbenachrichtigung wird ein Teil des fehlerverursachenden Pakets wieder zurückgesandt. Die Implementierungen unterscheiden sich hier in der Menge der mitgesendeten Daten. Fast alle Systeme senden den IP-Header und acht Bytes zurück, Solaris- und Linux-Betriebssysteme senden ein Bit bzw. noch mehr Daten mit zurück.
- ICMP Error Message Echoing Integrity. Diese Methode funktioniert wie ICMP Message Quoting, auch hier werden die ICMP Fehlerbenachrichtigungen untersucht. Hier werden speziell das Headerfeld und die Prüfsumme analysiert.
- Type of Service. Das Type of Service Feld in der ICMP Fehlerbenachrichtigung ICMP Port Unreachable ist standardmäßig auf 0 gesetzt. Manche Implementierungen weichen in diesem Punkt jedoch ab.

2.2.3 TCP-Anfragen

Bei diesen Anfragen werden TCP-Pakete eingesetzt. Die Pakete enthalten verschiedene gesetzte Flags und die Reaktionen des Zielsystems auf diese Flags werden analysiert.

- FIN Probe. Es wird ein Paket mit gesetztem FIN-Flag an einen offenen Port gesendet. Das korrekte in RFC 793 spezifizierte Verhalten wäre es, nicht zu antworten, aber viele Implementierungen senden ein RST zurück. FIN Probe ist ähnlich zum TCP FIN Scan, der jedoch davon ausgeht, dass nur geschlossene Ports ein RST zurücksenden. Die FIN Probe liefert also nur Informationen über eine Implementierung, wenn man aufgrund eines anderen Scans mit Sicherheit sagen kann, dass der angefragte Port offen ist.

- TCP ISN Sampling. Hier wird versucht die Initial Sequence Number der TCP Implementierung beim Verbindungsrequest festzustellen. Je nach Betriebssystem werden hierbei unterschiedliche Verfahren eingesetzt. Ältere Unix Systeme erhöhen in 64k Schritten, neuere erhöhen zufällig, Linux Systeme wählen zufällige Nummern und Microsoft benutzt ein zeitabhängige Verfahren, bei dem die ISN in jeder Zeiteinheit um einen festen Betrag erhöht wird.

- ACK Value. Die ACK Flag wird jeweils als Antwort auf ein FIN/URG/PSH gesetzt und das Acknowledgement Number Feld enthält je nach Implementierung eine andere Nummer. Die meisten Systeme setzen als Antwort auf eine FIN/PSH/URG Anfrage die Initial Sequence Number (ISN) des eingegangen Pakets als Acknowledgement Number, manche Implementierungen erhöhen die ISN um 1.

- TCP Options. Hier werden in den versendeten Paketen verschiedene TCP Optionen, die in RFC 793 und RFC 1323 definiert werden, eingesetzt. Je nachdem ob diese auch in der Antwort enthalten sind, kann man erkennen, welche Optionen unterstützt werden und somit auf die Implementierung schließen, da die Implementation ist freiwillig. In einem Paket können mehrere Optionen (Timestamp, Windows Scale, NOP, Max Segment Size, End of Ops) gesetzt werden und somit gleichzeitig getestet werden. TCP Initial Window TCP Initial Window überprüft die gesetzte Fenstergröße bei zurückgesendeten Paketen. Die Fenstergröße ist teilweise eindeutig einer Implementierung zuzuordnen.

2.3 Diensterkennung

Die Diensterkennung ist der Ansatz des Application-Mappings, Spekulationen über das Protokoll auf TCP/IP Anwendungsschicht hinter einem offenen Port zu betreiben.

Die Vergabe der Portnummern an Anwendungsprozesse geschieht dynamisch und wahlfrei. Für bestimmte, häufig benutzte Anwendungsprozesse sind feste Portnummern vergeben. Diese werden als Assigned Numbers bezeichnet. Die IANA [1] hat die Portnummern in Gruppen unterteilt und diese für bestimmte Dienste reserviert. Die Portnummern 0 bis 1.023 sind für bekannte Dienste reserviert, die als Well Known Services bezeichnet werden. Die darüber liegenden Portnummern von 1.024 bis 49.151 sind für registrierte Ports reserviert und die darüber liegenden Ports bis zu Portnummer 65.535 werden als dynamische oder Private Ports bezeichnet.

Vor allem aber lässt sich auf diese Weise nur das zugrunde liegende Anwendungs-Protokoll erkennen, nicht jedoch ein konkreter Dienst oder gar ein bestimmtes Release dieses Dienstes. Dies sollte jedoch das Ziel eines erfolgreichen Fingerprintings sein, um den passenden Exploit für den jeweiligen Dienst auswählen zu können. Schließlich ist dieser Ansatz untauglich, wenn

[1] Die Internet Assigned Numbers Authority (IANA) ist eine Gesellschaft, die für die Zuordnung von Nummern und Namen im Internet, insbesondere von IP-Adressen und Ports, zuständig

sich hinter den geöffneten Ports nicht bei der IANA registrierte Dienste und insbesondere Individualsoftware-Produkte verbergen.

Ziele des Service-Fingerprintings sind:

1. Das zugrunde liegende Anwendungsprotokoll zu ermitteln
2. Die verwendete Server-Software herauszufinden
3. Das genaue Release zu identifizieren

Dafür müssen wie bei der TCP/IP-Stack-Analyse entsprechende Testfälle erstellt, über das Netzwerk versandt und die Antworten-Nachricht gesammelt und analysiert wird. Service-Fingerprinting-Software wird meist unter der Annahme entwickelt, dass das Anwendungsprotokoll bereits bekannt ist. Da jedes Protokoll der Anwendungsschicht ein anderes Vorgehen erfordert, kann es keine universelle Service-Fingerprinting-Software geben, es sei denn, als Zusammenstellung verschiedener Fingerprinting-Ansätze. Mit der Kenntnis des Anwendungsprotokolls können speziell präparierte Nachrichten an das Zielsystem und den Zielport gesendet werden. Diese Nachrichten zeichnen sich dadurch aus, dass sie das jeweilige Anwendungsprotokoll grundsätzlich einhalten, jedoch gewisse (dem Protokoll konforme oder vom Protokoll abweichende) Änderungen vornehmen, die zu einem unterschiedlichen Antwort-Verhalten der Zielanwendungen führen. Aufgrund dieser Unterschiede im Antwort-Verhalten können daraufhin Fingerprints der jeweiligen Dienste erstellt werden, die zur späteren Identifikation des gleichen oder eines ähnlichen Dienstes dienen [Han08].

2.4 Themenverwandte Arbeiten

In der vorliegenden Arbeit wird die Erkennung von Diensten mittels aktives Scan durchgeführt um die Ergebnisse von verschiedenen Tools zu vergleichen. Sowohl die Technik, als auch das Software für aktives Scan wurde von einigen vorherigen Arbeiten bereits untersucht. In Folgenden werden solche Arbeiten kurz vorgestellt und deren Abweichung von dieser Arbeit erklärt.

Die Arbeit “Netzbasierte Erkennung von Systemen und Diensten zur Verbesserung der IT-Sicherheit” [Ber14] liegt der Schwerpunkt auf der passiven Erkennung von Systemen und Diensten. Es wurde sowohl aktive Erkennungsmethoden von Port- und Schwachstellenscanner untersucht als auch passive Erkennungsmethoden wie die Netflow-Technik oder die DPI betrachtet und bewertet. Ausgehend von diesen Erkenntnissen wurde ein Prototyp erstellt, der eines der vorgestellten Verfahren, die Netflow-Technik, zur Erreichung der erstellten Kategorisierung von Hostsystemen implementiert.

In Analysing Port Scanning Tools and Security Techniques [KS14] werden einige Scantools dargestellt, die verwendete Sicherheitstechniken analysiert und beschrieben, wie sie offene und geschlossene mit häufig vorkommenden Porten im Computersystem finden. In der Arbeit wurde solche Dienste wie FTP, SSH, Telnet, DNS, HTTP, HTTPS untersucht. Es fehlt aber eindeutige Kriterien um die vorgestellte Scan-Software zu vergleichen.

In der Arbeit „Evaluation of Network Port Scanning Tools” [END11] wurde versucht 8 Werkzeuge für Diensterkennung: Nmap, SuperScan, Advanced Port Scanner, Advanced Administratiev Tools, Angry IP Scanner, Atelier Web Security Port Scanner, Unicornscan, GFILANguard auf der Basis von mehreren Kriterien zu vergleichen 2.4, die wichtigste davon:

- TCP/UDP Scan: Fähigkeit des Werkzeugs TCP und UDP ports zu scannen;

- Banner Grabbing: das Aufzeichnen und Auswerten der Inhalte von Willkommensnachrichten, die Dienste als Antwort für die Anfrage senden;
- Port list DB: Verfügbarkeit der Datenbank mit Beschreibungen von Diensten, die der Portnummer zugeordnet sind;
- MAC Address: Ermittlung der MAC-Adresse;
- Query Application Protocols: Anwendungsprotokolle Abfragen: Ob das tool in der Lage ist, nach allen Arten von Anwendungsprotokollen wie Webservern, Datenbanken, DNS-Servern, FTP-und Gopher-Servern zu suchen;
- UN/PW Recovery: Fähigkeit, Benutzername (UN) und Passwort (PW) mit brute force wiederherzustellen.

	TCPSYN	UDP	Banner Grabbing	Port List DB	Active Port Mapping	MAC Address	Query Application	UN/PW Recovery
NMAP	+	+	+	+	+	+	+	+
SuperScan	+	+	+	+	-	-	-	+
Advanced Port	+	+	-	-	-	-	-	+
AATools	+	+	+	+	+	-	-	-
AngryIP	+	+	-	-	-	-	-	+
AWSP	+	+	-	+	+	+	-	-
Unicornscan	+	+	+	+	-	-	-	+
GFILANguard	+	+	-	+	+	+	-	-

Abbildung 2.4: Gegenüberstellung von Diensterkennung bei verschiedenen Werkzeugen.

In der Unterschied zu dieser Arbeit konzentriert sich der Werk nicht auf OS- und Diensterkennung. Die Gegenüberstellung basiert sich auf unterstützende aktive Scanmethoden. Außerdem, hat die Autoren keine Rucksicht auf die kommerziellen Schwachstellescannern genommen.

Eine Probe für den Vergleichstest von Schwachstellescannern (Nessus, OpenVAS, Nexpose und NMAP) wurde in [TAR12] [Jha14] ausgeführt. Die wichtigsten Ergebnisse (vgl. Abbildung 2.5) und Punkte der Testdurchführung:

- OpenVAS v5 wurde mit dem vollständigen und schnellen Scan-Profil getestet (Ports waren alle TCP-Ports und Top-100-UDP-Ports).
- Nessus v5 wurde unter Verwendung des externen Netzwerk-Scan-Vorgangs eingesetzt und zusätzlich mit einem internen Netzwerk-Scan getestet, die Ergebnisse waren aber gleich.
- Der Nexpose-Scanner wurde mit dem vollständigen Testprofil ausgeführt.
- Es wurden keine Anpassungen der Standard-Scan-Profile vorgenommen.
- Die Tests wurden auf einen externen Netzwerkdienst angepasst, sodass keine Anmeldeinformationen verwendet wurden.

- Diese Geräte wurden anhand eines Beispielsatzes ausnutzbarer und falsch konfigurierter Dienste im Metasploitable-Framework überprüft.

Der Fokus dieser Arbeit liegt an der Schwachstellenerkennung, wobei es keine Rücksicht auf das Potenzial der Diensterkennung genommen wurde, insbesondere der Einfluss der Genauigkeit auf die Endergebnisse.

Security Issue	Nessus	OpenVAS	Nexpose	Nmap
FTP 21 Anonymous FTP Access	✓	✓	✓	✓
FTP 21 VsFTPd Smiley Face Backdoor	✓	✓		
FTP 2121 ProFTPD Vulnerabilities		✓		
SSH 22 Weak Host Keys	✓	🔍	✓	
PHP-CGI Query String Parameter Injection	✓	✓	✓	✓
CIFS Null Sessions	✓	✓	✓	✓
INGRESLOCK 1524 known backdoor drops to root shell				
NFS 2049 /* exported and writable	🔍	✓	🔍	
MYSQL 3306 weak auth (root with no password)	✓	✓	✓	✓
RMI REGISTRY 1099 Insecure Default Config				
DISTCCd 3632 distributed compiler				
POSTGRESQL 5432 weak auth (postgresql)				
VNC 5900 weak auth (password)			✓	
IRC 6667 Unreal IRCd Backdoor				✓
Tomcat 8180 weak auth (tomcat/tomcat)	✓		✓	✓

Abbildung 2.5: Gegenüberstellung der Schwachstelleerkennung.

3 Produktauswahl

3.1 NMAP

Nmap oder auch „Network Mapper" ist ein Portscanner, der seit 1997 vom „Nmap-Developer-Team" entwickelt wird. Mit Nmap lassen sich Netzwerke oder Computer im Internet auf offene Ports und den darauf lauschenden Diensten prüfen. Nmap kann z.B. zum Testen der Firewall-Konfiguration eingesetzt werden oder auch zum Testen des Computers auf offene Ports und (eventuell unerwünschte) im Hintergrund laufende Dienste.

Ein besonderer Vorteil von Nmap ist, dass es sich hierbei um einen Open-Source Sicherheitsscanner handelt, der für die Detektion von sich im Netz befindlichen Systemen und dort betriebener Dienste konzipiert wurde. Hierfür bietet Nmap die Möglichkeiten Hosts in einem Netz zu erkennen, Ports zu scannen, Software und Versionen sowie Betriebssysteme zu detektieren. In den Standardeinstellungen untersucht Nmap alle Ports eines Hosts, indem es TCP SYN-Pakete, den sogenannten TCP-SYN-Scan, an den untersuchten Port sendet. Falls dieser Port geöffnet ist, wird mit einem TCP SYN+ACK geantwortet, andernfalls mit einem TCP RST. Durch diese Untersuchungstechnik lässt sich schnell herausfinden, welche Ports an einem Host geöffnet sind. Um weitere Informationen über den untersuchten Host zu generieren, versucht Nmap, den genutzten Service sowie dessen Version durch Senden weiterer Prüfpakete herauszufinden.

So wird hierbei unter anderem eine Verbindung aufgenommen und auf die Antwort des Dienstes für eine gewisse Zeit gewartet, da viele Dienste sich selbst mit einer Willkommensnachricht identifizieren. Die empfangenen Daten werden anschließend mit einer Signaturdatenbank verglichen [AW16].

Nmap verfügt über 2200 bekannte Dienste in seiner nmap-services Datenbank. Nachdem TCP- und/oder UDP-Ports mit Methoden entdeckt wurden, fragt die Versionserkennung diese Ports ab, um mehr darüber zu erfahren, was tatsächlich darauf läuft. Die Datenbank in nmap-service-probes enthält Testpakete für die Abfrage verschiedenster Dienste und Ausdrücke für den Vergleich und das Parsen der Antworten. Nmap versucht, das Dienstprotokoll zu bestimmen (z.B. FTP, SSH, Telnet, HTTP), aber auch Anwendungsnamen (z.B. ISC BIND, Apache httpd, Solaris telnetd), Versionsnummer, Hostnamen, Gerätetyp (z.B. Drucker, Router), die Betriebssystemfamilie (z.B. Windows, Linux) und manchmal verschiedene Details: etwa ob ein X-Server Verbindungen annimmt, die SSH-Protokollversion. Natürlich bieten die meisten Dienste nicht all diese Information. Falls Nmap mit OpenSSL-Unterstützung kompiliert wurde, verbindet es sich mit SSL-Servern, um den hinter dieser Verschlüsselungsebene lauschenden Dienst zu ermitteln. Wenn RPC-Dienste erkannt werden, wird automatisch die RPC-Programm- und Versionsnummern ermitteln. Manche UDP-Ports bleiben im Zustand offen/gefiltert, nachdem ein UDP-Port-Scan nicht bestimmen konnte, ob der Port offen oder gefiltert ist. Die Versionserkennung versucht, diesen Ports eine Antwort zu entlocken (genauso wie bei offenen Ports) und den Zustand auf offen zu ändern, wenn das gelingt. Offene/gefilterte TCP-Ports werden genauso behandelt[NRH18].

3.2 OpenVAS

Das Open Vulnerability Assessment System (OpenVAS) ist eine Kombination aus mehreren Diensten und Werkzeugen 3.1, die zusammen eine umfangreiche und mächtige Lösung für Schwachstellen-Scanning und Schwachstellen-Management darstellen. Bei Sicherheitsun-

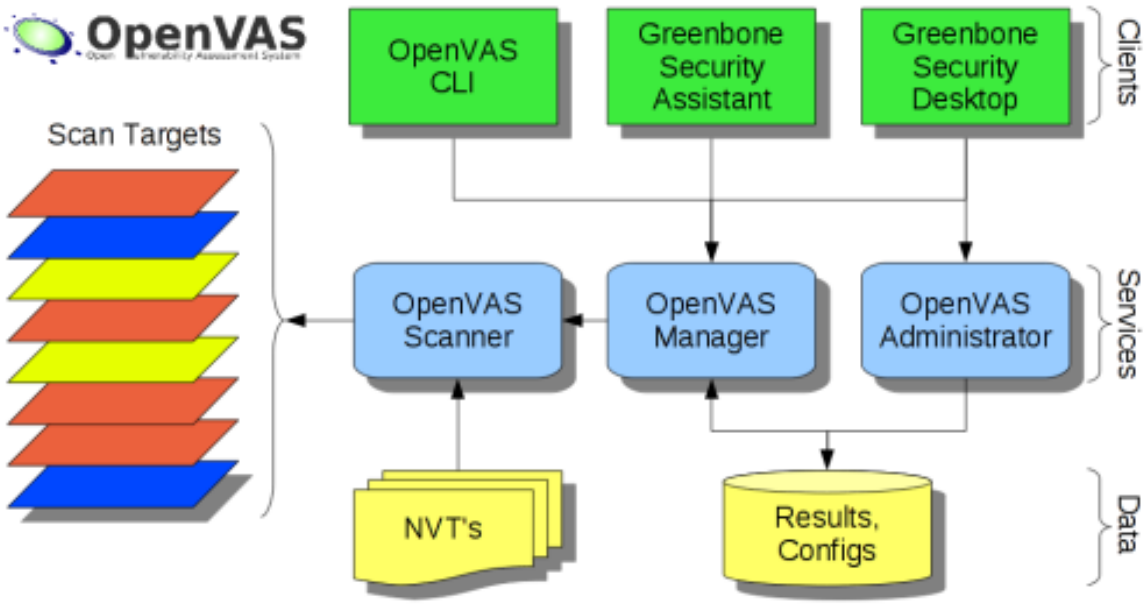

Abbildung 3.1: Die OpenVAS-Architektur.

tersuchungen der in einem Netzwerk vorhandenen IP-basierten Systeme mittels OpenVAS können verschiedene Prüftiefen eingestellt werden. OpenVAS ist ein Server-basiertes Sicherheitswerkzeug, welches auch über ein Command Line Interface (CLI) gesteuert werden kann. Es wird auf einem Linux basierten Rechner installiert und ist über die Web-Oberfläche Greenbone Security Assistant (GSA) von einem beliebigen Arbeitsplatz aus komfortabel bedienbar. Das Programm Greenbone Security Desktop (GSD) zur Steuerung von OpenVAS ist für Linux und Windows als Freie Software verfügbar, genauso wie alle anderen Komponenten von OpenVAS. Ergänzende Sicherheitssoftware kann für lokale Prüfungen oder aufbereitete Berichtauswertungen eingebunden werden, um so Schwachstellen auf den Zielsystemen genauer zu identifizieren oder zu analysieren. Es können übersichtlich Prüfberichte erstellt und Alarme für Sicherheitsvorkommnisse ausgelöst werden.

Durch Einordnung der gefundenen Probleme in Sicherheitsklassen können Sicherheitslücken einfach priorisiert und Sicherheitsmeldungen zugeordnet werden. Umfangreiche Möglichkeiten zur Bearbeitung von Prüfberichten sowie deren Vergleich oder Export erlauben zügig einen Netzwerk-weiten Überblick über die Trends der IT-Sicherheit zu gewinnen, um so Sicherheitslücken aufzuspüren und zu schließen, bevor Angreifer diese Lücken missbräuchlich nutzen können.

Der OpenVAS Manager ist der zentrale Dienst, der aus den Scan-Resultaten ein weitreichendes und mächtiges Schwachstellen-Management System macht. Er kontrolliert und steuert einen oder mehrere Scans, aber auch andere Manager in einem sog. Master-Slave Modus. Ebenso kontrolliert der Manager die interne zentrale SQL-Datenbank, in der alle

Scan-Ergebnisse und Konfigurationen gespeichert werden. Verschiedene Client-Programme können den Manager über das XML-basierte, zustandslose OpenVAS Management Protocol (OMP) nutzen. Intelligente Sortier- und Filter-Logiken sind ebenfalls innerhalb des Managers implementiert. Dadurch erhält der Anwender über verschiedene Client-Programme immer eine gleichbleibende Sicht auf die Ergebnisse.

3.3 Teanable Nessus

Nessus ist ein proprietärer Schwachstellenscanner, der von Tenable Network Security entwickelt wurde. Nessus bietet dem Anwender einige Features, die bei herkömmlichen Scannern nicht zu finden sind. Die Wichtigsten davon sind: - Plug-in Architektur Nessus bedient sich beim Testen eines Systems einer Plug-in-Datenbank, in der die eigentlichen Tests abgelegt sind. So hat der Anwender die Möglichkeit den Vulnerability-Scanner seinen Gegebenheiten nach exakt anzupassen, damit der nur das testet, was der Anwender auch will. - NASL (Nessus Attack Scripting Language) Die Nessus Attack Scripting Language wird von dem Nessus-Projekt jedem programmier-freudigen Anwender zur Verfügung gestellt. Damit kann jeder Benutzer seine eigenen Tests, auf einfache Art und Weise, entwickeln. - Up-to-date Security Vulnerability Datenbank Ähnlich zu anderen Vulnerability - Scannern gliedert sich Nessus in einen Benutzer-Client, über den der Anwender seinen Scanlauf konfigurieren kann, und einen Server (nessusd), der die eigentlichen Tests auf dem entfernten Host durchführt. Diese Architektur erlaubt es, Server und Client auf unterschiedlichen Systemen zu installieren. Damit ist man in der Lage, ein ganzes Netzwerk von einem einzelnen Rechner aus zu testen. Die Clients stehen für unterschiedliche Plattformen zur Verfügung: X11, Win32 und ein Plattform - übergreifender Client geschrieben in der Programmiersprache Java. Vorteile:

- Client - Server – Architektur
- Zeitgleiches Testen mehrerer Hosts
- Multiples Services
- Tests Kooperation
- Komplette und exportierbare Report – Dateien

Die Ergebnisse eines Scanlaufs werden dem Anwender in Form eines Reports zur Verfügung gestellt. Diese beinhalten nicht nur Informationen darüber, dass eine Sicherheitslücke entdeckt wurde wurde, sondern geben dem Benutzer auch Informationen, wie er diese Lücke schließen kann. Desweiteren teilt Nessus gefundene Sicherheitslöcher in Risikolevel ein, warnt den Anwender über mögliche Sicherheitslücken und gibt zusätzlich Informationen über neuere Versionen der Software. Diese Report - Dateien können nach ASCII Text, LaTeX, HTML (mit Graphen) und ein ein einfach zu parsendes File - Format exportiert werden, um dort gegebenenfalls weiterverarbeitet zu werden. - SSL – Unterstützung Nessus bietet Ihnen zusätzlich die Möglichkeit SSL-basierte Dienste wie https, smtps, imaps et cetera zu testen.

3.4 Rapid7 Nexpose

Nexpose ist eine Schwachstellenmanagement-Software, die Netzwerkumgebungen oder einzelene Hosts auf Sicherheitslücken prüfen kann. Sie wird von Rapid7 in folgenden Versionen

vertrieben:

- Enterprise – für mittlere und große Unternehmen
- Consultant – für Sicherheitsberatungsunternehmen
- Express – für kleine Unternehmen
- Community – als freie Version für einzelne Nutzer

Die einzelnen Versionen unterscheiden sich im Umfang der bereitgestellten Features und um die Anzahl der zu prüfenden IP-Adressen. Die Community-Version erlaubt nur das Scannen von 32 IP-Adressen. Dies ist zu Testzwecken und für die Nutzung in kleineren Netzwerken aber eine durchaus brauchbare Anzahl.

3.5 Softwarevergleich

Während NMAP ein Werkzeug zur Entdeckung von Hosts und Diensterkennung im Netzwerk ist, sind Schwachstellenscannern umfassende Sicherheitstools für die Bedrohungsüberwachung, Bewertung und Priorisierung von Risikos. Der Netzwerkscanner ist ein integraler Bestandteil von allen Schwachstellenscannern. Im Gegensatz zu NMAP, das keine grafische Oberfläche hat [1], werden alle anderen Softwareprodukte über eine Weboberfläche verwaltet. Die Fähigkeiten von Netzwerkscannern sind vergleichbar und lassen sich den Adresse- und Portsbereich anpassen, Präzision des Scans wählen. Die Systemanforderungen sind in der Tabelle 3.1 angegeben.

Alle Softwareprodukte bieten die Einstellungen zur Steuerung der Scangeschwindigkeit. Nmap hat einen einfacheren Ansatz mit sechs Timing-Templates. Diese kann man mit der Option -T und ihrer Nummer (0–5) oder mit ihrem Namen angeben. Diese Template-Namen lauten: paranoid (0), sneaky (1), polite (2), normal (3), aggressive (4) und insane (5). Die ersten beiden sind für die Umgehung von IDS[2] gedacht. Der Polite-Modus verlangsamt den Scan, damit er weniger Bandbreite und Ressourcen auf dem Zielrechner verbraucht. Der Normal-Modus ist der Standardwert, d.h. -T3 macht gar nichts. Der Aggressive-Modus beschleunigt Scans, indem er davon ausgeht, dass Sie sich in einem einigermaßen schnellen und zuverlässigen Netzwerk befinden. Und schließlich geht der Insane-Modus davon aus, dass sie sich in einem außergewöhnlich schnellen Netzwerk befinden bzw. gewillt sind, für mehr Geschwindigkeit auf etwas Genauigkeit zu verzichten.

Bei dem OpenVAS-Scanner ist möglich die Anzahl der Hosts und der gleichzeitig durchzuführenden Prüfungen anzupassen. Nexpose stellt mehrere Profile für die Diensterkennung, die die sich in der Scangeschwindigkeit unterscheiden.

[1] separat kann Zenmap heruntergeladen werden – eine graphische Oberfläche für NMAP
[2] Intrusion-Detection- und -Prevention-Systeme (IDS/IPS)

	NMAP	OpenVAS	Nessus	Nexpose
OS	Windows seit NT Linux (alle) MacOS FreeBSD	Linux image	Windows seit 7 Windows Server seit 2008 Linux MacOS	Ubuntu seit 14.04 Windows seit 7 Windows Server seit 2008R2 RHEL Server seit v.6 CentOS 7 Oracle Linux 7 SUSE Linux ES 12 openSUSE Leap 15
CPU	k. A.	2x	4x2GHz	2x
RAM	k. A.	2GB	4GB (8GB empfohlen)	8GB
Disk	k. A.	9GB	30GB	100GB
andere Anforde-rungen	k. A.	k. A.	k. A.	Englisches Betriebssystem mit regionalen Einstellungen für Englisch Vereinigte Staaten
Schnittstelle	Console	Web	Web	Web

Tabelle 3.1: Systemanforderungen

4 Versuchsaufbau

Im vorherigen Kapitel wurde die Software ausgewählt und verglichen, die für die Dienst- und Betriebssystemerkennung verwendet werden kann. Im nächsten Schritt müssen die konkreten Testszenarien definiert werden, um eine klare Vorstellung über die Leistungen der vorgestellten Dienstprogramme zu bekommen.

Die Tests werden wie folgt durchgeführt: es werden mehrere Testsysteme erstellt, die den Scannern zur Verfügung stehen. Gleichzeitig werden die Fähigkeiten der Programme getestet, um die Dienste zu bestimmen, die sowohl mit den Standardeinstellungen laufen, als auch mit ungewöhnlicher Konfiguration funktionieren. Dienste auf einem System werden auf den Ports laufen, welche Nummer von der IANA zugewiesenen abweicht. Ein anderes System wird die Dienste besitzen, die mit einer Willkommensmeldung mit einem gefälschten Programmnamen konfiguriert.

Die Dienste werden auf folgende Weise geändert:

1. Die Nummer des Ports, der abgehört wird, wird für diesen Dienst auf atypisch geändert.
2. Wenn der Dienst in seiner Grußnachricht über die Betriebssysteme berichtet, wird in diesem Fall die Nachricht so geändert, dass die Antwort statt des originellen Namens „Fake OS 4.2" enthält.
3. Der Name des Dienstes in der Grußnachricht wird auf „Fake Service 2.4" geändert.
4. Wenn es keine Möglichkeit gibt, die Nachricht zu modifizieren, wird die Begrüßungsnachricht in den Einstellungen deaktiviert.

Im Experiment nehmen teil, die meistverbreitende selbstinstallierte Linux-Distribution, die den bekanntesten Serverdiensten gestaltet: FTP, SSH, SMTP, DNS, HTTP, Datenbanken. Es werden auch mehrere Distributionen des Linux-Betriebssystems ohne verfügbare Dienste installiert, um festzustellen, wie genau das Betriebssystem erkannt wird. Alle verfügbaren Testsysteme sollen in einem gemeinsamen Klasse C Netzwerk verbinden.

Vor dem Start des Tests werden die verfügbaren Ports und die Prozesse, die sie überwachen, mit dem Befehl überprüft:

```
netstat -tulpn
```

aus den Paketen der network-tools.

4.1 Testszenario 1

Der nächste Schritt besteht darin, von einem separaten Host mit dem zu testenden Programm aus dem Abschnitt 2 den Bereich der aktiven IP-Adressen des Testnetzwerks zu starten, d. h. es läuft immer nur ein System mit Scanner und gleichzeitig abgetastete Systeme. Dies ist notwendig, um den Wettbewerb für begrenzte Host-Computerressourcen zwischen den

Scanprogrammen zu vermeiden. Die Sicherheitsscanner werden für den Test folgendermaßen konfiguriert:

1. Der Adressraum ist 192.168.122.2-192.168.122.150 (Insgesamt 149 IP Adresse)
2. Ports sind TCP 1-10240 und 27001-27032 (Insgesamt 10270 Ports)
3. Falls der Sicherheitsscanner ein voreingestelltes Profil für Diensterkennung bietet, wird dieses Profil verwendet. Ansonsten wird ein Profil erstellt, die alle verfügbare Methode zur Identifizierung von Dienstleistungen und Betriebssystemen anwendet.

4.2 Testszenario 2

In diesem Szenario wird die Korrektheit der Betriebssystemerkennung ohne die installierten Dienste ermittelt. Ähnlich wie in Szenario 1 werden hier die Systeme mit Scanprogrammen nacheinander gestartet. Das Scan-Profil aus dem Testszenario 1 wird so modifiziert, dass der Adressraum bis auf 3 Adressen reduziert wird d.h. es werden genau 3 IP-Adressen von Systemen ohne Diensten abgetastet. Die von den Programmen verbrachte Zeit wird in diesem Fall in der Evaluation ignoriert.

In der Endphase werden die Testergebnisse in Tabellen zusammengefasst und nach folgenden Parametern verglichen: Richtigkeit der Servicetypdefinition, Zuverlässigkeit der Erkennung von Programmnamen und Betriebssystemen, sowie seinen Versionen und der gesamten Scan Zeit.

5 Installation und Konfiguration

In diesem Kapitel wird die für die Entwicklung und zum Testen genutzte Umgebung beschrieben.

5.1 Beschreibung der Testsumgebung

Im Rahmen dieser Arbeit, um die Richtigkeit der Bestimmung der installierten Dienste zu testen, wurde auf einem Computer mit dem Betriebssystem Debian 9 Stretch eine KVM virtuelle Maschine installiert. Die Kernel-based Virtual Machine [1] ist eine Infrastruktur des Linux-Kernels zur Virtualisierung, die auf mit den Hardware-Virtualisierungstechniken von Intel (VT) oder AMD (AMD-V) ausgestatteten X86-Prozessoren lauffähig ist. Ein Vorteil von KVM ist, dass die Gastsysteme fast mit nativer Geschwindigkeit laufen, d.h. das Gastsystem reagiert nahezu so schnell wie ein natives System. QEMU emuliert die gesamte Hardware eines Computers und durch die dynamische Übersetzung der Prozessorinstruktionen des Gastprozessors [2] in Instruktionen für den Wirtprozessor [3] eine sehr gute Ausführungsgeschwindigkeit erreicht. Die Installation erfolgt mit folgendem Befehl:

```
apt-get install qemu-kvm virt-manager
```

Für die komfortable Bedienung aller virtuellen Maschinen wurde zusätzlich das virt-manager installiert. Danach wurde durch virt-manager Menü ein vollständig kontrolliertes virtuelles Netzwerk 192.168.122.0/24 eingerichtet 5.1. Diese Testumgebung setzt sich aus folgenden Systemen zusammen:

1. Systeme 1: Debian 9 Stretch (aktuellste Version) alle Dienste per Voreinstellung.
2. Systeme 2: Debian 9 Stretch alle Dienste mit modifizierten Ports.
3. Systeme 3: Debian 9 Stretch alle Dienste mit modifizierten Antworten
4. Systeme 4: Debian 8.2 Jessie (Version vom September 2015) alle Dienste per Voreinstellung.
5. Systeme 5: CentOS ohne Diensten.
6. Systeme 6: ArchLinux ohne Diensten.
7. Systeme 7: OpenSUSE ohne Diensten

Für jedes System wurde eine separate virtuelle Maschine mit den folgenden technischen Daten erstellt: 2 CPU, 1GB Arbeitspeicher, 9GB Festplatte, Gigabit-Netzwerkadapter e1000.

[1] KVM; deutsch „Betriebssystem-Kern-basierte virtuelle Maschine"
[2] englisch guest
[3] englisch host

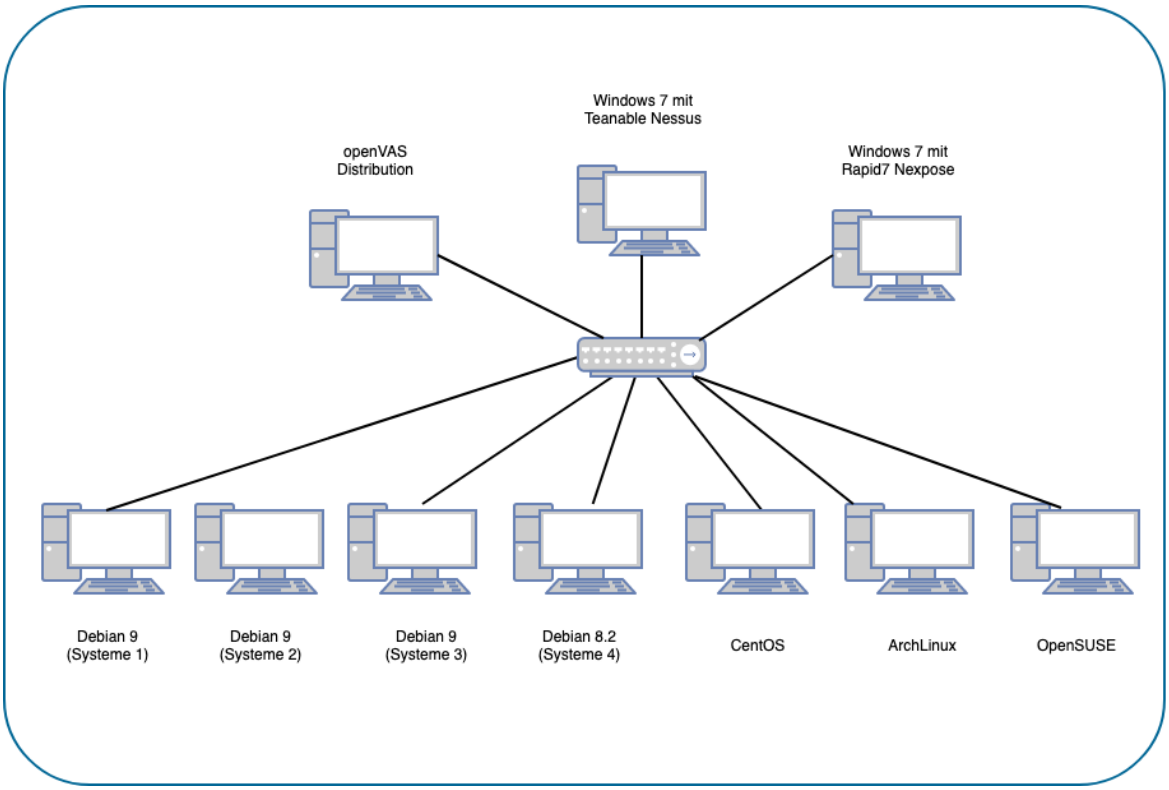

Abbildung 5.1: Netzwerk virbr0 in der QEMU virtuellen Umgebung

Da es keine speziellen Anforderungen an der Software besteht, wichtig ist lediglich ein ausreichender Festplattespeicher für mehrere Betriebssysteme sowie ausreichend Arbeitsspeicher für mehrere Gastprozessors zu haben.

Dienstname	Softwarepaket	Protokoll	IANA Port
FTP	vsftpd	TCP	21
SSH	openssh	TCP	22
SMTP	postfix	TCP	25
DNS	bind9	TCP	53
HTTP	NGINX	TCP	80
Datenbank	mysql	TCP	3306
HTTP	Apache	TCP	8080
Datenbank	Mongo DB	TCP	27017

Tabelle 5.1: Serverdienste

Die Systeme 1 bis 4 sind mit erreichbaren Diensten aus der Tabelle 5.1 aufgebaut.

5.2 Installation der Software

5.2.1 Der Server auf Basis von neuster Debian-Distribution.

Das System 1 wird auf Basis vom aktuellsten Betriebssystem „Debian 9 Stretch" aufgebaut: Alle diese Programme außer MongoDB sind in den offiziellen Paketquellen von Debian enthalten und können mit dem Absetzen des folgenden Befehls installiert werden:

```
apt-get install -y vsftpd openssh-server postfix bind9 apache2
   nginx mysql-server
```

Das mongodb-org-Paket kann mit den mongodb-Paketen im Debian-Repository in Konflikt geraten. Deswegen muss zuerst ein öffentlicher Schlüssel in das System importiert werden:

```
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9
   DA31620334BD75D9DCB49F368818C72E52529D4
```

und dann kann die MongoDB-Repository als die Quelle für Paket-Manager hinzugefugt werden:

```
echo ‘‘deb http://repo.mongodb.org/apt/debian  stretch/mongodb-
   org/4.0 main ‘‘ | tee /etc/apt/sources.list.d/mongodb-org
   -4.0.list
```

Nach dem Befehl:

```
apt-get update
```

lässt sich Mongo-DB auch durch den Paket-Manager installieren:

```
apt-get install -y mongodb-org
```

Dienstname	Softwarepaket	Version
FTP	vsftpd	3.0.3
SSH	openssh	7.4p1
SMTP	postfix	3.1.9-0+deb9u2
DNS	bind9	9.10.3-P4-Debian
HTTP	NGINX	1.10.3
Datenbank	mysql	15.1
HTTP	Apache	2.4.25-3+deb9u2
Datenbank	Mongo DB	4.0.8

Tabelle 5.2: Software-Version

Die vollständige Liste der Dienste und ihrer Versionen ist in der Tabelle 5.2 enthalten . Die Konfiguration-Dateien für alle Diensten befinden sich im Anhang B zu dieser Arbeit.

5.2.2 Der Server mit modifizierten Dienstports

Um festzustellen, inwieweit die Ergebnisse von der Portnummer abhängen, wurden auf der Basis vom System 1 neue Systeme entwickelt. In diesem System funktionieren alle Dienste auf nicht standardisierten Ports. Die Portnummern wurden zufällig ausgewählt und sind in der folgenden Tabelle 5.3 angegeben. Das Verschieben eines Dienstes auf einen anderen Port ist eine Standardoption für alle Dienste. Dies geschieht durch das Einstellen des Portwertes in der Einstellungsdatei. Die einzige Ausnahme war bei dem SMTP-Server postfix. Es wurde folgende Modifikation in der Datei „/etc/postfix/master.cf" vorgennomen:

```
# ==========================================================
# service type  private unpriv  chroot  wakeup  maxproc command
    + args
#               (yes)   (yes)   (no)    (never) (100)
# ==========================================================
#smtp       inet  n       -       y       -       -       smtpd
80         inet  n       -       y       -       -       smtpd
```

Hier wird smtp-Daemon auf dem Standartort 53 ausgeschaltet und auf dem Port 80 eingeschaltet.

Dienstname	Softwarepaket	Protokoll	Port
FTP	vsftpd	TCP	3306
SSH	openssh	TCP	25
SMTP	postfix	TCP	80
DNS	bind9	TCP	27017
HTTP	NGINX	TCP	21
Datenbank	mysql	TCP	8080
HTTP	Apache	TCP	22
Datenbank	Mongo DB	TCP	3389

Tabelle 5.3: Severdienste (die Ports wurden geändert)

5.2.3 Der Server auf Basis von der neusten Debian-Distribution mit modifizierter Software

In dieser Arbeit wurde ein besonderes Augenmerk auf die Frage gelegt, wie stark die Ergebnisse der Diensterkennung von der Methode „Banner Grabbing" abhängen. Deswegen wurden die Systeme mit Debian OS mehrmals getestet: einmal mit Komponenten aus dem Repository und ein zweites Mal mit einer modifizierten Software, sodass die Version in der Antwort durch eine andere ersetzt oder der Banner komplett ausgeschaltet wurde. Die Software wurde wie folgt modifiziert:

vsftpd

Vsftpd unterstützt benutzerdefinierte Willkommensbanner. Standardmäßig werden die vsftpd-Konfigurationsdateien in /etc/ gespeichert. Die Hauptkonfigurationsdatei ist /etc/vsftpd.conf. Der Parameter ftpd_banner lässt sich eine einzeilige Willkommenszeichenkette für nicht authentifizierte Benutzer konfigurieren. Für die Tests wurde der Wert wie folgt gesetzt:

```
ftpd_banner=Welcome to Fake service.
```

OpenSSH-server

Das Banner:

```
SSH-protoversion-softwareversion SP comments CR LF
```

ist Teil des SSH-Protokolls, der im Kapitel 4.2 des Standard RFC 4253 [YL06] beschrieben wurde. Die Kommentare sind optional und müssen nicht unbedingt vorhanden sein aber Debian setzt sie standardmäßig ein und gibt seine Version heraus. In der Einstellungendatei unter „/etc/ssh/sshd_config" gibt es Moglichkeit die Kommentare wie folgt auszuschalten:

```
DebianBanner no
```

Die Softwareversion kann man ändern, damit man der Quellecode neu kompiliert. Dafur wurde in der Datei „version.h" die Zeile mit der Variable „SSH_VERSION" geändert:

```
#define SSH_VERSION     ''Fake Service_2.4''
```

postfix

Um das Banner zu ändern, genügt es, in der Einstellungen „/etc/postfix/main.cf" die Option „smtpd_banner" zu modifizieren.

```
smtpd_banner = $myhostname Fake Service (Fake OS 4.2)
```

Bind9

In ähnlicher Weise funktioniert die Ausschaltung der Versionsausgabe in Bind 9. In der Datei „/etc/bind/named.conf" sind die Variable „version" für den Namen bzw. für die Version verantwortlich. Nach der Änderung:

```
version ''Fake Service 2.4'';
```

Apache

Einer der HTTP-Header, die der Apache httpd mit Antwortdaten zurückschickt, ist „Server". Zum Beispiel ist unmodifizierte Webserver-Maschine sendet Header zurück, die den folgenden ähnlich sind:

```
HTTP/1.1 200 OK
Date: Mon, 17 Jun 2019 22:32:03 GMT
Server: Apache/2.4.25 (Debian)
Last-Modified: Wed, 27 Mar 2019 17:13:52 GMT
ETag: "62-585168fce7da3-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 89
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Man kann das Option ServerSignature=off in „/etc/apache2/apache2.conf" setzen, um die Versionsausgabe auszumachen, aber die Kopfzeile „Server" bleibt und wird trodzem „Apache" enthalten. Außerdem gibt es noch andere Stelle, wie z.B. 404 Fehlerseite, wo unten der Softwarename steht. ModSecurity ist ein Toolkit zur Echtzeit-Überwachung von Webanwendungen und ermöglicht es, die Serversignatur zu ändern. Die Installation ist auch durch den apt Paket-Manager möglich:

```
apt-get install libapache2-mod-security2
```

Danach kann man mit der Option „SecServerSignature" die Ausgabe individuell anpassen:

```
SecServerSignature ''Fake Service 2.4''
```

NGINX

Die Serverantwort vom NGINX ist gleichartig wie beim Apache Server. Aber es wurde keine Modifikation gefunden, die ein solches Verhalten ändern lässt. Deswegen wurden solche Mod aus dem Quellcode erstellt. Zu diesem Zweck wurden folgende Variablen im Code geändert: in „src/http/ngx_http_header_filter_module.c"

```
static char ngx_http_server_string[] = ''Server: nginx'' CRLF;
```

und in "src/core/nginx.h"

```
#define NGINX_VER          ''nginx/'' NGINX_VERSION
#define NGINX_VERSION
```

Dementsprechend wird der ursprüngliche Name „nginx" durch „Fake Service" ersetzt.

```
static char ngx_http_server_string[] = ''Server: Fake Service''
    CRLF;
static char ngx_http_server_full_string[] = ''Server: ''
   NGINX_VER CRLF;

#define NGINX_VERSION      ''2.4''
#define NGINX_VER          ''Fake Service' ' NGINX_VERSION
```

Die vollständigen Konfigurationsdateien und Quellcodes sind im Anhang B angegeben.

5.2.4 Der Server auf Basis von veralteter Debian-Distribution.

Das vierte System läuft unter dem alten Betriebssystem Debian 8 und entsprechend den zugehörigen Diensten alt 5.4: Diese Pakete wurden aus dem offiziellen Repository herunter-

Dienstname	Softwarepaket	Version
FTP	vsftpd	3.0.2-17+deb8u1
SSH	openssh	6.7p1-5
SMTP	postfix	2.11.3-1+deb8u2
DNS	bind9	9.9.5-9+deb8u2-Debian
HTTP	NGINX	1.6.2
Datenbank	mysql	14.14
HTTP	Apache	2.4.10-10+deb8u3
Datenbank	Mongo DB	2.6.12

Tabelle 5.4: Software-Version

geladen und mit dem Befehl:

```
apt-get install package=version
```

installiert.

5.3 Installation der Schwachstellerscanner

5.3.1 NMAP

NMAP kann aus den offiziellen Debian-Paketquellen installiert werden:

```
apt-get install nmap
```

Alternativ kann auch die neueste Version von nmap kompiliert werden. Zuerst muss von der `https://nmap.org/dist/` der aktuelle Quelltext als eine Archivdatei heruntergeladen werden. Um Nmap zu kompilieren, sollte im Terminal die folgenden Befehle im Downloadverzeichnis ausgeführt werden:

```
tar -xjvf nmap-VERSION.tar.bz2
cd nmap-VERSION
./configure
make
make install
```

5.3.2 OpenVAS

Anstatt OpenVAS unter Linux zu installieren, können wir die virtuelle OpenVAS-Appliance in einer virtuellen Maschine laufen lassen. Die virtuelle Appliance kann über `https://www.greenbone.net/en/install_use_gce/` heruntergeladen werden. Nach dem Herunterladen der Virtual Appliance von der OpenVAS-Website muss man eine neue Virtual Machine konfigurieren. In dieser Arbeit wird KVM verwendet, aber man kann natürlich auch andere Hypervisor wie VMware, Hyper-V oder VirtualBox benutzen. Die virtuelle Maschine ist mit empfehlenden Anforderungen konfiguriert: 2 CPU, 2GB Arbeitsspeicher und eigene virtuelle 9GB Festplatte. Das System ist durch Netzwerkadapter mit virbr0 verbunden. Laut der Anweisungen des Herstellers sollten Audio, USB und Diskette ausgeschaltet werden. Nach der Installation muss der Feed aktualisiert werden, denn ohne ihn können keine Scans durchgeführt werden und der SecInfo-Abschnitt bleibt leer.

5.3.3 Nessus

Nessus kann auf einer großen Anzahl von Betriebssystemen ausgeführt werden. Für die Aufsetzung wurde Windows 7 ausgewählt. Die Konfiguration von VM-Geräten entsprach den Mindestanforderungen des Herstellers: 4 CPU, 4 GB Arbeitsspeicher, 30 GB Festplatte. Die Installation ist einfach und läuft unter Windows in der Schritt-für-Schritt Anwendung, in der man die Lizenzvereinbarung akzeptieren, den Zielordner auswählen und Aktivierungscode eingeben muss. Am Ende wird ein neuer Benutzerlogin mit Password eingelegt, der später für Authentisieren in der Weboberfläche notwendig ist.

5.3.4 Nexpose

Man setzt Nexpose auf die gleiche Art und Weise wie Nessus auf. Auf einer virtuellen Maschine mit Windows 7 (4 CPU, 8 GB Ram, 32 GB ROM) führt ein Installationsassistent Sie durch die Schritte. Es gibt zwei Installationsmöglichkeiten: Sicherheitskonsole mit einer lokalen Scan-Engine und nur Scan-Engine. Für unsere Ziele sind beide Komponenten notwendig.

6 Testdurchführung

In diesem Kapitel werden zunächst die notwendigen Vorbereitungen zum Test beschrieben. Anschließend werden verschiedene Eingaben vorgestellt, um die Ergebnisse bewerten zu können.

Der Test wurde in der folgenden Reihenfolge durchgeführt:

1. alle vier virtuellen Maschinen aus Abschnitt 4.1 werden gleichzeitig auf dem Hostcomputer ausgeführt;

2. es werden vom Hostcomputer alle diese Betriebssysteme mit Network Scanner NMAP gescannt. Dafür werden folgende Kommandos verwendet:

   ```
   nmap -A -sV -p 1-10240,27001-27032 --verion-all
      192.168.122.2-150 > report.txt
   ```

3. danach werden virtuelle Maschine mit OpenVAS-, Nessus- und Nexpose-Abbild voneinander getrennt gestartet und aus der Web-Oberfläche eine vollständige Untersuchung, wie im Kapitel 4 beschrieben, eingeleitet. Nach dem Scan wurde virtuelle Maschine ausgeschaltet um den Arbeitsspeicher verbrauch zu reduzieren.

6.1 Dienstverfügbarkeit

Bevor jedes System 1-4 gescannt wurde, wurde die Verfügbarkeit der Dienste anhand der folgenden Ergebnisse überprüft:

```
netstat -tulpn
```

Die Ergebnisse des Befehls sind im Anhang aufgelistet. Alle installierten Pakete waren für das Scannen verfügbar.

6.2 Scannen mit OpenVAS

Für den Tests werden in Menü „Configuration" neue Portliste und neues Targets (IP-Adresse) angelegt. Nach dieser Schritt kann man auf Basis „System Discovery" Konfiguration ein neues Scan-Config erstellen, das alle mögliche NVTs aus der Gruppe Diensterkennung enthält. Der neue Task wird unter „Scan-Tasks" eingerichtet. Weitere Einstellungen im angebotenen Fenster sind wie in der Abbildung dargestellt 6.1.

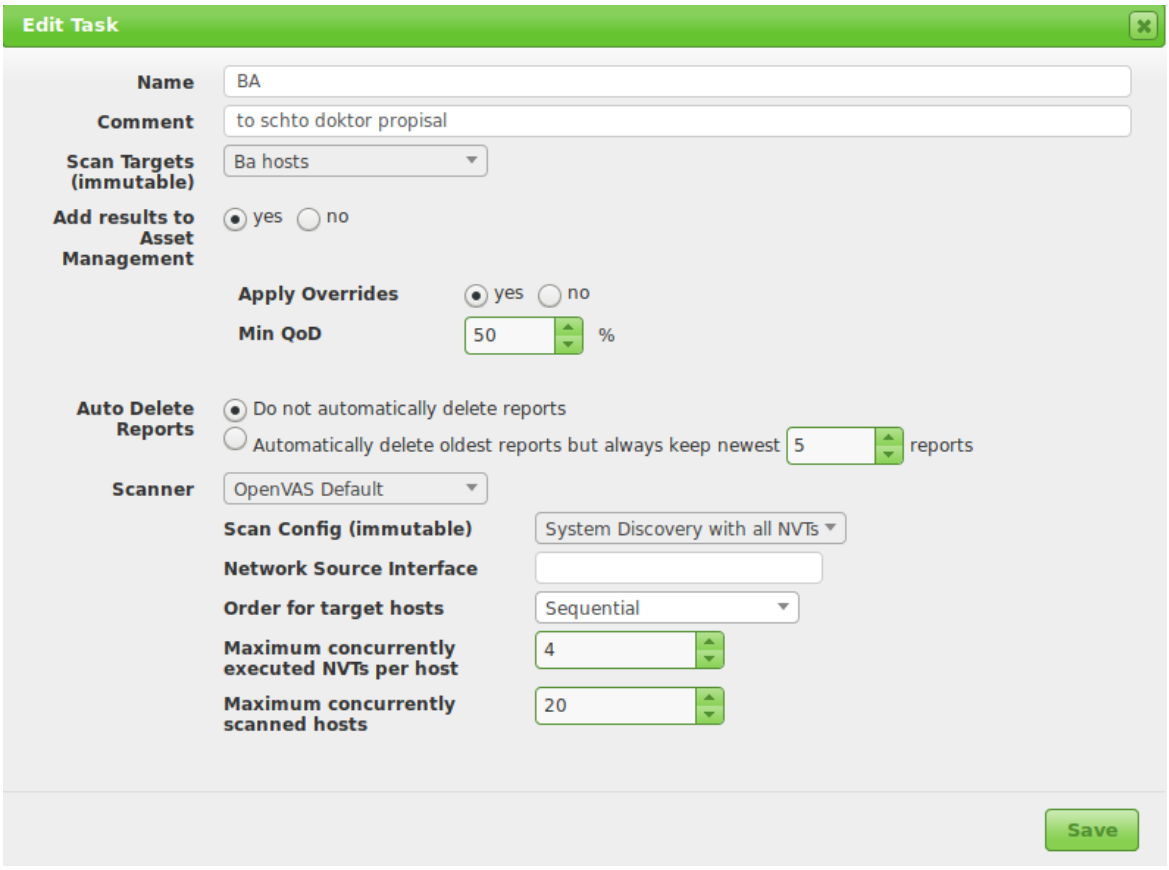

Abbildung 6.1: OpenVAS Task-Einstellungen.

6.3 Scannen mit Nessus

Eine kleine Anzahl von Schritten genügt, um einen Scan in Nessus durchzuführen. Zuerst wird im Menü „My Scan" ein neuer erweiterter Scan erstellt 6.2, wo „Targets" den IP-Adressen sind. Im Abschnitt „Service Discovery" wurden alle Ports ausgewählt. Die restlichen Optionen sind auf zusätzlichen Registerkarten versteckt. Man kann Identifikationsdaten für den entfernten Hosts eingeben, einzelne Plugins auswählen und die Methode zum Scannen von entfernten Computern feststellen.

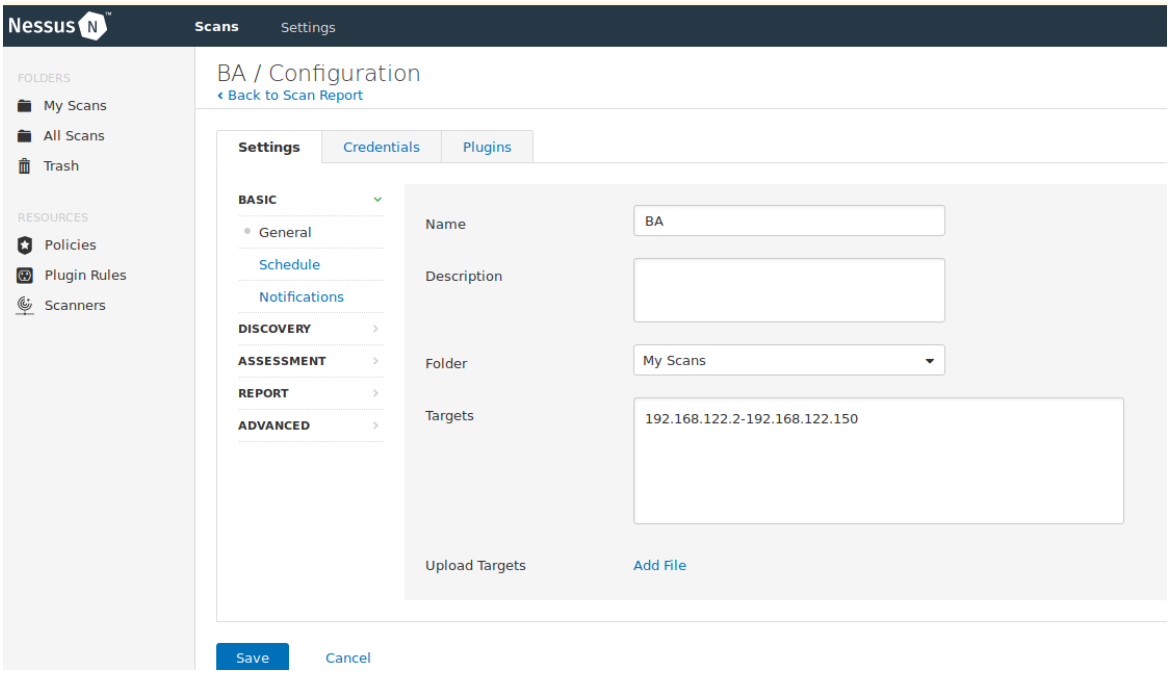

Abbildung 6.2: Nessus Scan-Einstellungen.

Nachdem die angegebene Einstellungen gespeichert haben, kann man den Scan Vorgang durch Drücken der Scan-Taste starten. Alternativ dazu kann man zeitgesteuertes Scannen einlegen.

6.4 Scannen mit Nexpose

Die Erstellung neuer Scans beginnt mit dem Klick auf die Schaltfläche „Create Site" in der DropDown-Menü auf der Homepage.

Um die Anforderungen dieser Arbeit zu erfüllen, sollte weitere Anpassungen vorgenommen werden:

- Auf der Registerkarte „Assets" sollte den IP-Bereich festgestellt werden.
- Wie OpenVAS und Nessus hat Nexpose vorangestellte Scan-Profile 6.3. Für die Diensterkennung sind „Discovery Scan" und „Discovery Scan - Aggressive" vorgesehen. Der Unterschied dazwischen besteht in der Geschwindigkeit des Scans. In diesen Profilen

ist eine begrenzte Anzahl von Ports voreingestellt, hauptsächlich die meist verwendete Dienste. Aus diesem Grund wurde dieses Profil als Grundlage genommen und alle andere Ports wurden gemäß den Versuchsaufbau hinzugefügt.

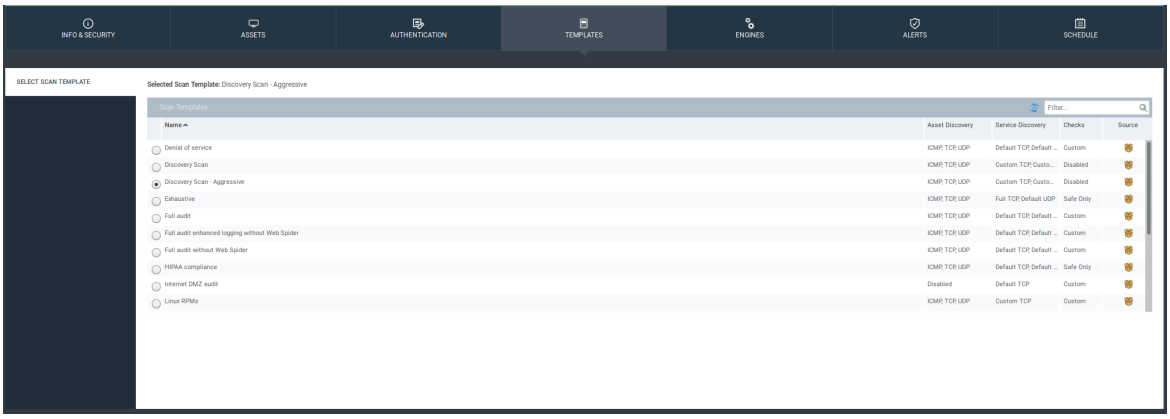

Abbildung 6.3: Nexpose voreingestellte Profile.

Der letzte Schritt im Konfiguration-Prozess ist die Übersicht über alle Einstellungen 6.4.

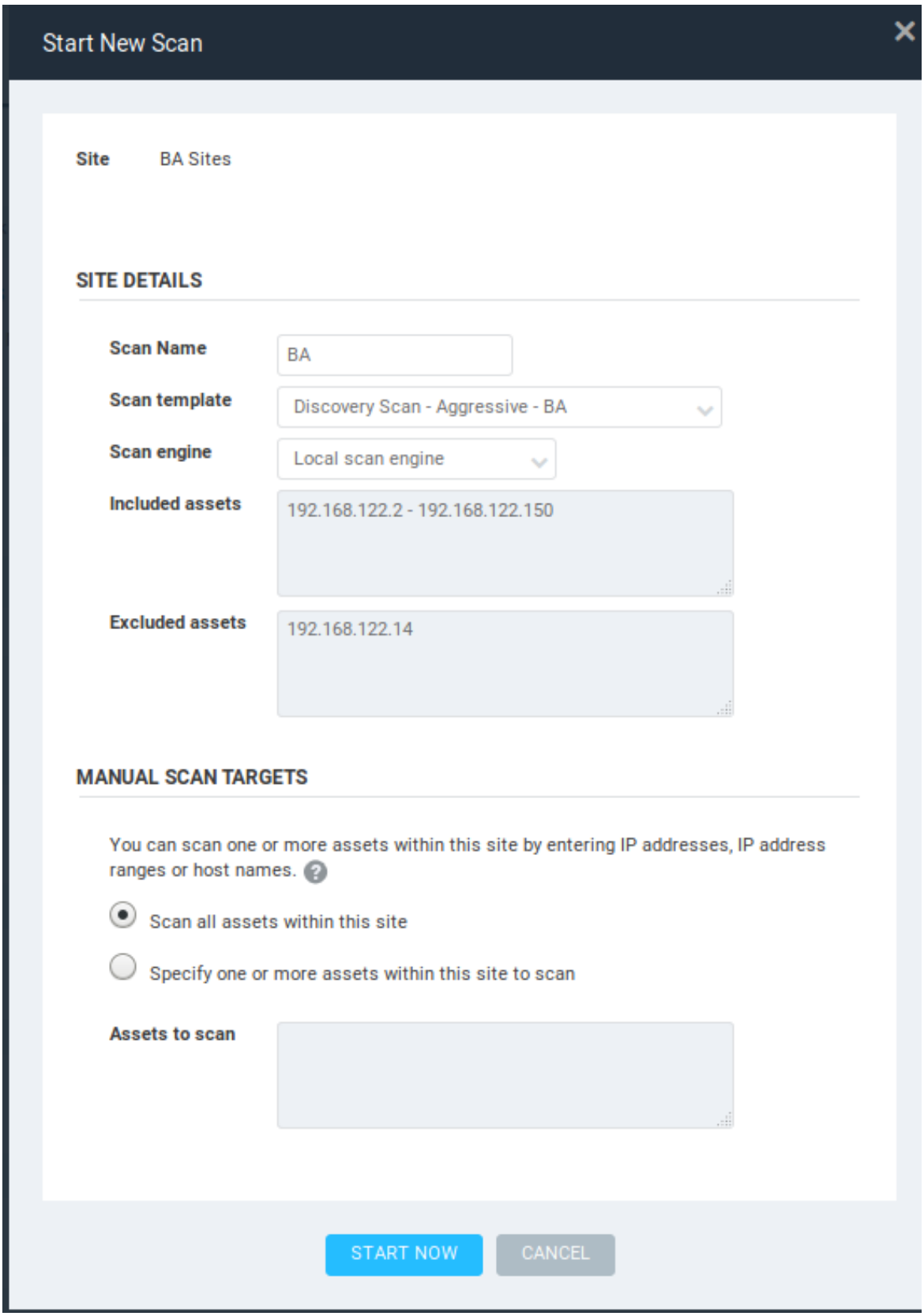

Abbildung 6.4: Nexpose Scan-Übersicht.

7 Evaluation

Im Rahmen dieses Kapitels werden die Ergebnisse der Evaluation vorgestellt. Hierfür werden zunächst die Testdaten tabellarisch zusammengefasst. Danach erfolgt eine Gegenüberstellung von Schwachtellerscannern mit NMAP und in Abhängigkeit von dem Resultat werden die möglichen Ursachen angegeben.

7.1 Beschreibung der Evaluation

Ziel dieser Arbeit ist es, verschiedene Anwendungen für die Diensterkennung zu vergleichen. Dafür wurde ein Testnetz aus 7 Maschinen erstellt, von denen 4 für die Diensterkennung und des Betriebssystems und 3 zusätzlich nur für die Betriebssystemerkennung aufgebaut wurden. Alle Softwareprodukte wurden möglichst mit den gleichen Parametern konfiguriert: mit demselben IP-Bereich und 10270 Ports, von denen 8 offen waren. Wie in vorhergehenden Abschnitten beschrieben ist, wurden die im Testnetz gewonnenen Daten für die Evaluation des Tools genutzt.

Die Evaluation selbst geschieht in mehreren Schritten: zuerst wird die Diensttyperkennung verglichen, dann die Dienstname- und Version-Erkennung und am Ende die Betriebssystemerkennung. Als optionaler Parameter wird auch die Abtastzeit verglichen.

7.2 Ergebnisse der Evaluation

7.2.1 Diensttyperkennung

Für das Scannen standen häufig genutzte Dienste zur Verfügung: FTP, SSH, SMTP, DNS, HTTP, Datenbanken. Die Daten wurden nach den Systemen und Schwachstellescannen gruppiert und die Anzahl der erkannten Diensten und Gesamtzahl sind in der Tabelle dargestellt.

Aus der Tabelle 7.1 kann man entnehmen, dass die Scanner in der Lage sind, die Diensttyp korrekt erkennen. Es spielt keine Rolle weder auf welchem Port der Dienst läuft noch

Diensttyp	NMAP	OpenVAS	Nessus	Nexpose
System 1	8/8	7/8	8/8	7/8
System 2	8/8	7/8	8/8	4/8
System 3	8/8	7/8	8/8	7/8
System 4	8/8	7/8	8/8	7/8

Tabelle 7.1: Diensttyperkennung. Die Anzahl der erkannten Diensten aus der Gesamtzahl

wie er sich selbst nennt. Die Ergebnisse von Rapid 7 Nexpose werden vor dem allgemeinen Hintergrund hervorgehoben. Der hat MongoDB Datenbank und Postfix-Smtp-Client als HTTP-Dienste identifiziert.

7.2.2 Dienstname- und Version-Erkennung

Die gewonnenen Daten über jedes System sind in Tabellen zusammengefasst. Alle vier Softwarelösungen zeigten bei der Erkennung der auf den Standardports betriebenen Dienste positive Resultate7.2. Die konnten betriebenen Dienste fehlerfrei erkannt werden. Die einzelne Ausnahme war MongoDB Datenbank bei OpenVAS und Nexpose. Eine ähnliche Situation war bei der Identifizierung der alten Dienste 7.5, die auf das System 4 gestartet wurden.

Dienstname	Software	Version	NMAP	OpenVAS	Nessus	Nexpose
FTP	vsftpd	3.0.3	vsftpd 3.0.3	vsFTPd 3.0.3	vsFTPd 3.0.3	vsftpd 3.0.3
SSH	openssh	7.4p1	OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)	OpenSSH	OpenBSD OpenSSH 7.4	OpenSSH 7.4p1
SMTP	postfix	3.1.9-0 +deb9u2	Postfix smtpd	Postfix	Postfix	Postfix
DNS	bind9	9.10.3 -P4-Debian	ISC BIND 9.10.3-P4-Debian	9.10.3 -P4-Debian	ISC BIND 9.10.3 P4	BIND 9.10.3 -P4-Debian
HTTP	NGINX	1.10.3	nginx 1.10.3	nginx/ 1.10.3	nginx/ 1.10.3	nginx 1.10.3
Datenbank	mysql	15.1	MariaDB	MySQL/ MariaDB	MariaDB	MariaDB
HTTP	Apache	2.4.25-3 +deb9u2	Apache httpd 2.4.25 ((Debian))	Apache 2.4.25	Apache/ 2.4.25	httpd 2.4.25
Datenbank	MongoDB	4.0.8	MongoDB 4.0.8		MongoDB HTTP	

Tabelle 7.2: Dienstname- und Version-Erkennung (Systeme 1)

Die Änderung der Ports wirkte sich negativ auf die Identifizierung der Dienste von nur einem Scanner aus 7.3. Nexpose war nicht mehr in der Lage FTP und SMTP Dienst zu erkennen.

Die negativen Ergebnisse der Definition von Diensten und deren Versionen in System 3 7.4, deuten darauf hin, dass die wichtigste Informationsquelle das Banner ist. In fast allen Fällen wurden die Dienste, die zuvor mit einem umgeänderten Banner korrekt identifiziert wurden, als Fake-Service bezeichnet. Nur NMAP konnte die Namen der FTP- und SMTP-Dienste korrekt erkennen.

Basierend auf diese Information kann man schließen, dass Open-Source Network Scanner NMAP hat die größtmögliche Menge von Angaben eingeholt und hat noch andere Methode, die vom Banner-Grabbing abweichen.

7.2.3 Betriebssystemerkennung

Die Ergebnisse der Betriebssystemdefinition 7.6 sind stark abhängig von den installierten Diensten. Die wichtigste Informationsquelle über das Betriebssystem ist das OpenSSH-Banner. Basierend auf diesen Daten haben Nessus- und NMAP-Scanner ihre Schlussfolgerungen gezogen. Auf andere Weise hat OpenVAS die Betribssystem korrekt erkannt: bei

Dienstname	Software	Version	NMAP	OpenVAS	Nessus	Nexpose
FTP	vsftpd	3.0.3	vsftpd 3.0.3	vsFTPd 3.0.3	vsFTPd 3.0.3	
SSH	openssh	7.4p1	OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)	OpenSSH	OpenBSD OpenSSH 7.4	OpenSSH 7.4p1
SMTP	postfix	3.1.9-0 +deb9u2	Postfix smtpd	Postfix	Postfix	
DNS	bind9	9.10.3 -P4-Debian	ISC BIND 9.10.3-P4-Debian	9.10.3 -P4-Debian	ISC BIND 9.10.3 P4	
HTTP	NGINX	1.10.3	nginx 1.10.3	nginx/ 1.10.3	nginx/ 1.10.3	nginx 1.10.3
Datenbank	mysql	15.1	MariaDB	MySQL/ MariaDB	MariaDB	MariaDB
HTTP	Apache	2.4.25-3 +deb9u2	Apache httpd 2.4.25 ((Debian))	Apache 2.4.25	Apache/ 2.4.25	httpd 2.4.25
Datenbank	MongoDB	4.0.8	MongoDB 4.0.8		MongoDB HTTP	

Tabelle 7.3: Dienstname- und Version-Erkennung (Systeme 2)

Dienstname	Software	Version	NMAP	OpenVAS	Nessus	Nexpose
FTP	vsftpd	3.0.3	vsftpd 2.0.8 or later	Fake Service	Fake Service	
SSH	openssh	7.4p1	(protocol 2.0)	(protocol 1.99, 2.0)	(protocol 1.99, 2.0)	
SMTP	postfix	3.1.9-0+ deb9u2	Postfix smtpd		Fake Service (Fake OS 4.2)	
DNS	bind9	9.10.3 -P4-Debian	ISC BIND Fake Service 2.4	Fake Service 2.4	Fake Service 2.4	ISC BIND Fake Service 2.4
HTTP	NGINX	1.10.3	Fake Service 2.4	Fake Service 2.4	Fake Service2.4	Fake Service2.4
Datenbank	mysql	15.1	MariaDB	MySQL	MariaDB	MariaDB
HTTP	Apache	2.4.25-3 +deb9u2	Fake Service 2.4	Fake Service 2.4	Fake Service 2.4	Fake Service 2.4
Datenbank	MongoDB	4.0.8	MongoDB 4.0.8		MongoDB HTTP	

Tabelle 7.4: Dienstname- und Version-Erkennung (Systeme 3)

Dienstname	Software	Version	NMAP	OpenVAS	Nessus	Nexpose
FTP	vsftpd	3.0.2-17+ deb8u1	vsftpd 3.0.2	vsFTPd	vsFTPd 3.0.2	vsFTPd 3.0.2
SSH	openssh	6.7p1	OpenSSH 6.7p1 Debian 5 (protocol 2.0)	OpenSSH	SSH-2.0-OpenSSH_6.7p1 Debian-5	OpenSSH 6.7p1
SMTP	postfix	2.1.13-1 +deb8u2	Postfix smtpd	Postfix	Postfix	Postfix
DNS	bind9	9.9.5-9 -deb8u2		9.9.5-9 +deb8u2 -Debian	9.9.5-9 +deb8u2 -Debian	BIND 9.9.5-9 +deb8u2 -Debian
HTTP	NGINX	1.6.2	nginx 1.6.2	nginx/ 1.6.2	nginx/ 1.6.2	nginx 1.6.2
Datenbank	mysql	14.14	MySQL	MySQL	MySQL	MySQL
HTTP	Apache	2.4.10-10 +deb8u3	Apache httpd 2.4.10 ((Debian))	Apache/ 2.4.10 (Debian)	Apache/ 2.4.10 (Debian)	HTTPD 2.4.10
Datenbank	MongoDB	2.6.12	MongoDB 2.6.12		MongoDB HTTP	

Tabelle 7.5: Dienstname- und Version-Erkennung (Systeme 4)

ihm hat er mehrere Versuche einige php-Seite zu finden und phpinfo()-Methode Ausgabe herauszukriegen. In allen anderen Fällen sind die Scanner durch die früheren beschriebenen Unterschiede in der Implementierung des TCP-IP-Protokollstapels ausgerichtet.

7.2.4 Scanzeit

Da die Wahl der passenden Werte manchmal mehr Zeit erfordern als der Scan selbst, den man damit optimieren möchten, wurden in dieser Arbeit alle Scanner auf der Basis von vordefinierten Angaben gegenübergestellt. Die Scan Zeit ist in der folgenden Abbildung dargestellt.

Um die besten Ergebnisse NMAP zu erzielen, brauchte der mehrere Zeiten als andere Scanner. Nexpose- und OpenVAS-Ergebnisse sind sowohl in der Genauigkeit als auch in der Scanzeit vergleichbar. Die unvollständige Diensterkennung von Nexpose wird durch einen sehr schnellen Scan ausgeglichen.

OS (Kernel)	NMAP	OpenVAS	Nessus	Nexpose
System 1 Debian 9 (4.9.0-8)	Linux 3.2 - 4.6	Debian GNU/ Linux 9	Linux Kernel 3.12	Linux 3.11
System 2 Debian 9 (4.9.0-8)	Linux 3.2 - 4.6	Debian GNU/ Linux 9	Linux Kernel 3.12	Debian Linux 9.0
System 3 Debian 9 (4.9.0-8)	Linux 3.2 - 4.6	Debian GNU/ Linux 9	Linux Kernel 3.12	Debian Linux 9.0
System 4 Debian 8.2 (3.16.0-4)	Linux 3.2 - 4.6	Debian GNU/ Linux 8	Linux Kernel 3.16 on Debian 8.0 (jessie)	Debian Linux
System 5 CentOS (3.10.0)	Linux 3.10 - 4.2 Linux 3.2 - 4.6	Linux Kernel	OpenBSD	Linux 3.11
System 6 ArchLinux (5.1.5)	-	Linux Kernel	-	-
System 7 openSUSE (4.12.14)	-	Linux Kernel	-	-

Tabelle 7.6: Betriebssystemerkennung

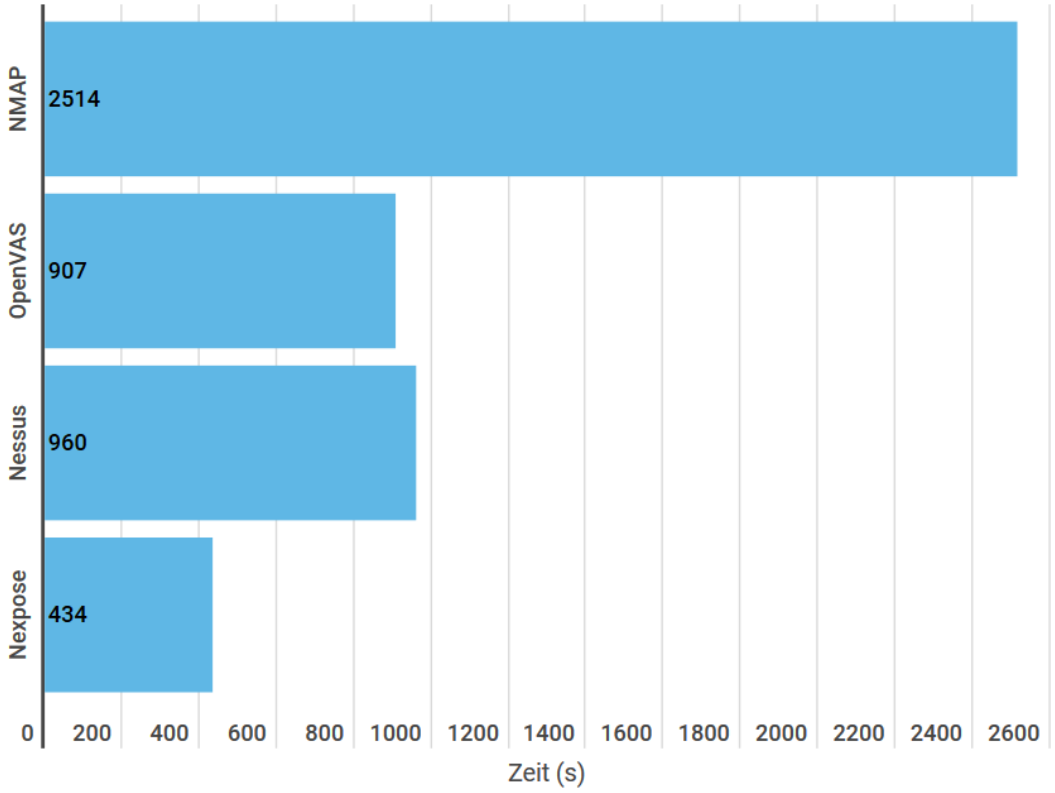

Abbildung 7.1: Abtastezeitvergleich.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

In dieser Arbeit wurden die bekanntesten kommerziellen Schwachstellenscanner mit dem freien Open-Source Werkzeug NMAP verglichen und die dafür benötigte Entwicklung einer Evaluierungsumgebung für Diensterkennung von Schwachstellenscanner beschrieben. Zum Vergleich wurden Tests durchgeführt. Diese sollten unter den gleichen Bedingungen für alle Werkzeuge stattfinden, damit die Resultate miteinander vergleichbar waren.

Dazu wurden in Kapitel 2 die Grundlagen für das Verständnis der Arbeit vermittelt. Es wurde ein besonders auf die Arbeitsweise von Fingerprint eingegangen und eine bekannte Methode für die Dienst- und Betriebssystemerkennung vorgestellt.

Im nächsten Kapitel wurde die Software ausgewählt und verglichen, die für die Dienst- und Betriebssystemerkennung verwendet werden kann. Im Kapitel 4 wurden die konkreten Testszenarien definiert, sodass die gewonnenen Daten später zusammengefasst und evaluiert werden konnten.

Die Implementierung der Umgebung wurde in Kapitel 5 besprochen. Es wurde ein virtuelles Netz eingerichtet, in dem unterschiedliche Betriebssysteme mit meist verwendeten Diensten zur Verfügung gestellt wurden. Im Anschluss erfolgte in Kapitel 6 die Beschreibung der Evaluierung in der festgestellt wurde, dass die Dienstklassifizierung in freien Open-Source Werkzeug NMAP beste Ergebnisse erzielt hat.

8.2 Fazit

Nach den Daten, die im vorherigen Abschnitt vorgestellt wurden, können wir feststellen, dass die Definition der Version der Dienste und des Betriebssystems in kommerziellen Produkten nicht besser als bei freier Software ist. Führende Produkte im Bereich der Sicherheitsscanner haben keine zusätzlichen Fähigkeiten zur Identifizierung von Diensten nachgewiesen.

Die einfachsten Änderungen in der Bannerantwort von Diensten ermöglicht es, alle im Test vorgestellten Softwarepakete leicht zu täuschen. Die Definition des Betriebssystems basiert auf den Unterschieden in der Implementierung des TCP/IP-Stacks und ermöglicht es, die Version des Linux-Kernels grob zu identifizieren. Ohne die installierten Dienste, die die zusätzlichen Informationen ausgeben würden, wäre eine genaue Definition der Distribution oder gar der Kernelversion nicht möglich.

Dementsprechend ist die beste Lösung aus der Sicht der gestellten Aufgabe zur Definition von Software auf entfernten Computern ohne die Daten zur Identifizierung, die NMAP. Vorteile kommerzieller Produkte sind die Vielfalt der Sicherheitslösungen und der Risikobewertung. Darüber hinaus kann man mit Hilfe von Authentifizierungsdaten, Daten über die Version von Produkten und mögliche Schwachstellen erhalten. Wenn man beispielsweise ein Login und ein Passwort für eine SSH-Sitzung verfügt, kann man potenzielle Schwachstellen zuverlässig identifizieren.

8.3 Ausblick

Das Ziel dieser Arbeit Open-Source Scanner NMAP mit anderen Open-Source und kommerziellen Schwachstellenscannern zu vergleichen wurden erreicht. Die gewonnenen Daten konnten in verschiedenen Bereichen Einsatz finden wie z. B. im Aufbau von Honeypot-Systemen[1].

In zukünftigen Arbeiten kann die Scan Zeit weiter untersucht werden. Es bleibt noch die Frage, wie stark sich die Ergebnisse verschlechtern werden, falls das Scan mit einem sehr aggressiven Profil durchgeführt wird.

Darüber hinaus kann die Anzahl der Dienste noch erweitert werden. Die Ergebnisse der Tests zeigten, dass die neue, schnell wachsende Software (wie MongoDB) kaum erkannt wurde.

[1] Als Honeypot wird ein Server bezeichnet, der die Netzwerkdienste eines Computers, eines ganzen Rechnernetzes oder das Verhalten eines Anwenders simuliert. Honeypots werden eingesetzt, um Informationen über Angriffsmuster und Angreiferverhalten zu erhalten.

A. Dienstverfügbarkeit-Protokolle

Systeme 1:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
           State       PID/Program name
tcp        0      0 0.0.0.0:27017           0.0.0.0:*
                  LISTEN      446/mongod
tcp        0      0 0.0.0.0:3306            0.0.0.0:*
                  LISTEN      562/mysqld
tcp        0      0 0.0.0.0:80              0.0.0.0:*
                  LISTEN      642/nginx: master p
tcp        0      0 192.168.122.52:53       0.0.0.0:*
                  LISTEN      449/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*
                  LISTEN      449/named
tcp        0      0 0.0.0.0:21              0.0.0.0:*
                  LISTEN      456/vsftpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*
                  LISTEN      564/sshd
tcp        0      0 0.0.0.0:25              0.0.0.0:*
                  LISTEN      793/master
tcp        0      0 127.0.0.1:953           0.0.0.0:*
                  LISTEN      449/named
tcp6       0      0 :::80                   :::*
                     LISTEN      642/nginx: master p
tcp6       0      0 :::8080                 :::*
                     LISTEN      688/apache2
tcp6       0      0 :::53                   :::*
                     LISTEN      449/named
tcp6       0      0 :::22                   :::*
                     LISTEN      564/sshd
tcp6       0      0 :::25                   :::*
                     LISTEN      793/master
tcp6       0      0 ::1:953                 :::*
                     LISTEN      449/named
tcp6       0      0 :::9090                 :::*
                     LISTEN      1/init
udp        0      0 192.168.122.52:53       0.0.0.0:*
                              449/named
udp        0      0 127.0.0.1:53            0.0.0.0:*
                              449/named
```

```
udp        0      0 0.0.0.0:68              0.0.0.0:*
                            410/dhclient
udp6       0      0 :::53                   :::*
                                 449/named
```

Systeme 2:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
           State       PID/Program name
tcp        0      0 192.168.122.51:27017    0.0.0.0:*
                 LISTEN      460/named
tcp        0      0 127.0.0.1:27017         0.0.0.0:*
                 LISTEN      460/named
tcp        0      0 0.0.0.0:3306            0.0.0.0:*
                 LISTEN      469/vsftpd
tcp        0      0 0.0.0.0:8080            0.0.0.0:*
                 LISTEN      673/mysqld
tcp        0      0 0.0.0.0:80              0.0.0.0:*
                 LISTEN      810/master
tcp        0      0 0.0.0.0:21              0.0.0.0:*
                 LISTEN      523/nginx: master p
tcp        0      0 0.0.0.0:25              0.0.0.0:*
                 LISTEN      489/sshd
tcp        0      0 127.0.0.1:953           0.0.0.0:*
                 LISTEN      460/named
tcp        0      0 0.0.0.0:3389            0.0.0.0:*
                 LISTEN      461/mongod
tcp6       0      0 :::27017                :::*
                     LISTEN      460/named
tcp6       0      0 :::80                   :::*
                     LISTEN      810/master
tcp6       0      0 :::21                   :::*
                     LISTEN      523/nginx: master p
tcp6       0      0 :::22                   :::*
                     LISTEN      568/apache2
tcp6       0      0 :::25                   :::*
                     LISTEN      489/sshd
tcp6       0      0 ::1:953                 :::*
                     LISTEN      460/named
tcp6       0      0 :::9090                 :::*
                     LISTEN      1/init
udp        0      0 192.168.122.51:27017    0.0.0.0:*
                            460/named
udp        0      0 127.0.0.1:27017         0.0.0.0:*
                            460/named
udp        0      0 0.0.0.0:68              0.0.0.0:*
                            483/dhclient
```

```
udp6       0      0 :::27017                :::*
                                460/named
```

Systeme 3:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
           State       PID/Program name
tcp        0      0 0.0.0.0:27017           0.0.0.0:*
                 LISTEN      390/mongod
tcp        0      0 0.0.0.0:3306            0.0.0.0:*
                 LISTEN      595/mysqld
tcp        0      0 0.0.0.0:80              0.0.0.0:*
                 LISTEN      437/nginx: master p
tcp        0      0 192.168.122.50:53       0.0.0.0:*
                 LISTEN      385/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*
                 LISTEN      385/named
tcp        0      0 0.0.0.0:21              0.0.0.0:*
                 LISTEN      402/vsftpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*
                 LISTEN      799/sshd
tcp        0      0 0.0.0.0:25              0.0.0.0:*
                 LISTEN      699/master
tcp        0      0 127.0.0.1:953           0.0.0.0:*
                 LISTEN      385/named
tcp6       0      0 :::8080                 :::*
                     LISTEN      706/apache2
tcp6       0      0 :::53                   :::*
                     LISTEN      385/named
tcp6       0      0 :::22                   :::*
                     LISTEN      799/sshd
tcp6       0      0 :::25                   :::*
                     LISTEN      699/master
tcp6       0      0 ::1:953                 :::*
                     LISTEN      385/named
udp        0      0 192.168.122.50:53       0.0.0.0:*
                           385/named
udp        0      0 127.0.0.1:53            0.0.0.0:*
                           385/named
udp        0      0 0.0.0.0:68              0.0.0.0:*
                           483/dhclient
udp6       0      0 :::53                   :::*
                           385/named
```

Systeme 4:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
```

```
           State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*
                  LISTEN      498/nginx -g daemon
tcp        0      0 192.168.122.148:53      0.0.0.0:*
                  LISTEN      427/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*
                  LISTEN      427/named
tcp        0      0 0.0.0.0:21              0.0.0.0:*
                  LISTEN      483/vsftpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*
                  LISTEN      426/sshd
tcp        0      0 0.0.0.0:25              0.0.0.0:*
                  LISTEN      1243/master
tcp        0      0 127.0.0.1:953           0.0.0.0:*
                  LISTEN      427/named
tcp        0      0 0.0.0.0:27017           0.0.0.0:*
                  LISTEN      446/mongod
tcp        0      0 0.0.0.0:3306            0.0.0.0:*
                  LISTEN      940/mysqld
tcp6       0      0 :::8080                 :::*
                     LISTEN      508/apache2
tcp6       0      0 :::80                   :::*
                     LISTEN      498/nginx -g daemon
tcp6       0      0 :::53                   :::*
                     LISTEN      427/named
tcp6       0      0 :::22                   :::*
                     LISTEN      426/sshd
tcp6       0      0 :::25                   :::*
                     LISTEN      1243/master
tcp6       0      0 ::1:953                 :::*
                     LISTEN      427/named
udp        0      0 192.168.122.148:53      0.0.0.0:*
                               427/named
udp        0      0 127.0.0.1:53            0.0.0.0:*
                               427/named
udp        0      0 0.0.0.0:68              0.0.0.0:*
                               402/dhclient
udp        0      0 0.0.0.0:43183           0.0.0.0:*
                               402/dhclient
udp6       0      0 :::53                   :::*
                                  427/named
udp6       0      0 :::58227                :::*
                                  402/dhclient
```

B. Konfigurationsdateien und Quellcode

1 VSFTPD

vsftpd.conf

```
# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This
   sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of
   vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea
   of vsftpd's
# capabilities.
#
#
# Run standalone?  vsftpd can run either from an inetd or as a
   standalone
# daemon started from an initscript.
listen=YES
#
# This directive enables listening on IPv6 sockets. By default,
    listening
# on the IPv6 "any" address (::) will accept connections from
   both IPv6
# and IPv4 clients. It is not necessary to listen on *both*
   IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen
    on specific
# addresses) then you must run two copies of vsftpd with two
   configuration
# files.
#listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
```

```
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change
    this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload
    files. This only
# has an effect if the above global write enable is activated.
    Also, you will
# obviously need to create a directory writable by the FTP user
    .
#anon_upload_enable=YES
#
# Uncomment this if you want the anonymous FTP user to be able
    to create
# new directories.
#anon_mkdir_write_enable=YES
#
# Activate directory messages - messages given to remote users
    when they
# go into a certain directory.
dirmessage_enable=YES
#
# If enabled, vsftpd will display directory listings with the
    time
# in your local time zone. The default is to display GMT.
    The
# times returned by the MDTM FTP command are also affected by
    this
# option.
use_localtime=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#
# Make sure PORT transfer connections originate from port 20 (
    ftp-data).
connect_from_port_20=YES
#
# If you want, you can arrange for uploaded anonymous files to
    be owned by
```

```
# a different user. Note! Using "root" for uploaded files is
   not
# recommended!
#chown_uploads=YES
#chown_username=whoever
#
# You may override where the log file goes if you like. The
   default is shown
# below.
#xferlog_file=/var/log/vsftpd.log
#
# If you want, you can have your log file in standard ftpd
   xferlog format.
# Note that the default log file location is /var/log/xferlog
   in this case.
#xferlog_std_format=YES
#
# You may change the default value for timing out an idle
   session.
#idle_session_timeout=600
#
# You may change the default value for timing out a data
   connection.
#data_connection_timeout=120
#
# It is recommended that you define on your system a unique
   user which the
# ftp server can use as a totally isolated and unprivileged
   user.
#nopriv_user=ftpsecure
#
# Enable this and the server will recognise asynchronous ABOR
   requests. Not
# recommended for security (the code is non-trivial). Not
   enabling it,
# however, may confuse older FTP clients.
#async_abor_enable=YES
#
# By default the server will pretend to allow ASCII mode but in
    fact ignore
# the request. Turn on the below options to have the server
   actually do ASCII
# mangling on files when in ASCII mode.
# Beware that on some FTP servers, ASCII support allows a
   denial of service
# attack (DoS) via the command "SIZE /big/file" in ASCII mode.
   vsftpd
```

```
# predicted this attack and has always been safe, reporting the
    size of the
# raw file.
# ASCII mangling is a horrible feature of the protocol.
#ascii_upload_enable=YES
#ascii_download_enable=YES
#
# You may fully customise the login banner string:
ftpd_banner=Welcome to Fake service.
#
# You may specify a file of disallowed anonymous e-mail
   addresses. Apparently
# useful for combatting certain DoS attacks.
#deny_email_enable=YES
# (default follows)
#banned_email_file=/etc/vsftpd.banned_emails
#
# You may restrict local users to their home directories.  See
   the FAQ for
# the possible risks in this before using chroot_local_user or
# chroot_list_enable below.
#chroot_local_user=YES
#
# You may specify an explicit list of local users to chroot()
   to their home
# directory. If chroot_local_user is YES, then this list
   becomes a list of
# users to NOT chroot().
# (Warning! chroot'ing can be very dangerous. If using chroot,
   make sure that
# the user does not have write access to the top level
   directory within the
# chroot)
chroot_local_user=YES
#chroot_list_enable=YES
# (default follows)
#chroot_list_file=/etc/vsftpd.chroot_list
#
# You may activate the "-R" option to the builtin ls. This is
   disabled by
# default to avoid remote users being able to cause excessive I
   /O on large
# sites. However, some broken FTP clients such as "ncftp" and "
   mirror" assume
# the presence of the "-R" option, so there is a strong case
   for enabling it.
#ls_recurse_enable=YES
```

```
#
# Customization
#
# Some of vsftpd's settings don't fit the filesystem layout by
# default.
#
# This option should be the name of a directory which is empty.
    Also, the
# directory should not be writable by the ftp user. This
   directory is used
# as a secure chroot() jail at times vsftpd does not require
   filesystem
# access.
secure_chroot_dir=/var/run/vsftpd/empty
#
# This string is the name of the PAM service vsftpd will use.
pam_service_name=vsftpd
#
# This option specifies the location of the RSA certificate to
   use for SSL
# encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=NO

#
# Uncomment this to indicate that vsftpd use a utf8 filesystem.
#utf8_filesystem=YES
```

2 OpenSSH

version.h

```
/* $OpenBSD: version.h,v 1.78 2016/12/19 04:55:51 djm Exp $ */

#define SSH_VERSION     "Fake Service_2.4"

#define SSH_PORTABLE    "p1"
#define SSH_RELEASE     SSH_VERSION SSH_PORTABLE
```

3 Postfix

main.cf

```
# See /usr/share/postfix/main.cf.dist for a commented, more
   complete version
```

```
# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname Fake Service (Fake OS 4.2)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# See http://www.postfix.org/COMPATIBILITY_README.html --
   default to 2 on
# fresh installs.
compatibility_level = 2

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${data_directory}/
   smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/
   smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc
   package for
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks
   permit_sasl_authenticated defer_unauth_destination
myhostname = debian
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = $myhostname, debian, localhost.localdomain,
   localhost
relayhost =
mynetworks = 192.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
```

```
inet_interfaces = all
default_transport = error
relay_transport = error
inet_protocols = all
virtual_alias_maps = hash:/etc/postfix/virtual
```

4 BIND9

named.conf.options

```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers
           you want
        // to talk to, you may need to fix the firewall to
           allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id
           /800113

        // If your ISP provided one or more IP addresses for
           stable
        // nameservers, you probably want to use them as
           forwarders.
        // Uncomment the following block, and insert the
           addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================

        // If BIND logs error messages about the root key being
            expired,
        // you will need to update your keys.  See https://www.
           isc.org/bind-keys
        //========================================================================

        dnssec-validation auto;

        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };
        version "Fake␣Service␣2.4";

};
```

5 Apache

apache2.conf

```
# This is the main Apache server configuration file.  It
    contains the
# configuration directives that give the server its
    instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed
    information about
# the directives and /usr/share/doc/apache2/README.Debian about
     Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite
    different to
# upstream's suggested way to configure the web server. This is
     because Debian's
# default Apache2 installation attempts to make adding and
    removing modules,
# virtual hosts, and extra configuration directives as flexible
     as possible, in
# order to make automating the changes and administering the
    server as easy as
# possible.

# It is split into several files forming the configuration
    hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
#       /etc/apache2/
#       |-- apache2.conf
#       |       `--  ports.conf
#       |-- mods-enabled
#       |       |-- *.load
#       |       `-- *.conf
#       |-- conf-enabled
#       |       `-- *.conf
#       `-- sites-enabled
#               `-- *.conf
#
#
# * apache2.conf is the main configuration file (this file). It
     puts the pieces
```

```
#   together by including all remaining configuration files
    when starting up the
#   web server.
#
# * ports.conf is always included from the main configuration
    file. It is
#   supposed to determine listening ports for incoming
    connections which can be
#   customized anytime.
#
# * Configuration files in the mods-enabled/, conf-enabled/ and
    sites-enabled/
#   directories contain particular configuration snippets which
    manage modules,
#   global configuration fragments, or virtual host
    configurations,
#   respectively.
#
#   They are activated by symlinking available configuration
    files from their
#   respective *-available/ counterparts. These should be
    managed by using our
#   helpers a2enmod/a2dismod, a2ensite/a2dissite and a2enconf/
    a2disconf. See
#   their respective man pages for detailed information.
#
# * The binary is called apache2. Due to the use of environment
    variables, in
#   the default configuration, apache2 needs to be started/
    stopped with
#   /etc/init.d/apache2 or apache2ctl. Calling /usr/bin/apache2
    directly will not
#   work with the default configuration.


# Global configuration
#

#
# ServerRoot: The top of the directory tree under which the
    server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise
    network)
# mounted filesystem then please read the Mutex documentation (
    available
```

```
# at <URL:http://httpd.apache.org/docs/2.4/mod/core.html#mutex
   >);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
#ServerRoot "/etc/apache2"

#
# The accept serialization lock file MUST BE STORED ON A LOCAL
   DISK.
#
#Mutex file:${APACHE_LOCK_DIR} default

#
# The directory where shm and other runtime files will be
   stored.
#

DefaultRuntimeDir ${APACHE_RUN_DIR}

#
# PidFile: The file in which the server should record its
   process
# identification number when it starts.
# This needs to be set in /etc/apache2/envvars
#
PidFile ${APACHE_PID_FILE}

#
# Timeout: The number of seconds before receives and sends time
    out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (
   more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an
   unlimited amount.
```

```
# We recommend you leave this number high, for maximum
   performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next
   request from the
# same client on the same connection.
#
KeepAliveTimeout 5


# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

#
# HostnameLookups: Log the names of clients or just their IP
   addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net
    if people
# had to knowingly turn this feature on, since enabling it
   means that
# each client request will result in AT LEAST one lookup
   request to the
# nameserver.
#
HostnameLookups Off

# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <
   VirtualHost>
# container, error messages relating to that virtual host will
   be
# logged here.  If you *do* define an error logfile for a <
   VirtualHost>
# container, that host's errors will be logged there and not
   here.
#
ErrorLog ${APACHE_LOG_DIR}/error.log

#
# LogLevel: Control the severity of messages logged to the
   error_log.
```

```
# Available values: trace8, ..., trace1, debug, info, notice,
   warn,
# error, crit, alert, emerg.
# It is also possible to configure the log level for particular
    modules, e.g.
# "LogLevel info ssl:warn"
#
LogLevel warn

# Include module configuration:
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

# Include list of ports to listen on
Include ports.conf


# Sets the default security model of the Apache2 HTTPD server.
   It does
# not allow access to the root filesystem outside of /usr/share
    and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the
   web server. If
# your system is serving content from a sub-directory in /srv
   you must allow
# access here, or in any related virtual host.
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

#<Directory /srv/>
#       Options Indexes FollowSymLinks
#       AllowOverride None
```

```
#       Require all granted
#</Directory>



# AccessFileName: The name of the file to look for in each
   directory
# for additional configuration directives.  See also the
   AllowOverride
# directive.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess and .htpasswd files
   from being
# viewed by Web clients.
#
<FilesMatch "^\.ht">
        Require all denied
</FilesMatch>


#
# The following directives define some format nicknames for use
    with
# a CustomLog directive.
#
# These deviate from the Common Log Format definitions in that
   they use %O
# (the actual bytes sent including headers) instead of %b (the
   size of the
# requested file), because the latter makes it impossible to
   detect partial
# requests.
#
# Note that the use of %{X-Forwarded-For}i instead of %h is not
    recommended.
# Use mod_remoteip instead.
#
LogFormat "%v:%p␣%h␣%l␣%u␣%t␣\"%r\"␣%>s␣%O␣\"%{Referer}i\"␣\"%{
   User-Agent}i\"" vhost_combined
LogFormat "%h␣%l␣%u␣%t␣\"%r\"␣%>s␣%O␣\"%{Referer}i\"␣\"%{User-
   Agent}i\"" combined
LogFormat "%h␣%l␣%u␣%t␣\"%r\"␣%>s␣%O" common
LogFormat "%{Referer}i␣->␣%U" referer
```

```
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup
   files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
SecServerSignature "Fake␣Service␣2.4"
```

6 nginx

ngx_http_header_filter_module.c

```
/*
 * Copyright (C) Igor Sysoev
 * Copyright (C) Nginx, Inc.
 */


#include <ngx_config.h>
#include <ngx_core.h>
#include <ngx_http.h>
#include <nginx.h>


static ngx_int_t ngx_http_header_filter_init(ngx_conf_t *cf);
static ngx_int_t ngx_http_header_filter(ngx_http_request_t *r);


static ngx_http_module_t  ngx_http_header_filter_module_ctx = {
    NULL,                                  /* preconfiguration
       */
    ngx_http_header_filter_init,           /* postconfiguration
        */

    NULL,                                  /* create main
       configuration */
    NULL,                                  /* init main
       configuration */
```

```
    NULL,                                   /* create server
       configuration */
    NULL,                                   /* merge server
       configuration */

    NULL,                                   /* create location
       configuration */
    NULL,                                   /* merge location
       configuration */
};


ngx_module_t  ngx_http_header_filter_module = {
    NGX_MODULE_V1,
    &ngx_http_header_filter_module_ctx,     /* module context */
    NULL,                                   /* module directives
        */
    NGX_HTTP_MODULE,                        /* module type */
    NULL,                                   /* init master */
    NULL,                                   /* init module */
    NULL,                                   /* init process */
    NULL,                                   /* init thread */
    NULL,                                   /* exit thread */
    NULL,                                   /* exit process */
    NULL,                                   /* exit master */
    NGX_MODULE_V1_PADDING
};


static char ngx_http_server_string[] = "Server: Fake Service"
   CRLF;
static char ngx_http_server_full_string[] = "Server: "
   NGINX_VER CRLF;


static ngx_str_t ngx_http_status_lines[] = {

    ngx_string("200 OK"),
    ngx_string("201 Created"),
    ngx_string("202 Accepted"),
    ngx_null_string,  /* "203 Non-Authoritative Information" */
    ngx_string("204 No Content"),
    ngx_null_string,  /* "205 Reset Content" */
    ngx_string("206 Partial Content"),

    /* ngx_null_string, */  /* "207 Multi-Status" */
```

```
#define NGX_HTTP_LAST_2XX  207
#define NGX_HTTP_OFF_3XX   (NGX_HTTP_LAST_2XX - 200)

    /* ngx_null_string, */  /* "300␣Multiple␣Choices" */

    ngx_string("301␣Moved␣Permanently"),
    ngx_string("302␣Moved␣Temporarily"),
    ngx_string("303␣See␣Other"),
    ngx_string("304␣Not␣Modified"),
    ngx_null_string,  /* "305␣Use␣Proxy" */
    ngx_null_string,  /* "306␣unused" */
    ngx_string("307␣Temporary␣Redirect"),

#define NGX_HTTP_LAST_3XX  308
#define NGX_HTTP_OFF_4XX   (NGX_HTTP_LAST_3XX - 301 +
   NGX_HTTP_OFF_3XX)

    ngx_string("400␣Bad␣Request"),
    ngx_string("401␣Unauthorized"),
    ngx_string("402␣Payment␣Required"),
    ngx_string("403␣Forbidden"),
    ngx_string("404␣Not␣Found"),
    ngx_string("405␣Not␣Allowed"),
    ngx_string("406␣Not␣Acceptable"),
    ngx_null_string,  /* "407␣Proxy␣Authentication␣Required" */
    ngx_string("408␣Request␣Time-out"),
    ngx_string("409␣Conflict"),
    ngx_string("410␣Gone"),
    ngx_string("411␣Length␣Required"),
    ngx_string("412␣Precondition␣Failed"),
    ngx_string("413␣Request␣Entity␣Too␣Large"),
    ngx_string("414␣Request-URI␣Too␣Large"),
    ngx_string("415␣Unsupported␣Media␣Type"),
    ngx_string("416␣Requested␣Range␣Not␣Satisfiable"),
    ngx_null_string,  /* "417␣Expectation␣Failed" */
    ngx_null_string,  /* "418␣unused" */
    ngx_null_string,  /* "419␣unused" */
    ngx_null_string,  /* "420␣unused" */
    ngx_string("421␣Misdirected␣Request"),

    /* ngx_null_string, */  /* "422␣Unprocessable␣Entity" */
    /* ngx_null_string, */  /* "423␣Locked" */
    /* ngx_null_string, */  /* "424␣Failed␣Dependency" */

#define NGX_HTTP_LAST_4XX  422
#define NGX_HTTP_OFF_5XX   (NGX_HTTP_LAST_4XX - 400 +
   NGX_HTTP_OFF_4XX)
```

```
    ngx_string("500␣Internal␣Server␣Error"),
    ngx_string("501␣Not␣Implemented"),
    ngx_string("502␣Bad␣Gateway"),
    ngx_string("503␣Service␣Temporarily␣Unavailable"),
    ngx_string("504␣Gateway␣Time-out"),
    ngx_null_string,        /* "505␣HTTP␣Version␣Not␣Supported"
        */
    ngx_null_string,        /* "506␣Variant␣Also␣Negotiates" */
    ngx_string("507␣Insufficient␣Storage"),

    /* ngx_null_string, */  /* "508␣unused" */
    /* ngx_null_string, */  /* "509␣unused" */
    /* ngx_null_string, */  /* "510␣Not␣Extended" */

#define NGX_HTTP_LAST_5XX  508

};


ngx_http_header_out_t  ngx_http_headers_out[] = {
    { ngx_string("Server"), offsetof(ngx_http_headers_out_t,
       server) },
    { ngx_string("Date"), offsetof(ngx_http_headers_out_t, date
       ) },
    { ngx_string("Content-Length"),
                offsetof(ngx_http_headers_out_t,
                   content_length) },
    { ngx_string("Content-Encoding"),
                offsetof(ngx_http_headers_out_t,
                   content_encoding) },
    { ngx_string("Location"), offsetof(ngx_http_headers_out_t,
       location) },
    { ngx_string("Last-Modified"),
                offsetof(ngx_http_headers_out_t, last_modified
                   ) },
    { ngx_string("Accept-Ranges"),
                offsetof(ngx_http_headers_out_t, accept_ranges
                   ) },
    { ngx_string("Expires"), offsetof(ngx_http_headers_out_t,
       expires) },
    { ngx_string("Cache-Control"),
                offsetof(ngx_http_headers_out_t, cache_control
                   ) },
    { ngx_string("ETag"), offsetof(ngx_http_headers_out_t, etag
       ) },
```

```
    { ngx_null_string, 0 }
};


static ngx_int_t
ngx_http_header_filter(ngx_http_request_t *r)
{
    u_char                    *p;
    size_t                     len;
    ngx_str_t                  host, *status_line;
    ngx_buf_t                 *b;
    ngx_uint_t                 status, i, port;
    ngx_chain_t                out;
    ngx_list_part_t           *part;
    ngx_table_elt_t           *header;
    ngx_connection_t          *c;
    ngx_http_core_loc_conf_t  *clcf;
    ngx_http_core_srv_conf_t  *cscf;
    struct sockaddr_in        *sin;
#if (NGX_HAVE_INET6)
    struct sockaddr_in6       *sin6;
#endif
    u_char                     addr[NGX_SOCKADDR_STRLEN];

    if (r->header_sent) {
        return NGX_OK;
    }

    r->header_sent = 1;

    if (r != r->main) {
        return NGX_OK;
    }

    if (r->http_version < NGX_HTTP_VERSION_10) {
        return NGX_OK;
    }

    if (r->method == NGX_HTTP_HEAD) {
        r->header_only = 1;
    }

    if (r->headers_out.last_modified_time != -1) {
        if (r->headers_out.status != NGX_HTTP_OK
            && r->headers_out.status !=
               NGX_HTTP_PARTIAL_CONTENT
            && r->headers_out.status != NGX_HTTP_NOT_MODIFIED)
```

```
        {
            r->headers_out.last_modified_time = -1;
            r->headers_out.last_modified = NULL;
        }
    }

    len = sizeof("HTTP/1.x␣") - 1 + sizeof(CRLF) - 1
          /* the end of the header */
          + sizeof(CRLF) - 1;

    /* status line */

    if (r->headers_out.status_line.len) {
        len += r->headers_out.status_line.len;
        status_line = &r->headers_out.status_line;
#if (NGX_SUPPRESS_WARN)
        status = 0;
#endif

    } else {

        status = r->headers_out.status;

        if (status >= NGX_HTTP_OK
            && status < NGX_HTTP_LAST_2XX)
        {
            /* 2XX */

            if (status == NGX_HTTP_NO_CONTENT) {
                r->header_only = 1;
                ngx_str_null(&r->headers_out.content_type);
                r->headers_out.last_modified_time = -1;
                r->headers_out.last_modified = NULL;
                r->headers_out.content_length = NULL;
                r->headers_out.content_length_n = -1;
            }

            status -= NGX_HTTP_OK;
            status_line = &ngx_http_status_lines[status];
            len += ngx_http_status_lines[status].len;

        } else if (status >= NGX_HTTP_MOVED_PERMANENTLY
                   && status < NGX_HTTP_LAST_3XX)
        {
            /* 3XX */

            if (status == NGX_HTTP_NOT_MODIFIED) {
```

```
                r->header_only = 1;
            }

            status = status - NGX_HTTP_MOVED_PERMANENTLY +
               NGX_HTTP_OFF_3XX;
            status_line = &ngx_http_status_lines[status];
            len += ngx_http_status_lines[status].len;

        } else if (status >= NGX_HTTP_BAD_REQUEST
                   && status < NGX_HTTP_LAST_4XX)
        {
            /* 4XX */
            status = status - NGX_HTTP_BAD_REQUEST
                            + NGX_HTTP_OFF_4XX;

            status_line = &ngx_http_status_lines[status];
            len += ngx_http_status_lines[status].len;

        } else if (status >= NGX_HTTP_INTERNAL_SERVER_ERROR
                   && status < NGX_HTTP_LAST_5XX)
        {
            /* 5XX */
            status = status - NGX_HTTP_INTERNAL_SERVER_ERROR
                            + NGX_HTTP_OFF_5XX;

            status_line = &ngx_http_status_lines[status];
            len += ngx_http_status_lines[status].len;

        } else {
            len += NGX_INT_T_LEN + 1 /* SP */;
            status_line = NULL;
        }

        if (status_line && status_line->len == 0) {
            status = r->headers_out.status;
            len += NGX_INT_T_LEN + 1 /* SP */;
            status_line = NULL;
        }
    }

    clcf = ngx_http_get_module_loc_conf(r, ngx_http_core_module
       );

    if (r->headers_out.server == NULL) {
        len += clcf->server_tokens ? sizeof(
           ngx_http_server_full_string) - 1:
```

```
                                    sizeof(
                                       ngx_http_server_string)
                                        - 1;
}

if (r->headers_out.date == NULL) {
    len += sizeof("Date: Mon, 28 Sep 1970 06:00:00 GMT"
       CRLF) - 1;
}

if (r->headers_out.content_type.len) {
    len += sizeof("Content-Type: ") - 1
           + r->headers_out.content_type.len + 2;

    if (r->headers_out.content_type_len == r->headers_out.
       content_type.len
        && r->headers_out.charset.len)
    {
        len += sizeof("; charset=") - 1 + r->headers_out.
           charset.len;
    }
}

if (r->headers_out.content_length == NULL
    && r->headers_out.content_length_n >= 0)
{
    len += sizeof("Content-Length: ") - 1 + NGX_OFF_T_LEN +
        2;
}

if (r->headers_out.last_modified == NULL
    && r->headers_out.last_modified_time != -1)
{
    len += sizeof("Last-Modified: Mon, 28 Sep 1970 06:00:00
       GMT" CRLF) - 1;
}

c = r->connection;

if (r->headers_out.location
    && r->headers_out.location->value.len
    && r->headers_out.location->value.data[0] == '/')
{
    r->headers_out.location->hash = 0;

    if (clcf->server_name_in_redirect) {
```

```
            cscf = ngx_http_get_module_srv_conf(r,
               ngx_http_core_module);
            host = cscf->server_name;

        } else if (r->headers_in.server.len) {
            host = r->headers_in.server;

        } else {
            host.len = NGX_SOCKADDR_STRLEN;
            host.data = addr;

            if (ngx_connection_local_sockaddr(c, &host, 0) !=
               NGX_OK) {
                return NGX_ERROR;
            }
        }

        switch (c->local_sockaddr->sa_family) {

#if (NGX_HAVE_INET6)
        case AF_INET6:
            sin6 = (struct sockaddr_in6 *) c->local_sockaddr;
            port = ntohs(sin6->sin6_port);
            break;
#endif
#if (NGX_HAVE_UNIX_DOMAIN)
        case AF_UNIX:
            port = 0;
            break;
#endif
        default: /* AF_INET */
            sin = (struct sockaddr_in *) c->local_sockaddr;
            port = ntohs(sin->sin_port);
            break;
        }

        len += sizeof("Location: https://") - 1
               + host.len
               + r->headers_out.location->value.len + 2;

        if (clcf->port_in_redirect) {

#if (NGX_HTTP_SSL)
            if (c->ssl)
                port = (port == 443) ? 0 : port;
            else
#endif
```

```
            port = (port == 80) ? 0 : port;

        } else {
            port = 0;
        }

        if (port) {
            len += sizeof(":65535") - 1;
        }

    } else {
        ngx_str_null(&host);
        port = 0;
    }

    if (r->chunked) {
        len += sizeof("Transfer-Encoding: chunked" CRLF) - 1;
    }

    if (r->headers_out.status == NGX_HTTP_SWITCHING_PROTOCOLS)
       {
        len += sizeof("Connection: upgrade" CRLF) - 1;

    } else if (r->keepalive) {
        len += sizeof("Connection: keep-alive" CRLF) - 1;

        /*
         * MSIE and Opera ignore the "Keep-Alive: timeout=<N>"
            header.
         * MSIE keeps the connection alive for about 60-65
            seconds.
         * Opera keeps the connection alive very long.
         * Mozilla keeps the connection alive for N plus about
            1-10 seconds.
         * Konqueror keeps the connection alive for about N
            seconds.
         */

        if (clcf->keepalive_header) {
            len += sizeof("Keep-Alive: timeout=") - 1 +
               NGX_TIME_T_LEN + 2;
        }

    } else {
        len += sizeof("Connection: close" CRLF) - 1;
    }
```

```
#if (NGX_HTTP_GZIP)
    if (r->gzip_vary) {
        if (clcf->gzip_vary) {
            len += sizeof("Vary:␣Accept-Encoding" CRLF) - 1;

        } else {
            r->gzip_vary = 0;
        }
    }
#endif

    part = &r->headers_out.headers.part;
    header = part->elts;

    for (i = 0; /* void */; i++) {

        if (i >= part->nelts) {
            if (part->next == NULL) {
                break;
            }

            part = part->next;
            header = part->elts;
            i = 0;
        }

        if (header[i].hash == 0) {
            continue;
        }

        len += header[i].key.len + sizeof(":␣") - 1 + header[i
           ].value.len
               + sizeof(CRLF) - 1;
    }

    b = ngx_create_temp_buf(r->pool, len);
    if (b == NULL) {
        return NGX_ERROR;
    }

    /* "HTTP/1.x␣" */
    b->last = ngx_cpymem(b->last, "HTTP/1.1␣", sizeof("HTTP/1.x
       ␣") - 1);

    /* status line */
    if (status_line) {
```

```
    b->last = ngx_copy(b->last, status_line->data,
       status_line->len);

} else {
    b->last = ngx_sprintf(b->last, "%03ui␣", status);
}
*b->last++ = CR; *b->last++ = LF;

if (r->headers_out.server == NULL) {
    if (clcf->server_tokens) {
        p = (u_char *) ngx_http_server_full_string;
        len = sizeof(ngx_http_server_full_string) - 1;

    } else {
        p = (u_char *) ngx_http_server_string;
        len = sizeof(ngx_http_server_string) - 1;
    }

    b->last = ngx_cpymem(b->last, p, len);
}

if (r->headers_out.date == NULL) {
    b->last = ngx_cpymem(b->last, "Date:␣", sizeof("Date:␣"
       ) - 1);
    b->last = ngx_cpymem(b->last, ngx_cached_http_time.data
       ,
                         ngx_cached_http_time.len);

    *b->last++ = CR; *b->last++ = LF;
}

if (r->headers_out.content_type.len) {
    b->last = ngx_cpymem(b->last, "Content-Type:␣",
                         sizeof("Content-Type:␣") - 1);
    p = b->last;
    b->last = ngx_copy(b->last, r->headers_out.content_type
       .data,
                       r->headers_out.content_type.len);

    if (r->headers_out.content_type_len == r->headers_out.
       content_type.len
        && r->headers_out.charset.len)
    {
        b->last = ngx_cpymem(b->last, ";␣charset=",
                             sizeof(";␣charset=") - 1);
        b->last = ngx_copy(b->last, r->headers_out.charset.
           data,
```

```
                                r->headers_out.charset.len);

            /* update r->headers_out.content_type for possible
               logging */

            r->headers_out.content_type.len = b->last - p;
            r->headers_out.content_type.data = p;
        }

        *b->last++ = CR; *b->last++ = LF;
    }

    if (r->headers_out.content_length == NULL
        && r->headers_out.content_length_n >= 0)
    {
        b->last = ngx_sprintf(b->last, "Content-Length:␣%O"
           CRLF,
                              r->headers_out.content_length_n);
    }

    if (r->headers_out.last_modified == NULL
        && r->headers_out.last_modified_time != -1)
    {
        b->last = ngx_cpymem(b->last, "Last-Modified:␣",
                             sizeof("Last-Modified:␣") - 1);
        b->last = ngx_http_time(b->last, r->headers_out.
           last_modified_time);

        *b->last++ = CR; *b->last++ = LF;
    }

    if (host.data) {

        p = b->last + sizeof("Location:␣") - 1;

        b->last = ngx_cpymem(b->last, "Location:␣http",
                             sizeof("Location:␣http") - 1);

#if (NGX_HTTP_SSL)
        if (c->ssl) {
            *b->last++ ='s';
        }
#endif

        *b->last++ = ':'; *b->last++ = '/'; *b->last++ = '/';
        b->last = ngx_copy(b->last, host.data, host.len);
```

```
    if (port) {
        b->last = ngx_sprintf(b->last, ":%ui", port);
    }

    b->last = ngx_copy(b->last, r->headers_out.location->
       value.data,
                       r->headers_out.location->value.len);

    /* update r->headers_out.location->value for possible
       logging */

    r->headers_out.location->value.len = b->last - p;
    r->headers_out.location->value.data = p;
    ngx_str_set(&r->headers_out.location->key, "Location");

    *b->last++ = CR; *b->last++ = LF;
}

if (r->chunked) {
    b->last = ngx_cpymem(b->last, "Transfer-Encoding:␣
       chunked" CRLF,
                         sizeof("Transfer-Encoding:␣chunked
                            " CRLF) - 1);
}

if (r->headers_out.status == NGX_HTTP_SWITCHING_PROTOCOLS)
   {
    b->last = ngx_cpymem(b->last, "Connection:␣upgrade"
       CRLF,
                         sizeof("Connection:␣upgrade" CRLF)
                             - 1);

} else if (r->keepalive) {
    b->last = ngx_cpymem(b->last, "Connection:␣keep-alive"
       CRLF,
                         sizeof("Connection:␣keep-alive"
                            CRLF) - 1);

    if (clcf->keepalive_header) {
        b->last = ngx_sprintf(b->last, "Keep-Alive:␣timeout
           =%T" CRLF,
                              clcf->keepalive_header);
    }

} else {
    b->last = ngx_cpymem(b->last, "Connection:␣close" CRLF,
```

```
                                sizeof("Connection: close" CRLF) -
                                    1);
    }

#if (NGX_HTTP_GZIP)
    if (r->gzip_vary) {
        b->last = ngx_cpymem(b->last, "Vary: Accept-Encoding"
           CRLF,
                                sizeof("Vary: Accept-Encoding"
                                   CRLF) - 1);
    }
#endif

    part = &r->headers_out.headers.part;
    header = part->elts;

    for (i = 0; /* void */; i++) {

        if (i >= part->nelts) {
            if (part->next == NULL) {
                break;
            }

            part = part->next;
            header = part->elts;
            i = 0;
        }

        if (header[i].hash == 0) {
            continue;
        }

        b->last = ngx_copy(b->last, header[i].key.data, header[
           i].key.len);
        *b->last++ = ':'; *b->last++ = ' ';

        b->last = ngx_copy(b->last, header[i].value.data,
           header[i].value.len);
        *b->last++ = CR; *b->last++ = LF;
    }

    ngx_log_debug2(NGX_LOG_DEBUG_HTTP, c->log, 0,
                   "%*s", (size_t) (b->last - b->pos), b->pos);

    /* the end of HTTP header */
    *b->last++ = CR; *b->last++ = LF;
```

```
    r->header_size = b->last - b->pos;

    if (r->header_only) {
        b->last_buf = 1;
    }

    out.buf = b;
    out.next = NULL;

    return ngx_http_write_filter(r, &out);
}


static ngx_int_t
ngx_http_header_filter_init(ngx_conf_t *cf)
{
    ngx_http_top_header_filter = ngx_http_header_filter;

    return NGX_OK;
}
```

nginx.h

```
/*
 * Copyright (C) Igor Sysoev
 * Copyright (C) Nginx, Inc.
 */


#ifndef _NGINX_H_INCLUDED_
#define _NGINX_H_INCLUDED_


#define nginx_version      1010003
#define NGINX_VERSION      "2.4"
#define NGINX_VER          "Fake Service" NGINX_VERSION

#ifdef NGX_BUILD
#define NGINX_VER_BUILD    NGINX_VER " (" NGX_BUILD ")"
#else
#define NGINX_VER_BUILD    NGINX_VER
#endif

#define NGINX_VAR          "NGINX"
#define NGX_OLDPID_EXT     ".oldbin"
```

```
#endif /* _NGINX_H_INCLUDED_ */
```

Abbildungsverzeichnis

Literaturverzeichnis

[AW16] Andreas, Christian ; Walonka, Werner: *Flow-Record-Fingerprinting*, Ludwig Maximilians Universität München, Masterarbeit, November 2016

[Ber14] Bernhard, Andreas: *Netzbasierte Erkennung von Systemen und Diensten zur Verbesserung der IT-Sicherheit*, Ludwig Maximilians Universität München, Bachelorarbeit, March 2014

[Co13] Cooper, Alissa ; other: Privacy Considerations for Internet Protocols / IETF. Version: July 2013. `https://tools.ietf.org/html/rfc6973`. IETF, July 2013 (6973). – RFC. – 1–36 S.. – ISSN 2070–1721

[END11] El-Nazeer, Nazar ; Daimi, Kevin: Evaluation of Network Port Scanning Tools / udmercy. 2011. – Forschungsbericht. – The 2011 International Conference on Security and Management

[Fyo99] Fyodor: *Das Erkennen von Betriebssystemen mittels TCP/IP Stack FingerPrinting*. `https://nmap.org/nmap-fingerprinting-article-de.html`. Version: 1999. – abgerufen am 22. Januar 2019

[Han08] Hanspach, Michael: *Verbesserung von OS- und Service-Fingerprinting mittels Fuzzing*, FHDW, Masterarbeit, September 2008

[Jha14] Jhala, Nikita Y.: *Network Scanning and Vulnerability Assessment with Report Generation*, Nirma University, Major Project, May 2014

[KS14] Kaur, Rajwinder ; Singh, Gurjot: Title: Analysing Port Scanning Tools and Security Techniques. In: *International Journal of Electrical Electronics and Computer Science Engineering* 1 (2014), oct, S. 58–64

[Ner06] Nerakis, Eleftherios: *IPV6 HOST FINGERPRINT*, Naval postgraduate school, Masterarbeit, September 2006

[NRH18] Nmap-Referenz-Handbuch: *Dienst- und Versionserkennung*. `https://nmap.org/man/de/man-version-detection.html`. Version: 2018. – abgerufen am 19. November 2018

[PC04] Peikari, Cyrus ; Chuvakin, Anton A.: *Kenne deinen Feind - Fortgeschrittene Sicherheitstechniken*. O'Reilly, 2004

[Tan92] Tanenbaum, Andrew S.: *Computer-Netzwerke*. 2. Auflage. Wolfram's Fachverl., 1992

[TAR12] TARGET, HACKER: *Nessus, OpenVAS and Nexpose VS Metasploitable*. `https://hackertarget.com/nessus-openvas-nexpose-vs-metasploitable`. Version: August 2012. – abgerufen am 04. Dezember 2018

[YL06] Ylonen, Tatu ; Lonvick, Chris: The Secure Shell (SSH) Transport Layer Protocol / IETF. Version: January 2006. `https://tools.ietf.org/html/rfc4253`. IETF, January 2006 (4253). – RFC. – 1–32 S.. – ISSN 0000–0000