Fachhochschule Wedel
University of Applied Sciences

Seminar IT-Sicherheit WS 12/13

Penetration Testing

Grundlagen und Automatisierung

Eingereicht von:

Christian Dohrmann (inf9589)

Betreuer:

Prof. Dr. Gerd Beuster

Fachhochschule Wedel

Feldstraße 143

22880 Wedel

Tel: (04103) 80480

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren gab es immer wieder Nachrichten über große Datendiebstähle bei Unternehmen. Diese konnten häufig auf Grund von Sicherheitslücken in den IT-Systemen der betroffenen Unternehmen durchgeführt werden. Zu den betroffenen Unternehmen gehörten zum Beispiel Sony [7], nVidia [5] oder Hetzner [6]. Der größte Vorfall in letzter Zeit war der Hack des Sony PlayStation Networks, bei dem ca. 70 Millionen Kundendaten gestohlen werden konnten. Neben den Nachrichten zeigen auch offizielle Berichte einen starken Anstieg der Cyberkriminalität an. Laut dem Symantec Internet Security Threat Report 2011 [3] stieg die Anzahl von bösartigen Attacken um 81%. Die Verwendung von einzigartiger Malware stieg um 43% und die Häufigkeit von Web Attacken nahm um 34% zu. Dabei sind, wie man aus den Medien vermuten könnte, nicht nur Großunternehmen betroffen. In der Realität sind klein- und mittelständische Unternehmen genauso häufig von Cyberangriffen betroffen wie Großunternehmen. In dieser Zeit, in der die Cyberkriminalität immer weiter ansteigt, werden Sicherheitstests immer wichtiger. Eine Form dieser Tests ist das Penetration Testing.

2 Penetration Testing

2.1 Allgemein

Beim Penetration Testing handelt es sich um den legalen und authorisierten Versuch die Schwachstellen eines Computersystems zu finden und auszunutzen. Dadurch soll die Sicherheit des Systems gegen zukünftige Angriffe gesteigert werden. Wichtig ist, dass die Schwachstellen nicht nur theoretisch betrachtet, sondern auch durch ein „Proof of Concept“ bewiesen werden. Daher werden in der Regel auch konkrete Angriffe gegen das System durchgeführt [4]. Generell teilt sich der Penetration Test in drei Phasen auf. Zunächst gibt es eine Verhandlungsphase zwischen dem Unternehmen, welches den Penetration Test in Auftrag gibt und dem Unternehmen, das den Penetration Test durchführt. Wenn die Verhandlungen abgeschlossen sind, wird der Penetration Test durchgeführt. Zum Abschluss des Penetration Tests werden die Ergebnisse dem Auftraggeber präsentiert. In den folgenden Abschnitten werden die drei Phasen genauer beschrieben.

2.2 Verhandlung

In der Verhandlungsphase werden wie bei jedem Servicevertrag, Leistungsumfang, Zeiträume oder Kosten besprochen. Zusätzlich kommen noch Penetration Testing spezifisch Themengebiete hinzu. Diese Themengebiete werden in den nächsten Abschnitten genauer erläutert.

2.2.1 Art des Penetration Tests

Zunächst muss die Art des Penetration Tests festgelegt werden. Laut Whitaker und Newman [1] wird zwischen drei Arten unterschieden:

- **Black-Box Test:** Beim Black-Box Test bekommt der Penetration Tester keine Informationen vom Unternehmen. Das heißt er weiß nur, welches Unternehmen das Ziel sein soll und muss alle Informationen über das Unternehmenssystem selbst herausfinden.
- **White-Box Test:** Beim White-Box Test erhält der Penetration Tester alle Informationen die er benötigt. Diese umfassen dann zum Beispiel die Netzwerk-Architektur des Unternehmens oder eine Übersicht aller Betriebssysteme und den Anwendungen, die auf den einzelnen Maschinen laufen.
- **Grey-Box Test:** Beim Grey-Box Test wird der Angriff eines Angestellten simuliert. Daher bekommt der Penetration Tester einen Zugang zum internen Netzwerk mit Standard-Zugriffsrechten.

Welche Art von Test im Endeffekt durchgeführt werden soll hängt vom Unternehmen ab. Der Black-Box Test simuliert am ehesten den Fall eines bösartigen Angreifers von außerhalb, wohingegen der White-Box Test die akkuratesten Ergebnisse liefert.

2.2.2 Bereiche

Nachdem die Testart festgelegt wurde, muss bestimmt werden, welche Bereiche des Systems getestet werden sollen. So kann der Penetration Test auf das gesamte System angewendet werden oder auch nur auf Teile des Systems, zum Beispiel nur auf den Webauftritt des Unternehmens.

2.2.3 Erlaubte Techniken

Ein wichtiger Punkt der zu klären ist, sind die erlaubten Techniken. Dabei wird festgelegt, welche Techniken der Penetration Tester verwenden darf, um den Test durchzuführen. So wird zum Beispiel von vielen Unternehmen festgelegt, dass keine Denial of Service Attacken durchgeführt werden, da hierdurch der Geschäftsbetrieb massiv gestört

werden kann. Ein weiteres Beispiel ist der Einsatz von Trojaner oder Rootkits durch den Penetration Tester. Zusätzlich wird noch festgelegt, ob Log-Dateien gelöscht werden dürfen und ob der Penetration Tester Daten sammeln und vom gehackten Server entfernen darf.

2.2.4 Eingeweihte Personen

Als letzter wichtiger Punkt stehen noch die eingeweihten Personen zur Diskussion. Hierbei wird festgelegt, wer alles von dem anstehenden Penetration Test weiß. In der Regel ist es sinnvoll, dass nur wenige Personen über den Test Bescheid wissen, da sonst die Administratoren versucht sein könnten, dass bestehende System nochmal gegen Angreifer zu härten. Dadurch wird das Ergebnis dann verfälscht. Daher gibt es häufig eine Einzelperson, die als Kontaktperson fungiert. Die Kontaktperson wird dann über alle Aktionen der Penetration Tester unterrichtet.

2.3 Durchführung

Wenn die Verhandlungsphase abgeschlossen ist, kann der eigentliche Penetration Test durchgeführt werden. Ein wichtiges Element der Durchführung ist die Dokumentation. Jeder Schritt und jede Erkenntnis muss genauestens dokumentiert werden, da diese Dokumentation die Grundlage für das Ergebnis des Penetration Tests darstellt. Die Durchführung kann grob in vier Phasen unterteilt werden. Bei den vier Phasen handelt es sich um Reconnaissance, Scanning, Exploitation und Maintaining Access. Die vier Phasen werden im Folgenden genauer beschrieben [1].

2.3.1 Reconnaissance

In der Reconncaissance-Phase geht es darum, Informationen über das jeweilige Ziel zu sammeln. Dabei wird in der Regel zwischen aktiver und passiver Reconnaissance unterschieden. Bei der aktiven Reconnaissance wird direkt mit dem Ziel interagiert um an Informationen zu gelangen. Ein Beispiel dafür ist die genaue Untersuchung des Webauftritts. Dabei wird in der Regel die Website heruntergeladen, um sie dann lokal untersuchen zu können. Bei der passiven Reconnaissance werden indirekt Informationen über

das Unternehmen gesammelt. Hier greift der Penetration Tester häufig auf Informationen aus dem Internet zurück, wie zum Beispiel Jobangebote oder Newsgroup-Postings. Verglichen mit der aktiven Reconnaissance ist das Entdeckungsrisiko bei der passiven Reconnaissance wesentlich geringer.

2.3.2 Scanning

Nachdem nun Informationen über das Ziel gesammelt wurden, wird mit dem Scanning des Ziels begonnen. Dabei werden Port Scans bei den IP-Adressen des Ziels durchgeführt. Da für Systeme, Services und Applikationen bekannt ist welche Ports diese standardmäßig verwenden, kann daraus erkannt werden, welche Systeme laufen und was für Services und Applikationen aktiv sind. Im Anschluss werden die jeweiligen Versionen auf mögliche Schwachstellen untersucht.

2.3.3 Exploitation

Mit den Informationen aus der Scanning-Phase kann man nun in der Exploitation-Phase mit dem eigentlichen Angriff auf das System beginnen. Ziel ist es dabei immer Administrations-Rechte auf der Zielmaschine zu erhalten. Mit Hilfe dieser Rechte hat der Angreifer dann die uneingeschränkte Kontrolle über die Maschine und kann zum Beispiel Daten herunterladen oder eigene Programme installieren. Die Rechte werden durch die Anwendung von Exploits bei den vorher aufgedeckten Schwachstellen erhalten. Die Exploitation-Phase kommt im Prinzip dem, was man sich allgemein unter „Hacking“ vorstellt am nächsten.

2.3.4 Maintaining Access

In der letzten Phase geht es darum den Zugang zu der Maschine zu sichern. Dies geschieht in der Regel durch Backdoors, die auf der Maschine installiert werden. Diese Phase ist stark von den vorherigen Verhandlungen abhängig, da dort der Einsatz von Backdoors explizit untersagt werden kann. In diesem Fall fällt die Phase fast vollständig aus.

2.4 Ergebnis

Nach dem der Test durchgeführt worden ist, werden in der letzten Phase dem Unternehmen die Ergebnisse in Form eines Reports präsentiert. Der Report setzt sich aus den folgenden Punkt zusammen[1]:

- **Kurzdarstellung:** Die Kurzdarstellung beinhaltet kaum technische Details, sondern fasst das Ergebnis des Penetration Tests knapp zusammen und zeigt Verbesserungsmöglichkeiten auf. Dieser Teil des Berichts ist meistens für die Führungsetage des Unternehmens interessant.
- **Projektabgrenzung:** In der Projektabgrenzung werden nochmal die Rahmenbedingungen, die in der Verhandlungsphase festgelegt wurden, dargestellt. Zusätzlich werden hier auch die durchgeführten Test erwähnt.
- **Ergebnisanalyse:** Die Ergebnisanalyse ist das Herzstück des Reports. Sie beinhaltet die genaue Dokumentation der einzelnen Test, in der das genaue Vorgehen und die erhaltenen Ergebnisse genau dargestellt werden.
- **Zusammenfassung:** Die Zusammenfassung ist ähnlich aufgebaut wie die Kurzdarstellung, beinhaltet aber mehr technische Details und Empfehlungen auf technischer Ebene. Daher richtet sich diese Zusammenfassung auch eher an das technische Personal.
- **Anhang:** Im Anhang befinden sich dann noch Daten, die zu den Tests gehören, wie zum Beispiel Screenshots oder Log-Dateien.

Grundsätzlich ist es wichtig, dass der Report streng vertraulich behandelt wird, da er klar die Schwachstellen des Systems darlegt. Daher werden nur wenige Kopien erzeugt und diese je nach Medium entsprechend verschickt. Das heißt, wenn eine gedruckt Version gefordert wird, muss diese per Courier überbracht werden und bei einer digitalen Form muss der Report entsprechend verschlüsselt werden.

3 Automatisiertes Penetration Testing

Die nachfolgenden Abschnitte beschäftigen sich mit der Automatisierung von Penetration Tests. Dabei werden zunächst die Vorteile einer Automatisierung aufgezeigt und anschließend werden kurz frühere Ansätze präsentiert. Danach wird ein aktueller Ansatz genauer vorgestellt.

3.1 Allgemein

Wie das vorherige Kapitel gezeigt hat, ist die Durchführung eines Penetration Tests relativ komplex und kann in der Regel nur von Experten durchgeführt werden. Ein Programm, das automatisierte Penetration Tests anbietet, ist am ehesten mit einem Virenscanner vergleichbar. Die Entwickler des Programms implementieren die notwendigen Verfahren und der Anwender muss dann nur noch den Test laufen lassen. Dadurch bietet die Automatisierung des Verfahrens den Vorteil, dass man keinen Experten mehr benötigt und zum Beispiel ein Administrator sein eigenes System testen kann, ohne auf die Hilfe von Experten angewiesen zu sein. Dadurch sinken die Kosten, die ein Penetration Test verursacht, stark, wodurch die Tests häufiger durchgeführt werden können. Somit würde eine Automatisierung zu einer größeren Sicherheit des Systems führen.

Für die Durchführung eines automatisierten Penetration Tests wird zunächst ein Modell benötigt. Das Modell muss in der Lage sein, das System und seine Zustände abzubilden. Mit Hilfe dieses Modells kann dann versucht werden mögliche Attacken auf das System zu finden. Die Schwierigkeit der Automatisierung besteht nun darin ein passendes Model zu finden, das das System vollständig abbildet. Da die Evaluation möglicher Attacken vom jeweiligen Modell abhängig ist, kann dieser Schritt unter Umständen sehr komplex sein. In den nachfolgenden Abschnitten werden nun einige Modelle vorgestellt.

3.2 Frühere Ansätze

In den folgenden Abschnitten werden einige Modellarten vorgestellt, die früher verwendet wurden, um Attacken oder Systeme zu beschreiben.

3.2.1 Attack Trees

Die Idee der Attack Trees stammt von Bruce Schneier aus dem Jahre 1999 [13]. Er schlägt dabei vor, Attacken durch einen Baum darzustellen, der sich aus AND und OR Knoten zusammensetzt. Die Wurzel des Baumes stellt dabei immer das Ziel der Attacke dar. Im Penetration Testing Umfeld könnte das Ziel also das Hacken eines bestimmten Servers sein. Die Kinder eines Knotens repräsentieren Aktionen, die durchgeführt werden müssen, um den Elternknoten zu erreichen. Wenn es sich um einen AND Knoten handelt, müssen also alle Kindknoten erfüllt sein, bei einem OR Knoten hingegen reicht die Erfüllung eines Kindknotens. In Abbildung 3.1 ist ein einfacher Attack Tree dargestellt. Wie sich gezeigt hat eignen sich Attack Trees gut zum Beschreiben von Attacken. Der Nachteil ist, dass mit dieser Art von Modell keine Automatisierung möglich ist.

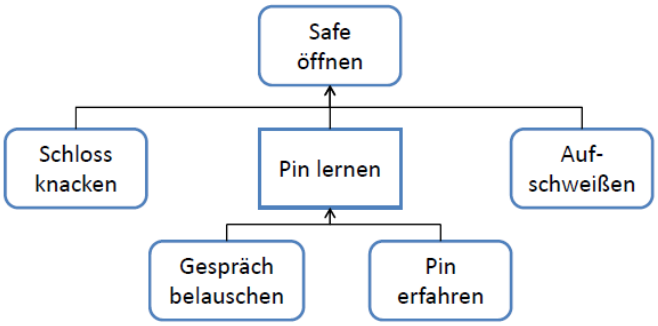

Abbildung 3.1: Attack Tree zum Öffnen eines Safes. Es gibt zwei Arten von Knoten: Knoten mit abgerundeten Ecken (OR Knoten) und Knoten mit spitzen Eckene (AND Knoten).

3.2.2 Attack Graphs

Ein weiterer Modellansatz ist die Verwendung von Attack Graphs. Die Idee stammt von Philips und Swiler aus dem Jahre 1999 [9] und sieht wie folgt aus: Der Graph setzt sich

aus Knoten zusammen, die den jeweiligen Zustand des angegriffenen Systems darstellen. Aktionen, die in dem jeweiligen Zustand ausgeführt werden können, werden durch die Kanten des Graphen dargestellt. Ziel ist es von einem Anfangszustand in einen Endzustand zu gelangen. Die Attacke dafür kann mit Hilfe von Wegfindungsalgorithmen gefunden werden. Dabei gibt der Pfad dann die Aktionsabfolge an, die der Angreifer durchführen muss, um zu seinem Ziel zu gelangen. Der Nachteil dieses Modells ist allerdings, dass die Graphen sehr schnell anwachsen. Gerade in einem echten System wird der Graph so groß, dass die Berechnung einer Attacke sehr lange dauert.

Ein weiterer Ansatz, der auf Attack Graphs basiert ist der Ansatz von Sarraute et. al [12]. Im Gegensatz zur vorherigen Methode setzt sich ihr Attack Graph aus Zielen (Goals), Gewinnen (Assets), Aktionen (Actions) und Agenten (Agents) zusammen. Der Graph besteht dabei aus abwechselnden Ebenen von Zielen und Aktionen. Dabei werden die Aktionen mit den Zielen, die sie erfüllen können, verbunden. Durch einen geschickteren Aufbau des Graphen findet kein massives Anwachsen des Graphen statt. Dadurch fällt der große Nachteil, den der vorherige Ansatz hatte, weg. Dieser Ansatz wird auch in einem aktuellen automatisierten Penetration Testing Tool verwendet.[11] Allerdings besitzt auch dieser Ansatz einen Nachteil, da das Modell nicht mit Unsicherheiten umgehen kann. In einem echten Szenario besitzt man nicht alle Informationen über das Gesamtsystem. Zwar kann man durch verschiedene Aktionen, zum Beispiel Scans, versuchen möglichst viele Informationen zu bekommen, doch dadurch entstehen große Preprocessing Kosten und die Informationen, die man durch Scans erhält, müssen auch nicht richtig sein [11].

3.3 Automatisiertes Penetration Testing mit POMDPs

Wie der vorherige Abschnitt gezeigt hat, haben die bisherigen Ansätze noch den Nachteil, dass sie nicht gut mit Unsicherheiten umgehen können. Daher wurde von Sarraute, Buffet und Hoffmann ein neuer Ansatz gewählt, bei dem das Modell mit Hilfe eines Partially Observable Markov Decision Process(POMDP) modelliert wird [10]. Die nachfolgenden Abschnitte werden daher zunächst das Prinzip der *POMDPs* erläutern. Anschließend wird dann die Funktionsweise des neuen Modells erklärt und auf ein Beispiel angewendet.

3.3.1 Partially Observable Markov Decision Process

Ein POMDP wird durch das Tupel $(S, A, O, T, \Omega, R, b_0)$ definiert. Dabei besitzen die Elemente die folgenden Bedeutungen:

- S ist die Menge der Zustände in denen sich das System befinden kann.
- A ist die Menge der Aktionen die ausgeführt werden können.
- O ist die Menge der Beobachtungen, die getätigt werden können.
- T ist die Übergangsfunktion, die angibt, mit welcher Wahrscheinlichkeit man in den Zustand $s' \in S$ gelangt wenn man die Aktion $a \in A$ im Zustand $s \in S$ anwendet. $T(s, a, s') = Pr(s'|s, a)$
- Ω ist die Beobachtungsfunktion, die angibt, mit welcher Wahrscheinlichkeit man die Beobachtung $o \in O$ wahrnimmt, wenn man die Aktion $a \in A$ ausführt und in den Zustand $s' \in S$ wechselt. $\Omega(s', a, o) = Pr(o|s', a)$
- R ist die Belohnungsfunktion, die angibt, wie groß die Belohnung ist, wenn man die Aktion $a \in A$ im Zustand $s \in S$ anwendet. $R(s, a)$
- b_0 ist die anfängliche Wahrscheinlichkeitsverteilung über die Zustände in S.

Die Unsicherheit wird bei diesem Modell mit Hilfe der Wahrscheinlichkeiten berücksichtigt. So befindet sich das System nicht zu jedem Zeitpunkt in einem Zustand s, sondern es wird eine Wahrscheinlichkeitsverteilung über alle Zustände angegeben. Diese Wahrscheinlichkeitsverteilung über die Zustände wird auch als Belief State bezeichnet. Wenn nun eine Aktion ausgeführt wird, verändert sich der Belief State in Abhängigkeit von der Aktion und der Beobachtung. Die Übergangsfunktion für den Belief State lautet $b' = \tau(b, a, o)$. Die einzelnen Wahrscheinlichkeiten des neuen Belief States werden wie folgt berechnet:

$$b'(s') = \frac{\Omega(s', a, o)}{Pr(o|a, b)} \sum_{s \in S} T(s, a, s') b(s)$$

mit $Pr(o|a, b) = \sum_{s, s' \in S} \Omega(s', a, o) T(s, a, s') b(s)$ (Normalisierung)

Auf Grund der Belief States kann man auch von einem Belief Markov Decision Process sprechen, der durch das Tupel (B, A, τ, r) definiert ist. Die Elemente haben die folgende Bedeutung:

- B ist die Menge der Belief States, die in dem POMDP existieren
- A ist die Menge der Aktionen des POMDPs
- τ ist die Übergangsfunktion des Belief States $\tau(b, a, o)$
- r ist die neue Belohnungsfunktion $r(b, a) = \sum_{s \in S} b(s)R(s, a)$

Das Ziel bei der Berechnung eines POMDPs ist es nun, eine Aktionsabfolge zu finden, bei der in jedem Schritt k die Aktion gewählt wird, die für die zukünftigen Schritte die größte Belohnung bringt. Dabei sollen auch die vorherigen Aktionen und Beobachtungen mit in die Entscheidung einbezogen werden. Die Berechnung kann mit Hilfe des Optimalitätsprinzips nach Bellman [2] durchgeführt werden:

$$V(b) = \max_{a \in A}[r(b, a) + \gamma \sum_{o \in O} \Omega(o|b, a)V(\tau(b, a, o))]$$

Für einen endlichen Horizont kann so eine optimale Aktionsabfolge gefunden werden. In der Praxis ist eine direkte Berechnung allerdings sehr aufwendig. Daher werden in der Regel Approximationsverfahren verwendet, um die Gleichung zu lösen. Im Fall von Sarraute, Buffet und Hoffmann wurde auf SARSOP [8] zurückgegriffen.

3.3.2 Modellierung mit POMDP

Nachdem es mit den POMDPs nun eine Möglichkeit gibt auch Unsicherheiten zu berücksichtigen, kann damit begonnen werden, das Verfahren auf das Penetration Testing anzuwenden. Dabei sieht die Zuordnung wie folgt aus:

- **Zustände**
  Die Zustandsmenge beinhaltet alle Zustände, die das System beziehungsweise ein einzelner Computer annehmen kann. Mit Hilfe des Zustands wird die Konfiguration des Computers beschrieben. So gibt der Zustand zum Beispiel an, welches Betriebssystem der Computer besitzt, welche Ports offen sind und welche Software auf dem Computer installiert ist und ob diese durch Exploits angegriffen

werden kann. Zusätzlich muss durch den Zustand auch abgebildet werden, ob der Computer bereits erfolgreich gehackt worden ist.

- **Aktionen**
  Die Aktionsmenge umfasst zwei Arten von Aktionen nämlich Tests und Exploits. Die Tests dienen dazu Informationen über einen Computer zu erhalten. Wichtig ist, dass durch die Anwendung von Tests der Zustand des Systems, beziehungsweise der Belief State nicht verändert wird. Beispiele für solche Tests sind Portscans oder OS-Detections. Die Exploits hingegen dienen dazu, Zugriff auf einen Computer zu erhalten oder den Zugriff auf einem Computer zu vergrößern. Im Gegensatz zun den Tests verändern die Exploits den Belief State.

- **Beobachtungen**
  Die Menge der Beobachtungen hängt mit den Aktionen zusammen, da sie sich aus den möglichen Ergebnissen der Tests und Exploits zusammensetzt.

- **Belohnung**
  Die Belohnungsfunktion berücksichtigt drei Aspekte bei der Vergabe von Belohnungen für Aktionen in einem Belief State. Zum einen muss der Wert des attackierten Computers berücksichtigt werden. Gegen den Wert des Computers werden dann die benötigte Zeit sowie das Entdeckungsrisiko als Kosten gegen gerechnet.

Durch diese Zuordnung ist es möglich ein gutes Modell für den Penetration Test zur Verfügung zu stellen, in dem auch Unsicherheiten berücksichtigt werden. Die Zustände, Tests, Exploits und Beobachtungen können für die Modellerstellung zum Beispiel aus Datenbanken oder Tools extrahiert werden.
Der schwierige Teil der Modellerstellung ist es, den anfänglichen Belief State festzulegen. Die Schwierigkeit besteht darin, aus dem letzten bekannten Belief States des Systems einen neuen Belief State zu folgern, da hier Faktoren wie die vergangene Zeit oder Aktualisierungszyklen des Systems berücksichtigt werden müssen. Generell kann man aber sagen, dass sich der Belief State leicht bestimmen lässt, wenn der letzte Test erst vor kurzer Zeit oder schon vor langer Zeit statt fand. In diesen Fällen kann man davon ausgehen, dass sich kaum etwas verändert hat (kurzer Zeitraum) oder dass das System komplett überholt worden ist und somit ein allgemeiner Belief State am effektivsten ist (langer Zeitraum). Ein größeres Problem stellt die Bestimmung des Belief States dar, wenn man sich zwischen diesen Extremen bewegt. Hier hat man eine sehr große Un-

sicherheit, wodurch die Bestimmung sich stark erschwert. Daher spricht man bei der Bestimmung des Belief States auch von einem Easy-Hard-Easy-Problem.[10]

Beispiel

Zur Verdeutlichung der Modellerstellung wird nun ein kleines Beispiel vorgestellt. Dabei handelt es sich um eine vereinfachte Variante des Beispiels von Sarraute et. al [10]. In diesem Beispiel wird eine Maschine attackiert. Die Zustands-, Aktions- und Beobachtungsmengen sehen wie folgt aus:

Zustände (S)	Aktionen (A)	Beobachtungen (O)
terminal	Probe-M0-p445	Success
M0-win2003	Exploit-M0-win2003-SMB	Failure
M0-win2003-p445		Open-Port
M0-win2003-p445-SMB		Closed-Port
M0-win2003-p445-SMB-vuln		
M0-win2003-p445-SMB-agent		
M0-winXPsp2		
M0-winXPsp2-p445		
M0-winXPsp2-p445-SMB		
M0-winXPsp2-p445-SMB-vuln		
M0-winXPsp2-p445-SMB-agent		

Tabelle 3.1: Zustands-, Aktions- und Beobachtungsmengen des Beispiels

Anhand der Zustände wird deutlich, dass die Maschine M0 zwei verschiedene Betriebssysteme aufweisen kann (Win2003, WinXPsp2). Zusätzlich wird mit den Zuständen angezeigt, ob der Port 445 offen ist (p445) und ob ein Sambaserver (SMB) läuft. Sollte dies der Fall sein, gibt es noch die Zustände, in denen der Server angreifbar ist (vuln) und einen Zustand, der angibt, dass der Angreifer Zugriff auf die Maschine hat (agent). Bei den Aktionen haben wir einen Port-Test (Probe-M0-p445) und einen Exploit für einen Sambaserver auf einem Windows 2003 Server (Exploit-M0-win2003-SMB). Die Beobachtungsmenge setzt sich aus den entsprechenden Beobachtungen zu den Aktionen zusammen. In diesem Fall gehören die Beobachtungen „Success“ und „Failure“ zum Exploit und „Open-Port“ und „Closed-Port“ zum Port-Test. Zur Vereinfachung des Beispiels ergeben die Übergangs- und die Beoachtungsfunktion immer eine Wahrscheinlichkeit von 1. Der anfängliche Belief State sieht wie folgt aus: $b_0 = (0, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0)$ Das bedeutet der Angreifer weiß zum aktuellen Zeitpunkt nur, dass auf der Maschine M0 Windows XP mit Service Pack 2 oder Windows Server 2003 läuft und er auf die

Maschine M0 keinen Zugriff hat. Wenn nun die Aktion „Probe-M0-p445“ angewendet wird, erhält der Angreifer je nach Beobachtung einen neuen Belief State. Dieser ist der Tabelle 3.2 zu entnehmen. Wichtig ist, dass sich durch den Test der Belief State b_0 nicht

Anfang	$b_0 = (0, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0)$
Closed-Port	$b = (0, \frac{1}{2}, 0, 0, 0, 0, \frac{1}{2}, 0, 0, 0, 0)$
Open-Port	$b = (0, 0, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, 0, 0, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, 0)$

Tabelle 3.2: Anwendung des Port-Tests auf den Belief State b_0

geändert hat, sondern der Angreifer nur Informationen über den Belief State bekommen hat. Ein Veränderung tritt erst ein, wenn ein Exploit durchgeführt wird. Wenn der Angreifer also nach dem Portscan den Samba-Exploit durchführt, verändert sich der Belief State. Die Veränderung ist in der Tabelle 3.3 zu sehen. Das Ergebnis der Durchführung

Anfang	$b_0 = (0, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0)$
Success	$b_1 = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)$
Failure	$b_1 = (0, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, 0, 0, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, 0)$

Tabelle 3.3: Anwendung des Samba-Exploits auf die Maschine M0 und die damit verbundene Änderung des Belief States

lässt sich so interpretieren, dass der Angreifer im Erfolgsfall (Success) genau weiß, dass er sich im Zustand „M0-win2003-p445-SMB-agent“ befindet und es damit geschafft hat Zugriff auf die Maschine M0 zu erhalten. Im Fall eines Misserfolgs (Failure) kann er den Zustand „M0-win2003-p445-SMB-vuln“ ausschließen, da sonst der Angriff erfolgreich gewesen wäre. Damit verteilt sich die Wahrscheinlichkeit nur noch auf die verbleibenden sieben Zustände, die im Belief State b_0 eine Wahrscheinlichkeit besaßen.

3.3.3 Ergebnisse

Sarraute et al [10] haben für ihren Artikel mehrere Versuche durchgeführt, in denen Angriffe auf Systeme mit verschiedenen Parametern simuliert wurden. Die Parameter waren dabei die vergangene Zeit seit dem letzten Penetration Test, die Anzahl der zur Verfügung stehenden Exploits sowie die Anzahl an Computern in dem attackierten System. Bei der Durchführung der Tests hat sich allerdings gezeigt, dass das Verfahren bei mehr als sieben Computern im System an seine Grenzen stößt, da die Berechnung zu komplex wird. In Figur 3.2 sieht man die Ergebnisse der einzelnen Tests. In den Diagrammen (a) und (b) kann man anhand des Kurvenverlaufs das Easy-Hard-Easy-Problem bei

der Abschätzung des Belief States erkennen. Bei den Diagrammen (c) und (d) hingegen sieht man eher einen linearen Anstieg der Laufzeit, wenn die Anzahl der Exploits oder der Maschinen steigt.

Ein weiteres Ergebnis der Versuche ist, dass das Verfahren das Verhalten eines „richtigen“ Hackers nachahmt. Das heißt es werden nur dann Tests durchgeführt, wenn es wirklich notwendig ist. Dadurch werden viele unnötige Aktionen eingespart. Ein Beispiel für dies Verhalten ist das folgende:
Der Angreifer kennt vier verschiedene Exploits: einen SSH Exploit (OpenBSD, Port 22), einen wuftpd Exploit (Linux, Port 21), einen IIS Exploit (Windows, Port 80) und einen Apache Exploit (Linux, Port 80). Zusätzlich geht der Angreifer davon aus, dass die Wahrscheinlichkeit für Windows als Betriebssystem höher ist als für die anderen Betriebssysteme. Das Vorgehen des Angreifers sieht nun so aus, dass er zunächst prüft, ob der Port 80 offen ist. Wenn dies der Fall ist, wird er den IIS Exploit durchführen, da hier die Erfolgschance beziehungsweise die erwartete Belohnung am größten ist. Wenn der Exploit erfolgreich ist, kann der Angreifer aufhören, ansonsten führt er den Apache Exploit aus, da er ja bereits weiß, dass Port 80 offen ist. Bei einem Misserfolg greift er auf die verbleibenden Exploits zurück [10].

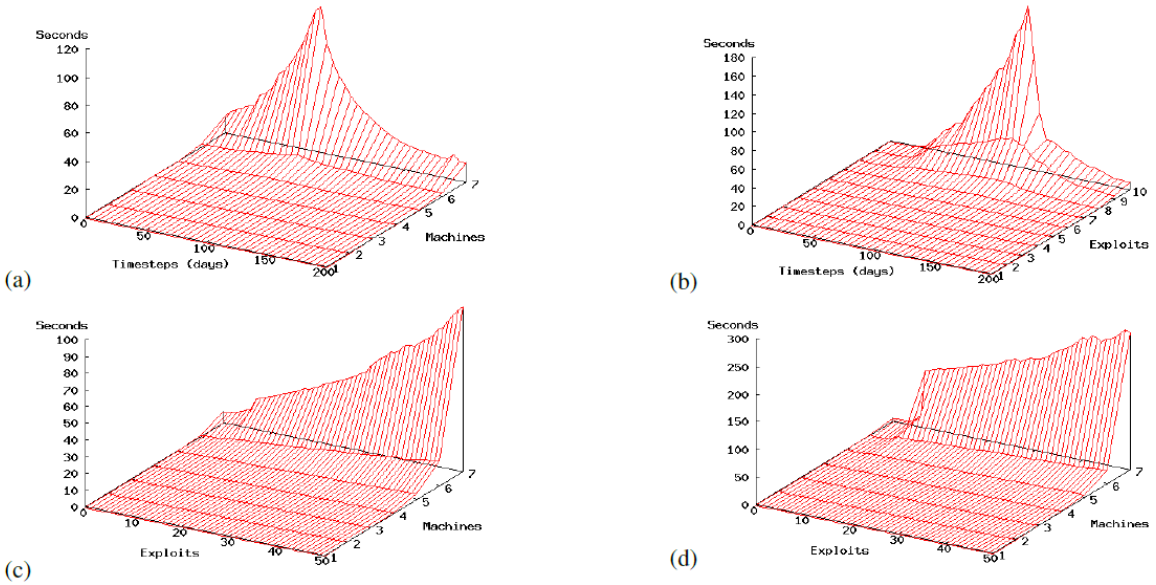

Abbildung 3.2: Laufzeit des POMDP Solvers: (a) Zeitspanne vs. Anzahl der Maschinen (b) Zeitspanne vs. Anzahl der Exploits (c) Anzahl der Exploits vs. Anzahl der Maschinen mit Zeitspanne von 10 Tagen (d) Anzahl der Exploits vs. Anzahl der Maschinen mit Zeitspanne von 80 Tagen

4 Fazit und Ausblick

Der Einsatz von Penetration Tests ist in Anbetracht der heutigen Cyberkriminalität sehr sinnvoll, da sie dafür sorgen, dass die Systemsicherheit signifikant gesteigert wird. Allerdings zeigen die Vorfälle der letzten Zeit, dass Unternehmen immer noch nicht häufig genug auf Penetration Tests zurückgreifen, um ihre Systeme zu testen. Der Grund hierfür ist gerade im Bereich der Großunternehmen nicht klar ersichtlich, lässt aber auf ein ungenügendes Bewusstsein für die Gefahren schließen. Bei kleineren Unternehmen kann prinzipiell auch der Kosten-Nutzen-Faktor als zu gering angesehen werden.

Die Automatisierung von Penetration Tests ist durchaus sinnvoll, da hierdurch die Kosten stark reduziert werden können und die Durchführung auch keine Experten benötigt. Dadurch wird die Verwendung von Penetration Tests einfacher und diese dadurch vielleicht auch häufiger angewendet werden. Der von Sarraute et. al [10] gewählt Ansatz erscheint mir vielversprechend, da hier die meisten Nachteile, die es in vorherigen Ansätzen gab, nicht mehr vorhanden sind. Allerdings gibt es auch hier noch Probleme, die gelöst werden müssen. So ist das Modell noch in einigen Bereichen unvollständig. Zum Beispiel werden in dem bisherigen Ansatz keine Seiteneffekte, die durch Exploit auftreten können, berücksichtigt. Außerdem haben die Tests des Ansatzes gezeigt, dass es noch erhebliche Probleme bei der Skalierung bei größeren Systemen gibt.

Die angesprochenen Probleme sind im Endeffekt die Felder, in denen noch weiter geforscht werden kann. So kann das Modell weiter verfeinert werden, um noch mehr Situationen berücksichtigen zu können und damit auch bessere Ergebnisse zu erhalten. Außerdem muss für den praktischen Einsatz die Skalierbarkeit stark verbessern, da echte Systeme häufig aus mehr als 7 Maschinen bestehen. Zur weiteren Verbesserung der Performance kann auch an verbesserten Lösungsverfahren für POMDPs gearbeitet werden. Dies würde dann auch in anderen Themengebieten zu einer Verbesserung der Performance führen.

Literaturverzeichnis

[1] Andrew Whitaker, Daniel P. N.: *Penetration Testing and Network Defense*. Cisco Press, 2005

[2] Bellman, Richard: *The Theory of Dynamic Programming*. 1954

[3] Corporation, Symantec: *Internet Security Threat Report*. `http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_2011_21239364.en-us.pdf`. Version: 04 2012

[4] Engebretson, Patrick ; Broad, James (Hrsg.): *The Basics of Hacking and Penetration Testing*. Syngress Press, 2011

[5] Ernst, Nico: *Hacker veröffentlichen Anwenderdaten aus Nvidia-Forum*. `http://www.golem.de/1207/93220`. Version: Juli 2012

[6] Klass, Christian: *Hetzner empfiehlt Passwortwechsel*. `http://www.golem.de/1110/86879.html`. Version: Oktober 2011

[7] Klass, Christian: *Persönliche Daten von Millionen Sony-Kunden kopiert*. `http://www.golem.de/1104/83045.html`. Version: April 2011

[8] Kurniawati, Hanna ; Hsu, David ; Lee, Wee S.: SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. (2008)

[9] Phillips, Cynthia ; Swiler, Laura P.: A Graph-Based System for Network-Vulnerability Analysis. (1999)

[10] Sarraute, Carlos ; Buffet, Olivier ; Hoffmann, Jörg: Penetration Testing == POMDP Solving. (2011)

[11] Sarraute, Carlos ; Buffet, Olivier ; Hoffmann, Jörg: POMDPs Make Better Hackers: Accounting for Uncertainty in Penetration Testing. (2012)

[12] Sarraute, Carlos ; Futoransky, Ariel ; Notarfrancesco, Luciano ; Richarte, Gerardo: Building Computer Network Attacks. (2003)

[13] Schneier, Bruce: Attack Trees: Modeling security threats. In: *Dr. Dobb's Journal* (1999)