TCP/IP ATTACK

Av Evan Saboo – saboo@kth.se

Introduktion

Labbet går ut på att testa 3 olika nätverks attacker och lära sig vad attackeraren behöver göra för att lyckas med attackerna. I praktiken är dessa attacker inte direkt användbara eftersom man har flera lösningar för att sätta stop till dem men dessa attackera bygger på andra attacker som kan vara ännu problematiska. Man behöver känna till grunden av dessa attacker för att man ska komma på hur man stoppa dem och andra attackera som har någon koppling till dem.

TCP/IP ATTACK

Av Evan Saboo – saboo@kth.se

Uppgift 1 – ARP Cache Poisoning.

Första uppgiften går ut på att simulera en ARP cache poisoning. Första skulle man sätta upp en virtuell användare, brandvägg och attackerare. Användaren är offret som skickar TCP packet till brandväggen medan attackeraren lurar användaren att skicka paket istället till attackeraren med hjälp av ARP spoofing.
För att attackeraren ska kunna lyckas med attacken måste hen vara i samma subnät som användaren och brandväggen. Användaren fick IP adressen ”10.0.20.2/24”, brandväggen fick ”10.0.20.1/24” och ”10.0.10.1/24” för att komma åt ett annat subnät som kommer användas i senare uppgifter, och till sist fick attackeraren ”10.0.20.3/24” vilket är ingår i samma subnät som användaren och brandväggen.
Jag började först skicka packet från användaren till brandväggen genom att skriva kommandot ”ping 10.0.20.1” i användarterminalen. Samtidigt som jag pingade användes wireshark för att hålla koll på kommunikationsflödet mellan användaren och brandväggen. Man såg att användaren skickade en förfrågan till brandväggen och brandväggen skickade tillbaka ett svar för användaren ska veta att har koppling till brandväggen.
Sedan skickade jag som attackerare en ARP förfrågan till användaren med linux kommandot (*sudo netwox 33 --eth-dst ff:ff:ff:ff:ff:ff --arp-ipsrc 10.0.20.1 --arp-ipdst 10.0.20.2*) och bytte min IP adress till brandväggens IP adress så att användaren tror jag är brandväggen. När användaren såg ARP förfrågan så byttes brandväggens MAC adress (som tillhörde till ”10.0.20.1”) till attackerarens MAC adress i användarens ARP tabell. Linux kommandot betyder att en ARP broadcast förfråga ska skickas till IP adressen ”10.0.20.2” och fråga efter MAC adressen till destinationsnoden med spoofad källnod.
Efter att paketet skickades såg jag i wireshark att användaren skickade ping förfrågan varje sekund till ”10.0.20.1” men fick inget svar eftersom ping förfrågorna skickades till **attackerarens** MAC adress och inget skickades tillbaka från attackerarens nod. Jag såg också i användarens ARP tabell att brandväggens IP adress tillhörde attackerarens MAC adress. Efter flera ping förfrågor skickade användaren en egen ARP förfrågan till alla noder i subnätet (Broadcast) och frågade efter MAC adrsssen av ägaren till IP adressen ”10.0.20.1”. Användaren skickade ARP förfrågan eftersom den insåg att den inte fick tillbaka något svar efter flera försök. Brandväggen skickade tillbaka svaret som användaren ville ha och flödet återställdes till ursprungliga kommunikationstillståndet.

Uppgiften förbättrade min kunskap om hur attackeraren kan gå till vägas för att få känslig information från en användare med hjälp av ARP cache poisoning.

Uppgift 2 – ICMP Redirect Attack.

I andra uppgiften skulle man utföra en ICMP redirect attack, där alla meddelanden som skickas från offret till mottagaren omdirigeras istället till attackeraren. Till denna uppgift behövde man 4 stycken hosts; en användare (offret), en brandvägg som var en gateway mellan två subnät, en attackerare som var kopplad till samma subnät som användaren och en mottagare som var kopplad till andra subnätet.
Jag började först (som användare) med att pinga till mottagararen för att illustrera ett pågående kommunikationsflöde mellan offret och mottagaren. Sedan använde jag (som attackerare) kommandot ”netwox 33 --eth-dst ff:ff:ff:ff:ff:ff --arp-ipsrc 10.0.20.1 --arpipdst 10.0.20.2” för att utföra ARP cache poisoning på offrets ARP tabell. ARP poisoning behövde utföras eftersom användaren skulle tro att hen skickade paket till brandväggen men i värkliga fall så skickar användaren paket till attackeraren då användaren tror att attackeraren är brandväggen. Då brandväggen var en gateway mellan de två subnäten så behövde användaren skicka packet till gateway för att den ska komma till mottagaren. Sedan skrev jag kommandot (sudo netwox 86 --spoofip raw --filter "dst host 10.0.10.2" --gw 10.0.20.3 --src-ip 10.0.20.1) för att sniffa på alla paket som kom från användaren. Med kommandot skickade attackeraren också tillbaka en ”redirect reply” till användaren för att meddela hen att skicka paketet istället till attackerarens IP adress ”10.0.20.3”. Eftersom varje ny packet har destinationsmålet ”10.0.10.2” så måste attackeraren utföra ”redirect reply” för varje packet, vilket man såg tydligt i wireshark:

7047 85026.366455000 10.0.20.2 10.0.10.2 ICMP 98 Echo (ping) request id=0x01ca, seq=2811/64266, ttl=64
7048 85026.419466000 10.0.20.1 10.0.20.2 ICMP 70 Redirect (Redirect for host)
7049 85027.367362000 10.0.20.2 10.0.10.2 ICMP 98 Echo (ping) request id=0x01ca, seq=2812/64522, ttl=64
7050 85027.418780000 10.0.20.1 10.0.20.2 ICMP 70 Redirect (Redirect for host)
7051 85028.368778000 10.0.20.2 10.0.10.2 ICMP 98 Echo (ping) request id=0x01ca, seq=2813/64778, ttl=64
7052 85028.380945000 10.0.20.1 10.0.20.2 ICMP 70 Redirect (Redirect for host)

Användaren skickade också ibland ARP förfrågan och till attackeraren för att fråga efter MAC adressen till IP adressen ”10.0.20.1” och attackeraren skickade en reply som användaren behövde eftersom attackeraren sniffade på paket som kom från användaren medan hen höll konstant spoofad IP adress.

Uppgiften var enkel att utföra men det var lite svårt att förstå alla delar i processen och vad kommandon gjorde specifikt. Med lite hjälp från assistenterna fick man bättre förståelse på hur hela processen fungerade.

Uppgift 3 – TCP Session Hijacking.

I sista uppgiften skulle man som attackerare ta över ett pågående TCP kommunikation mellan servern och klienten.

För att skapa en TCP kommunikation startade jag en netcat server daemon på outside host med kommandot ”nc -l 10.0.10.2”, där outside host betedde sig som en server (host). För att koppla sig till servern från inside host använde jag kommandot ”nc 10.0.10.2 1024”, där ”10.0.10.2” är adressen till servern och ”1024” är portnumret till servern.
När klienten hade kopplat till servern så kunde klienten skicka meddelande till serven och vice versa. Man såg också i wireshark att klienten först skickade en Push & Acknowledgement Paket med sekvens- & acknowledgement nummer och sedan skickade serven tillbaka en Acknowledgement packet med klientens acknowledgement nummer och (klientens sekvensummer + längden av meddelandet) = nytt sekvensnummer, t.ex. om klienten skickar sekvensnumret 1003 med meddelandet ”hej” så skickar serven tillbaka sekvensnumret 1006 (1003 + längden av ”hej”).
Efter några observationer så var det dags för attackeraren att skicka en spoofat packet till servern för att ta över TCP sessionsförbindelsen. Som attackerare använde jag kommandot:
(sudo netwox 40 -l 10.0.20.2 -m 10.0.10.2 -o <klient port number> -p <server port number> -q <sequence number> -r <acknowledgement number> -H "<text message in hexadecimal>" -E <window size number> -j <time to live number>) -A -z)
för att skicka spoofade paketet. Source IP adressen ska vara klientens IP adress och destinations IP adressen ska vara servens IP adress, då attackeraren vill skicka en packet som ser ut som att den kommer från klienten så att serven accepterar den. Jag behövde också klientens port nummer och serverns port nummer, som kunde sniffa fram med hjälp av wireshark. När man sniffade kommunikationen mellan noderna i wareshark kunde man också se klientens sekvens- och acknowledgements nummer för att skicka spoofade paketet med nästa sekvensnummer, då man vet att acknowledgement numret blir samma eftersom det ändras bara när servern skickar ett nytt meddelande till klienten. Till sist behövde jag bara window size, time to live (ttl) från sniffandet och ett meddelande (hexadecimal) för att paketet ska vara komplett. Kommandot ”-A” säger att paket är PUSH och ”-z” är ACKNOWLEDGEMENT ([PSH, ACK]). När paketet skickades till serven såg man i wireshark att servern skickade tillbaka en ACK paket men destinations adressen var klientens IP adress och inte attackerarens IP. Meddelandet i den spoofade paketet syndes också upp i servens terminal, vilket betydde att attackeraren lyckades sno TCP förbindelsen från klienten. Jag skickade igen en likadan spoofad packet till serven men med nästa sekvens nummer och den blev också accepterad av serven.
Efter attacken försökte jag som klient att skicka ett paket till serven men den blev nekad eftersom sekvensnumret var för lågt än vad serven ville ha, man såg också i wireshark att paketet försökte skickas om flera gånger men den blev nekad varje gång. Om man som server försökte skicka meddelande till klienten så accepterades inte paketet av klienten eftersom sekvensnumret var högre än vad klienten förväntade sig, vilket skickades om flera gånger men blev ändå nekad varje gång.

Denna uppgift tog längre tid än vad man hade förväntad sig eftersom man inte visste vad man skulle göra för att ta över en session, innan man gick igenom föreläsningen som förklarade hur attacken utfördes. Det var också jobbigt att bygga upp spoofade paketet eftersom man inte visste exakt vad som behövdes för att serven ska acceptera paketet, då man trodde t.ex. att sekvens nummer räckte men man behövde också ACK nummer, ttl och window size.