Practical WLAN Security

Ulf Kargén Fredrik Hansson
Email: ulfka531,freha053@student.liu.se
Supervisor: David Byers, davby@ida.liu.se
Project Report for Information Security Course
Linköpings universitet, Sweden

Sammanfattning

Syftet med det här arbetet har varit att utveckla en praktisk Man-In-The-Middle-attack mot trådlösa klienter på NetLogon-systemet vid Linköpings Universitet. Målsättningen har varit att attacken skall ske så transparent som möjligt ur offrets synvinkel. Attacken använder DNS-spoofing och en MITM-metod med förfalskade avsändaradresser genom packet injection.

Vi har implementerat ett fungerande verktyg för DNS-spoofing i trådlösa nätverk. På grund av hårdvarubegränsningar är den praktiska användbarheten av verktyget dock något begränsad. Implementationen av MITM-verktyget visade sig, sannolikt på grund av hårdvarubegränsningar, fungera allt för otillförlitligt för att vara praktiskt användbar.

Den huvudsakliga slutsatsen av arbetet är att möjligheten att lyckas med attacker av den här typen är starkt beroende av hårdvaruutrustning och drivrutiner.

1. Introduktion

Ett av de fundamentala säkerhetsproblemen med trådlösa nätverk, som 802.11x, är att det rör sig om delade medium, dvs. alla kan höra alla. Detta är självklart ett problem vid olika former av autentisering där andra kan avlyssna t.ex. lösenord om dessa inte skyddas med någon form av kryptering. Även om kryptering används möjliggör dock det delade mediumet Man-In-The-Middle-attacker. I det här arbetet har vi valt att utveckla ett verktyg för att utföra en praktisk MITM-attack mot Linköpings Universitets NetLogon-system. NetLogon är en s.k. captive portal, som låter användare logga in med sitt LiU-ID för att få tillgång till internet via universitets nät. Ett säkerhetsproblem med systemet är att samma LiU-ID används till flera olika tjänster. Om någon lyckas utföra en attack mot NetLogon och få tillgång till andras LiU-ID kan personen få tillgång till olika personliga uppgifter, som exempelvis offrens e-post, eller utföra handlingar i någon annans namn.

Målsättningen är att göra ett verktyg som lyssnar efter användare som försöker autentisera sig mot NetLogon och kapa anslutningen till NetLogon-portalen för att utföra en MITM-attack, som loggar användarnamn och lösenord vid inloggning. Offret kommer se en inloggningsportal som ser likadan ut som NetLogon. Efter inloggningen skall inte offret märka någon skillnad mot det normala fallet, dvs. han eller hon skall kunna använda internet som vanligt.

För att vi under utvecklingsarbetet skall kunna testa verktyget utan att störa det verkliga NetLogon-systemet har vi konfigurerat en egen captive portal, som i stora drag simulerar de tekniska funktioner vi kommer attackera hos den verkliga portalen.

2. Bakgrund

2.1 Captive portals

En captive portal är en mekanism för att kräva att en användare autentiserar eller registrerar sig för att få tillgång till internet och detta används oftast i publika nätverk. Det sker genom att all trafik från en klient blockeras på något sätt tills det att användaren har öppnat en webbläsare och försökt nå en webbsida. Användaren omdirigeras då till en sida för autentisering, registrering eller liknande. När användaren korrekt har autentiserat eller registrerat sig så släpps blockering av klientens trafik och användaren kan använda internet som vanligt. Vanligtvis identifierar servern klienter genom klientens MAC-adress. Det finns flera sätt en captive portal kan fungera på, men de största skillnaderna mellan olika varianter är hur trafiken omdirigeras till captive portalens webbsida. Vanliga varianter är omdirigering via DNS eller HTTP.

Vid omdirigering med DNS så är nätverkets brandvägg konfigurerad så att alla DNS-uppslagningar, från klienter som ännu inte är autentiserade, skickas till portalens egna DNS-server. DNS-servern är konfigurerad att alltid svara med adressen till captive portalens autentiseringssida, oavsett förfrågan. När användaren försöker nå en webbsida via sin webbläsare skickas en DNS-uppslagning för att slå upp IP-adressen till webbsidan som användaren försöker nå, men istället

fås IP-adressen till autentiseringssidan och användaren dirigeras dit. När sedan användaren har autentiserat eller registrerat sig korrekt så korrigeras brandväggen så att klienten kan använda en annan DNS-server och på så vis kunna få tag i de korrekta IP-adresserna till webbsidor och använda internet.

Omdirigering via HTTP går till så att när användaren försöker nå en webbsida så skickar klienten som vanligt en DNS-uppslagning för sidan och får tillbaka webbsidans IP-adress. När klienten har IP-adressen så skickar den en HTTP-request till IP-adressen för att visa webbsidan i användarens webbläsare. Portalens brandvägg är då konfigurerat för att fånga HTTP-request och skickar dem vidare till en server som svarar klienten med en HTTP-redirect till captive portalens webbsida för autentisering eller registrering. När autentisering eller registrering är korrekt genomförd konfigureras brandväggen så att HTTP-request tillåts för den klienten och då kan användaren använda internet [1].

2.2 NetLogon

NetLogon är en captive portal vid Linköpings Universitet som kräver att en användare autentiserar sig med sitt LiU-ID för att få tillgång till universitetsnätet. NetLogon fungerar genom omdirigering via HTTP-request som beskrivits i avsnitt 2.1. När en klient är associerat med LiU-nätet och användaren försöker nå en webbsida via sin webbläsare så omdirigeras han till netlogon.liu.se. För att få fram IP-adressen så gör klienten en DNS-förfrågan om netlogon.liu.se och får ett svar med IP-adressen och blir på så vis dirigerad till Netlogons autentiseringssida.

NetLogon använder sig även av SSL, som beskrivs närmare i avsnitt 2.3, för att användarens LiU-ID och lösenord inte skall skickas i klartext över nätverket.

2.3 SSL

SSL står för Secure Socket Layer och är en säkerhetsmekanism som gör det möjligt att kryptera information som skickas mellan en klient och en server. Detta är önskvärt så att känslig information inte skickas över ett nätverk (t.ex. internet) i klartext så att obehöriga kan komma över informationen. SSL bygger på både asymmetrisk och symmetrisk kryptering för att utbyta nycklar respektive att kryptera kommunikationen. För att autentisera en part används certifikat som bland annat innehåller partens publika nycklar. Med andra ord bygger säkerheten i SSL till stor del på att man kan lita på att certifikaten är korrekta.

SSL sessionen börjar med en handskakning där klienten och servern utbyter certifikat och information. Klienten skapar utifrån informationen en hemlighet som krypteras med serverns publika nyckel innan den skickas till servern. De båda parterna räknar sedan utifrån hemligheten fram en nyckel som kan användas för att kryptera och dekryptera kommunikationen med en överenskommen symmetrisk krypteringsalgoritm.

När handskakningen är klar så har klienten och servern kommit överrens om en symmetrisk krypteringsalgoritm att använda och vilken nyckel som skall användas för krypteringen och dekrypteringen. När detta är klart startar den egentliga kommunikationen mellan de båda parterna och all kommunikation är då krypterad med den gemensamma nyckeln [2].

3. Praktisk attack mot NetLogon

3.1 Översikt av attacken

Attacken består av två huvudsteg; kapning av anslutningen till NetLogon, och själva MITM-attacken. Kapningen av anslutning åstadkoms genom s.k. DNS-spoofing. Detta innebär att ett falskt svar skickas till offret när denne gör en DNS-uppslagning av ”netlogon.liu.se”. Det falska svaret anger adressen till den attackerande datorn snarare än den verkliga adressen till NetLogon.

När offret då upprättar en anslutning till NetLogon för att autentisera sig kommer anslutningen i själva verket göras till den attackerande datorn. På den attackerande datorn finns ett program som accepterar anslutningen och vidarebefordrar kommunikationen transparent till den verkliga Netlogon-servern. Eftersom all kommunikation går genom den attackerande datorn kan denna se all information som skickas, inklusive användarnamn och lösenord. Eftersom NetLogon använder SSL för kryptering och autentisering av servern måste kommunikationen mellan offret och den attackerande datorn antingen göras utan SSL, eller med ett falskt certifikat. Kommunikationen mellan attackerare och den verkliga NetLogon-servern görs på vanligt sätt med SSL.

3.2 DNS-spoofing

När användaren gör en uppslagning med DNS skickas ett paket med förfrågan till DNS-servern. Verktyget för DNS-spoofing lyssnar hela tiden efter DNS-paket med förfrågningar om IP-adressen till ”netlogon.liu.se” och skickar så fort ett sådant paket upptäcks ett förfalskat svar. Genom att titta på avsändar- och mottagar-IP-adress, samt portnummer och ID-nummer för DNS [3] i det sända paketet kan ett svarspaket som ser ut att komma från den verkliga DNS-servern skapas. Eftersom den attackerande datorn oftast kan svara snabbare än den verkliga DNS-servern kommer det förfalskade svaret fram till offret först. När det verkliga svaret kommer ignoreras det ofta av offrets dator, eftersom det inte längre hör till en aktiv DNS-förfrågan. Det förfalskade DNS-svaret anger den attackerande datorns IP-adress i

stället för den verkliga IP-adressen till NetLogon. Därför kommer anslutningar som offret gör till NetLogon i själva verket göras till den attackerande datorn.

3.3 Man-In-The-Middle-attack mot NetLogon

Efter att användaren dirigerats om till den attackerande datorn genom DNS-spoofing fungerar MITM-verktyget som en transparent proxy, som vidarebeordrar trafiken till den verkliga NetLogon-servern. För att inloggningen skall fungera som vanligt, dvs. offret loggas in och kan använda internet efteråt, kan inte den attackerande datorn använda sin egen MAC- och IP-adress vid kommunikationen med NetLogon. Detta eftersom NetLogon använder MAC-adressen hos klienter för identifikation. Därför måste trafiken till NetLogon ske med förfalskad MAC- och IP-adress, så att den ser ut att komma från offret. Detta innebär att svarstrafiken kommer till offret, där den kastas eftersom den inte verkar höra till en existerande anslutning. Eftersom kommunikationen sker på ett trådlöst nätverk kan dock den attackerande datorn sniffa svarspaketen och på så sätt åstadkomma tvåvägskommunikation. Denna kommunikation sker genom att helt enkelt byta MAC- och IP-adress på utgående IP-paket, och sedan byta tillbaka adresserna på inkommande sniffade paket. Tunnlingen sker alltså helt transparent på IP-nivå, utan att innehållet i paketen behöver beaktas.

4. Implementation av attacken

För att implementera DNS-spoofing och MITM-attacken krävs åtkomst till det trådlösa nätverket på lägre nivå än vad operativsystemets nätverksstack vanligtvis tillåter. Dels måste man kunna få in (sniffa) paket från alla användare i nätverket, inte bara paket som är adresserade till det egna nätverkskortet. För att kunna skicka ut paket med förfalskade MAC- och IP-adresser måste man också kunna förbipassera operativsystemets nätverksstack och injicera godtyckliga paket in i nätverket.

4.1 Sniffning av paket

Ett 802.11 nätverkskort kan användas i flera olika lägen, bland annat *managed mode* och *monitor mode* [4]. Managed mode är det vanligaste att använda i trådlösa nätverk och nätverkskortet är då associerad med en av nätverkets accesspunkter. Vid paketsniffning i managed mode så sniffas bara paket som kommer från accesspunkten och inte alla paket som sänds över nätverket. För att kunna genomföra vår attack så räcker det inte, utan vi behöver sniffa alla paket som skickas, det vill säga även paket som skickas från klienter till accesspunkten. Vi använder därför läget monitor mode som medför att nätverkskortet inte är associerat med någon accesspunkt och alla paket som skickads över nätverket kan sniffas.

För att kunna behandla alla sniffade paket kan inte operativsystemets vanliga IP-stack användas, eftersom den döljer lågnivådetaljer vid nätverkskommunikation. För att kunna behandla alla paket behövs s.k. råa sockets (*raw socket*). Vi använder programbiblioteket *libpcap*, som beskrivs mer ingående i 4.3, för att förenkla paketsniffning med råa sockets.

4.2 Packet injection

För att utföra injicering av paket i nätverket använde vi programbiblioteket *libnet* (se nedan). Libnet tillhandahåller ett API för att konstruera godtyckliga paket av olika slag och sända ut dem via ett Ethernet-interface.

Drivrutinerna för trådlösa nätverkskort i Linux tillhandahåller ett simulerat Ethernet-interface, med samma MAC-adress som det trådlösa kortet, för varje trådlöst nätverkinterface. Detta är möjligt eftersom MAC-adresserna för Ethernet och 802.11-standarden har samma format. Program som använder nätverksanslutningen kan alltså skicka paket precis som på ett vanligt Ethernet-kort. Drivrutinerna placerar sedan datadelen i paketen i en 802.11-ram och skickar det över det trådlösa nätverket. Eftersom vi måste kunna konstruera paket på datalänknivå för att kunna förfalska MAC-adresser räcker inte libnets sändfunktion. Detta eftersom drivrutinerna ofta bara tillåter att man skickar ut paket med den egna MAC-adressen som avsändare genom det simulerade Ethernet-interfacet.

För att kunna skicka ut paket på datalänknivå direkt i det trådlösa nätverket krävs s.k. *packet injection*. Packet injection innebär att nätverkskortet skickar ut paket med mer eller mindre godtyckligt innehåll utan att ta hänsyn till nätverkskortets arbetsläge och dylikt. Detta innebär att man kan skicka ut paket utan att vara associerad till en accesspunkt, att man kan skicka ut paket som ser ut att komma från en accesspunkt eller från en annan användare osv. Eftersom packet injection inte är nödvändig vid vanlig användning av nätverkskortet stöds det inte av alla nätverkskort och drivrutiner. Vid implementationen av vår attack har vi använt programbiblioteket *LORCON* (se 4.3), som bl.a. möjliggör packet injection.

4.3 Programbibliotek

Vid implementationen har vi använt biblioteket libpcap [5] för sniffning av paket. Libpcap har ett förenklat API för att öppna och använda råa sockets jämfört med Linux egna nätverks-api. För konstruering av paket för injecering har vi använt biblioteket libnet [6], som tillåter enkel konstruering av flera olika pakettyper i TCP/IP-sviten, samt automatisk beräkning

av checksummor m.m. För den faktiska injiceringen i det trådlösa nätverket använder vi biblioteket LORCON [7], som också tillhandahåller funktionsanrop för att ändra arbetsläget på nätverkskortet (monitor mode, managed mode osv.).

4.4 DNS-spoofing

Vid implementationen av DNS-spoofing har vi utgått från verktyget *airpwn* [8], som lyssnar efter TCP-paket och skickar ut förfalskade svar till paket som uppfyller ett visst mönster. Airpwn kan t.ex. användas för att skicka falska svar på HTTP-GET-requests. Vi har implementerat ganska omfattande modifikationer av airpwn för att i stället kunna lyssna efter UDP-paket på port 53 (DNS) och skicka falska svarspaket. Airpwn använder biblioteket pcap, som diskuteras ovan, för att lyssna efter paket, och biblioteken libnet och LORCON för konstruktion respektive injicering av falska svarspaket. Airpwn har från början en funktion för att ange paketfilteruttryck, som inkommande paket matchas mot. Med hjälp av dessa filter är det t.ex. möjligt att specificera vilka IP-adresser attacken skall riktas mot, vilket förenklar testning av attacken. Om inget filter anges försöker programmet attackera alla användare inom nätverkskortets räckvidd. Filteruttrycken följer BPF-formatet [9].

4.5 MITM-attacken

För själva MITM-attacken mot NetLogon använder vi verktyget *webmitm* [10], som kan göra MITM-attacker både mot vanlig HTTP och mot HTTPS med falska certifikat. För att kunna använda falska MAC- och IP-adresser, som beskrivs i avsnitt 3.3, använder vi en lokal proxy som byter adresser på trafiken på ett sätt som liknar NAT. Kommunikationen fungerar alltså på så sätt att offret ansluter sig till webmitm, som sedan ansluter till en lokal proxy på samma dator. Denna proxy byter sedan MAC- och IP-adresser på utgående paket, så att de ser ut att komma från offret, och injicerar dem i nätverket. På samma sätt sniffar proxyn svarstrafiken, byter tillbaka till de adresser webmitm förväntar sig och skickar paketen till webmitm. Tunnlingen sker alltså helt transparent på IP-nivå, vilket har den stora fördelen att vi kan utnyttja operativsystemets egen TCP-stack. För att detta skall fungera vid attacker mot mer en ett offer åt gången måste man kunna hålla reda på vilken trafik som hör till vilket offer. Den princip vi valt för att uppnå detta är att låta webmitm hålla reda på alla pågående HTTP-sessioner och starta en ny instans av proxyn för varje session. Varje instans av proxyn lyssnar på en unik port, som anges av webmitm vid start av proxyn. På detta sätt kan webmitm tunnla trafik för varje användare genom en unik instans av proxyn.

5. Resultat och diskussion

Vi har implementerat merparten av den funktionalitet som beksrivs i föregående avsnitt. Vi har verifierat att attacken i teorin bör vara genomförbar, dock visade sig attacken vara svår att tillämpa i praktiken, främst p.g.a. diverse hårdvarubegränsningar i vår utrustning och den begränsade tidsramen.

5.1 Resultat

Vi har gjort en fungerande implementation av DNS-spoofing, som klarar av att sniffa efter DNS-förfrågningar och skicka ut förfalskade svar. P.g.a. problem med hårdvara/drivrutiner är den praktiska tillämpbarheten av attacken något begränsad (se diskussion). Vi har också implementerat proxydelen av MITM-attacken. Vi har verifierat att proxyn principiellt fungerar genom att utföra en fullständig HTTP-GET genom proxyn. Dvs. utförande av TCP-handskakning vid anslutning och sändande av GET samt mottagande av svar. Dock fungerar tunnlingen otillförlitligt eftersom injicering av paket ofta misslyckas. Se 5.2 för en diskussion av möjliga orsaker till problemet. P.g.a. problemen med proxyn har vi valt att inte utföra modifikationen av webmitm, som beskrivs i 4.5.

5.2 Diskussion

Även om implementationen av den fullständiga attacken inte kunde slutföras inom den givna tidsbegränsningen har vi visat att DNS-spoofing i stor skala är relativt lätt att implementera för trådlösa nätverk. Detta i sig självt är ett betydande säkerhetsproblem, som kan användas för olika attacker. Bl.a. skulle man kunna sätta upp falska varianter av olika tjänster som kräver inloggning, som exempelvis internetbanker.

Vid implementationen av DNS-spoofing stötte vi på ett antal problem med hårdvara och drivrutiner. I vår testmiljö, som använder 802.11b-standarden, fungerar DNS-spoofing fullt ut med ett av nätverkskorten vi använt. I den verkliga miljön på LiU fungerar dock inte sniffning av paket med det nätverkskortet. Vi har inte lyckats fastställa orsaken till problemet fullständigt, men sannolikt rör det sig om hårdvarubegränsningar av något slag. Med ett annat nätverkskort fungerar sniffning och injicering i båda miljöer. I det fallet hinner vår attackerande dator inte injicera det falska DNS-svaret innan det verkliga svaret kommer fram till offret. Detta beror sannolikt också på begränsningar i hårdvara. Sammanfattningsvis visade det sig alltså att DNS-spoofing kräver mycket snabbare respons än t.ex. spoofade svar på HTTP-requests, som airpwn utför. Anledningen är att HTTP-requests kräver svar från en

server på internet, medan DNS-servern ofta finns lokalt och därför har betydligt kortare svarstid.

Ett problem, som begränsar framförallt möjligheten att testa attacken mot specifika IP-adresser är att NetLogon-systemet tilldelar användare olika kanaler. Om den attackerande datorn använder en annan kanal än det tilltänkta offret kan inte DNS-spoofingverktyget sniffa DNS-förfrågningarna. Det skulle vara möjligt att delvis avhjälpa problemet genom att köra DNS-spoofing på ett separat nätverkskort och kontinuerligt cirkulera mellan alla kanaler på detta, s.k. *channel hopping*.

Som vi nämner i resultatet har vi trots långvarig felsökning inte lyckats identifiera orsaken till problemet med att injicering av paket ofta misslyckas i proxydelen. En möjlig förklaring är någon slags hårdvaruproblem p.g.a. att vi försöker injicera paket i för hög takt. Ett annat generellt problem med injicering av paket är att det är svårt att fullt ut följa 802.11-standarden. Bl.a. skall MAC-headern för 802.11 innehålla ett sekvensnummer och ett fält som beskriver hur lång tid det förväntas ta att skicka paketet [11]. Eftersom dessa värden normalt sett fylls i av drivrutinerna till nätverkskortet är det svårt att korrekt bestämma värdena vid lågnivåinjicering. Det är också lite oklart hur stor tolerans olika implentationer har mot mindre brott mot standarden som dessa.

Ett annat problem vid vidarebefordring av trafik med förfalskad IP-adress är att TCP-standarden säger att TCP-segment som skickas till en stängd port skall besvaras med ett RST-segment [12]. Vid den tunnlade kommunikationen kommer då svarstrafik som skickas till offrets riktiga IP-adress resultera i att offret skickar ut ett RST-segment till servern, vilket leder till att anslutningen avbryts. Ett möjligt sätt att avhjälpa problemet är att använda samma avsändarportnummer vid anslutningen från den attackerande datorn till NetLogon, som används vid anslutningen från offret till den attackerande datorn. Svarstrafiken kommer då komma in på en port som redan är öppen hos offret, och alltså bara kastas eftersom det har ett felaktigt sekvensnummer. Vi har dock inte kunnat verifiera teorin i praktiken p.g.a. problemen med proxyn. Eftersom Windows bryter mot TCP-standarden i det här avseendet skulle dock problemet inte uppstå vid attacker mot Windows-datorer.

Den största möjligheten till upptäckt av attacken ligger i att användaren uppmärksammar det falska certifikatet. Möjligheten till upptäckt med ett Intrusion Detection System (IDS) är begränsad. Det mest uppenbara tecknet på attacken är det falska DNS-svaret, men eftersom det skickas direkt till offret och inte går genom accesspunkten måste en IDS sniffa paket i monitor mode lokalt för att kunna se paketet. Ett annat tecken på attacken är HTTP-kommunikationen som använder NetLogons URL, men som går till en annan IP-adress än NetLogon-portalen. Eftersom denna kommunikation också går genom accesspunkten borde den också vara lättare att upptäcka för en IDS.

6. Slutsatser

Den huvudsakliga slutsatsen med arbetet är att attacker av den här typen, som kräver lågnivåaccess till hårdvara, är mycket beroende av utrustning och drivrutiner för att fungera väl i praktiken. Vi har inte kunnat verifiera att attacken fungerar i sin helhet, men givet de resultat vi fått fram skulle sannolikt attacken fungera, om hårdvaruproblemen kan lösas. En bidragande orsak till att vissa problem inte kunde lösas är också den begränsade tidsramen för projektet. Med mer tid till att undersöka orsaken till problemen skulle det sannolikt vara möjligt att komma med lösningar till dem.

Även om hela attacken ej kunnat verkställas har vi visat att DNS-spoofing i trådlösa nätverk är relativt enkelt, vilket i sig är ett stort säkerhetsproblem.

Referenser

[1] http://en.wikipedia.org/wiki/Captive_portal, Hämtad: 2010-04-15.

[2] D. Gollmann, Computer Security, John Wiley & Sons, 2006.

[3] P. Mockapetris, IETF RFC 1035, Domain Names – Implementation and Specification, November 1987.

[4] J. Cache & V. Liu, Hacking Exposed Wireless, McGraw-Hill, 2007.

[5] http://www.tcpdump.org/, Hämtad: 2010-04-29.

[6] http://packetfactory.openwall.net/projects/libnet/index.html, Hämtad: 2010-04-29.

[7] http://802.11ninja.net/lorcon, Hämtad: 2010-04-29.

[8] http://airpwn.sourceforge.net, Hämtad: 2010-04-30.

[9] http://en.wikipedia.org/wiki/Berkeley_Packet_Filter, Hämtad: 2010-04-30.

[10] http://monkey.org/~dugsong/dsniff, Hämtad: 2010-04-30.

[11] IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2007.

[12] IETF RFC 793, Transmission Control Protocol, September 1981.