TDDC03-Projekt, våren 2005

Intrångsdetekteringssystem
Svart magi när den är som bäst

Grupp Proj007

Claes Leufvén clale457@student.liu.se

Rikard Nordin rikno188@student.liu.se

Handledare: David Byers

v. 2.3

Abstract

En IDS *är en viktig del i det Informationssäkerhetssystem som dagens företag behöver för att skydda sina system. I denna rapport kommer vi att berskriva den grundläggande teorin om hur* IDS *system fungerar och hur man kan utvärdera ett* IDS *system på ett bra sätt. Vi kommer också att genomföra ett mindre test där vi testar opensource* IDS*:en* SNORT *och den kommersiella* IDS*:en* ISS *Real Secure och jämför dessa två emot varandra.*

1 Inledning

1.1 Vad är en IDS?

IDS är en engelsk förkortning för Intrusion Detection System och är ett system som på något vis identifierar felaktig eller otillåten användning av ett datorsystem eller ett nätverk.

Det finns flera olika sätt för en IDS att göra något åt en upptäckt felaktighet. Vissa system försöker förhindra upptäckta felaktigheter, andra larmar en systemadministratör och låter denne avgöra ifall felaktigheten kan peka på ett potentiellt intrång eller intrångsförsök.

IDS:er kan indelas i två huvudkategorier: nätverksbaserade (Network based, NIDS) och värdbaserade (host based, HIDS). I det nätverksbaserade fallet handlar det om ett system som avlyssnar trafiken i ett nätverk och analyserar den. I det värdbaserade fallet är det en IDS som kör på en dator och undersöker den information den kan få från operativsystemet och andra körande program. I många fall kan en IDS vara en hybrid mellan de två huvudkategorierna.

1.2 Syfte

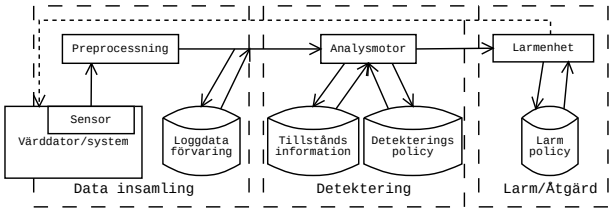
Syftet med rapporten är dels att sammanställa en teoretisk bas för att kunna utvärdera en IDS och peka på vilka hjälpmedel det finns för att genomföra bra tester. Rapporten behandlar specifikt vad man kan använda som testdata och vad det finns för verktyg för att utföra tester.

Rapporten beskriver också utförandet samt resultat från tester som gruppen utfört på tre olika IDS:er. Testerna utförs på IDS:er med olika ursprung för att visa hur kommersiella, Open Source och forskningssystem står sig mot varandra.

1.3 Frågeställning

Vi ämnar besvara följande frågor:

- Vilka utvärderings- och inlärningsdata finns tillgängliga och vad håller de för kvalité?



Figur 1. Funktionen hos en IDS

- Vad finns det för testprogram och testmetoder för att undersöka kvalitén hos en NIDS?
- Var ligger forskningsfronten idag?
- Hur bra står sig NIDS:er från Open Source-aktörer, kommersiella företag och forskningsdomänen mot varandra?
- Hur svårt är det att genomföra ett praktiskt test av olika NIDS:er?

1.4 Metod

Vår valda metod går ut på att först skapa oss en teoretisk grund genom att läsa olika skrifter som finns tillgängliga inom området. På detta sätt får vi en bra uppfattning om var forskningsfronten ligger idag och hur testerna ska kunna utföras på bästa sätt.

Vi går sedan vidare genom att ta vår insamlade kunskap och omsätta denna till praktiska tester på tre olika NIDS:er. Resultatet från dessa tester används sedan tillsammans med inläst teori för att besvara vår frågeställning.

1.5 Avgränsning

Vi har valt att avgränsa de praktiska testerna till att endast omfatta NIDS:er som går att hitta gratis eller som har en gratis testversion på Internet. Vi kommer inte att genomföra några tester av IDS:er som kräver speciell hårdvara.

2 Teori

2.1 IDS-teori

En IDS fungerar generellt genom att den tar in data från en eller flera sensorer, analyserar resultatet och utför en åtgärd eller larmar om systemet anses attackerat. Det finns flera teorier om andra sätt att strukturera en IDS på, men detta är den mest använda strukturen hos idag implementerade IDS:er [3].

2.1.1 Datainsamling

Sensorerna kan vara av flera olika modeller. Ofta läser den på ett eller annat sätt av trafik, vilket kan vara nätverkstrafik, processanrop till operativsystemets kärna eller läsa av loggdata från program.

Sensorerna innehåller sedan en preprocessor som skapar en mer användbar struktur av datamängden och sorterar bort ovidkommande information. Strukturförändringen kan innebära att skapa statistik över olika typer av data eller skapa en annan datastruktur för att underlätta senare analyser.

2.1.2 Detektering

Den datastruktur som sedan skapats går vidare till analysmotorn i IDS:en. Analysmotorn använder sig av tillståndsinformation och detekteringspolicer för att sortera data i två olika klasser: normal data och ickenormal data.

En enkel men ofta använd metod för att analysera data är att helt enkelt söka efter vissa uttryck i det data som kommer in till analysmotorn. De data som innehåller en matchande sträng ses som onormal.

2.1.3 Larm / åtgärd

Det data som sedan klassificerats som onormalt går vidare till larmenheten. Den har en policydatabas för vad den ska göra med olika former av onormal data. Då till exempel en sorts onormalitet ska resultera i ett larm till systemadministratören så ska ett annat resultera i att en nätverksuppkoppling ska stängas.

Larmenheten kan också innehålla funktioner för att räkna felaktigheter på ett eller annat sätt, eller för att poängsätta felaktigheter, och sedan alarmera eller utföra en åtgärd när mängden felaktigheter uppnår ett visst värde.

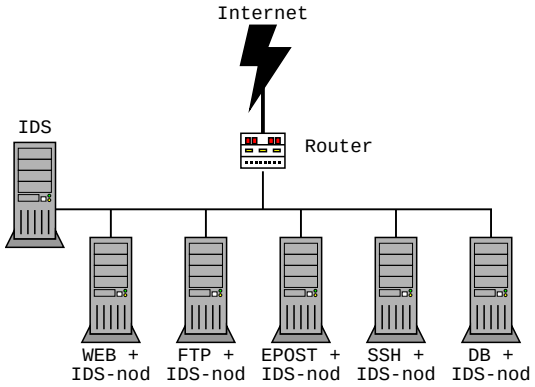
2.2 Olika arkitekturer

Det finns flera olika sorters arkitekturer hos IDS:er. Nedan beskrivs de huvudtyper som man kan träffa på.

2.2.1 Centraliserade

Den första modellen, och än idag den allra vanligaste, är en centraliserad IDS. Den består av en fristående IDS. Ibland är det en fristående hårdvaruplattform som kör IDS:en, och ibland körs den på en server vars huvuduppgift är en annan, men det är alltid bara en IDS.

Den stora sårbarheten med att använda ett sådant system är att en lyckad attack mot IDS:en slår ut allt skydd, det finns ingen backup. Arkitekturen kan också ställa till med problem på högt belastade system, där en centraliserad IDS lätt kan bli en flaskhals i datorsystemet. Ett sista problem är att



Figur 2. En hierakiskt distribuerad IDS

en centraliserad IDS bara får in data i en punkt. Den ser till exempel bara trafiken på ena sidan om en brandvägg, och vet inte vilken trafik som finns på den andra sidan brandväggen.

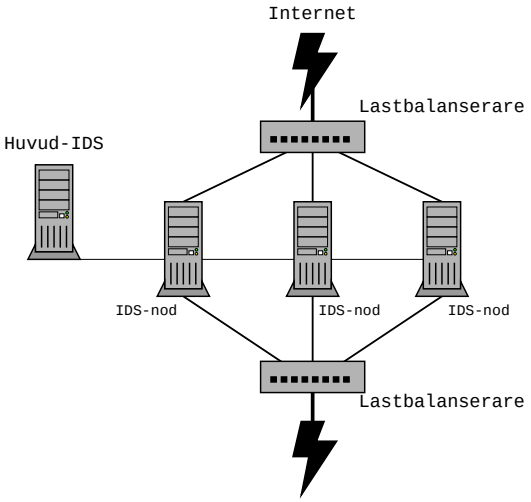
2.2.2 Hierarkiskt distribuerade

Distribuerade IDS:er [13] fungerar genom att ett flertal IDS:er arbetar på ett nätverk med att samla upp och preprocessa trafik. Resultatet skickas sedan vidare till en huvud-IDS som analyserar och eventuellt larmar vid problem.

Det kan finnas flera anledningar till att använda sig av distribuerade IDS:er. Den vanligaste anledningen är att spara resurser i nodpunkterna. Med hjälp av en distribuerad IDS kan till exempel alla arbetsstationer och servrar på ett företag köra en liten datainsamlande IDS som sedan skickar data till en central IDS som gör det tunga arbetet med att analysera informationen. På så sätt använder inte IDS:en upp alla resurser på till exempel webbservern.

Denna typ av lösning kan också vara bättre på att upptäcka attacker, genom att den kan sätta ett helt nätverk i beredskapsläge om en attack upptäcks mot en dator. Kanske märks inte attacken på bara en server, men när samma sak händer på alla servrar så kan IDS:en identifiera det hela som en attack.

En annan anledning till att använda en distribuerad IDS är om man har hög bandbredd i ett nät. En internetleverantör kan till exempel vilja ha en IDS som bevakar sin 10-Gigabitsuppkoppling mot Internet. Endast en IDS på den uppkopplingen kan ha svårt att hinna processa all information. Istället delas först trafiken upp med hjälp av lastbalansering, för att sedan passera genom ett flertal IDS:er som samlar upp information, som skickas till en huvud-IDS som analyserar och eventuellt inför åtgärder mot misstänkt trafik. Uppdelningen av trafiken behöver ske på så sätt att varje dataförbindelse alltid går igenom samma IDS-nod och inte sprids över flera noder. Trafiken samlas sedan ihop igen efter IDS:erna och passerar det hela som om ingenting har



Figur 3. IDS som bygger på lastbalansering

hänt.

Vid utveckling av en distribuerad IDS är ett av de svåraste problemen hur man ska lösa dataförbindelser, till exempel TCP-uppkopplingar, där IDS:noderna endast ser en liten del var av dataförbindelsen. Sådana fall sker ofta när till exempel ett företag har flera olika fysiska förbindelser och tillämpar dynamisk routning. Detta kan lösas genom att noderna rapporterar den data de samlat in, och huvudnoden sedan får sammanställa datafragmenten i rätt ordning för att kunna analysera dataförbindelsen i sin helhet.

2.2.3 P2P-distribuerade

Det finns en annan svaghet med hierakiskt distribuerade IDS:er. Om huvud-IDS:en slås ut kommer hela IDS-systemet att slås ut. Det är alltså mycket angreppskänsligt vid huvudnoden. För att undvika detta finns det forskning som löser problemet genom att använda ett Peer-to-Peer-nätverk mellan IDS-noderna istället för att undvika den hierakiska strukturen [16]. Detta ger en stabilare struktur som inte är känslig för att en IDS-nod slås ut – de övriga kommer att fortsätta arbeta ändå.

P2P-distribuerade IDS:er har dock svagheten att de inte ger någon större arbetslättnad för noderna. De kommer att kunna fungera som en vanlig IDS, men med fördelen att de kan utbyta information med de övriga noderna. I ett P2P-nätverk kan man införa lastbalansering så att en IDS-nod som inte har så mycket att göra kan analysera en annan överbelastad nods data.

2.3 Önskvärda egenskaper hos en IDS

Det finns flera aspekter på vad man vill uppnå hos en bra IDS. Givetvis är det svårt att uppnå alla, och ibland till och med omöjligt. Vi har här tagit upp de vanligaste kraven på en bra IDS [10].

2.3.1 Stabilitet

En IDS behöver vara stabil. Om den ofta falierar så blir den en belastning för systemadministratörerna, ungefär som en sjuk vakthund. Meningen är att man ska kunna installera en IDS för att sedan låta den köra utan större problem flera år i sträck. Det är också därför som många kommersiella IDS:er använder sig av egen hårdvara för att köra på. Då kan tillverkarna släppa många krav på att mjukvaran ska kunna köras på många mer eller mindre kompatibla hårdvarusystem. Bara en sådan enkel sak som ett nätverkskort som inte riktigt följer standarden kan omkullkasta funktionen hos en IDS genom att inte leverera tillräcklig prestanda.

Om en IDS trots att den är stabil råkar krascha så måste den ha sparat undan information så att den kan återuppta arbetet därifrån den kraschade.

2.3.2 Resurssnålhet

En IDS behöver i många fall vara resurssnål. Ibland är det meningen att en IDS ska köras på ett system som redan kör annan mjukvara, till exempel en webbserver. Om då IDS:en kräver stora resurser så kommer webbservern att inte fungera tillfredsställande.

Många IDS:er fungerar så att de släpper igenom data utan att kontrollera det, ifall den inte hinner analysera data i realtid. Detta för att IDS:en inte ska skapa en flaskhals i ett datorsystem. Om det inträffar så kan en hacker ta sig förbi oupptäckt genom att kombinera sin attack mot datorsystemet med en DOS-attack mot IDS:en.

2.3.3 Effektivitet

En IDS måste vara effektiv, både i fråga om att upptäcka attacker och larma eller förhindra den på rätt sätt. Om den släpper igenom attacker utan att larma är den värdelös, och fyller ingen funktion. Eftersom säkerhetshål och nya sätt att attackera ett datorsystem upptäcks varje dag måste en IDS ligga långt fram och till exempel ständigt uppdatera sina signaturdatabaser om den är signaturbaserad.

2.3.4 Feltolererande

Ett feltolerant system tolererar när användare gör ett ofarligt fel. Om en IDS larmar på falska attacker kommer den snart att tas ur drift eftersom administrationen blir för stor. Många falska larm kan också leda till att personalen tappar koncentrationen och ignorerar efterföljande larm, vilket kan leda till att även äkta larm förbises.

Om IDS:en fungerar genom att spärra misstänkta intrångsförsök kan falska larm ställa till med enorma problem. Om en IDS ständigt spärrar ute användare för att undvika eventuella intrångsförsök så kommer ingen att kunna göra sitt arbete ordentligt.

2.3.5 Anpassingsbart

En IDS måste själv anpassa sig så att den kan motstå nya typer av attacker, eller på ett enkelt sätt gå att uppdatera med nya funktioner för att ta hand om nya attacker. Kravet på effektivitet är beroende av anpassningsbarhet. En IDS som inte är anpassningsbar kommer snart att bli ineffektiv på att upptäcka nya attacker. Ett sätt att uppnå automatisk anpassningsbarhet hos en IDS är att använda sig av en IDS som är självlärande.

2.3.6 Skalbart

En IDS måste skala bra. De flesta IDS:er körs hos stora företag med hög nätverkstrafik och systemanvändning, som till exempel vid en router som ansluter mot Internet. Skalar den dåligt blir den snart en flaskhals i systemet. Ett sätt att uppnå hög skalbarhet hos en IDS är att konstruera den enligt en distribuerad modell. Den går då att utöka med fler noder allt efter att datorsystemet den ska övervaka växer.

2.3.7 Lättkonfigurerat

Som alla andra datorprogram är det viktigt att en IDS är lättkonfigurerad. Om mjukvaran är svår att ställa in kan det leda till att den blir felaktigt inställd och fungerar inte som administratörerna tänkt sig. Specifikt för en IDS är att larmfunktionen är lättanvänd och ger god information som till exempel datum, avsändare, mottagare, typ av felaktighet.

2.4 Olika sorters IDS:er

Det finns ett antal modeller en IDS kan arbeta på för att identifiera felaktigheter. Vissa IDS:er kombinerar också flera av dessa modeller.

2.4.1 Normalavvikande modeller / Expertsystem

Två modeller som arbetar på liknande sätt är normalavvikande modeller och expertsystem. Dessa två modeller är bland de äldsta använda bland IDS:er [6].

Normalavvikande modeller har någon form av data från systemet som den jämför med ett normalvärde. Om värdet från systemet inte stämmer överens med normalvärdet så slår den larm. De kan till exempel räkna felaktiga inloggningar för en användare, och slår larm när antalet fel blir ovanligt många. Den kan också föra statistik över olika systemresurser och slå larm när värdena faller utanför normalvärdet.

Expertsystem kombinerar flera olika typer av data och sammanställer enligt en normalavvikande modell. Expertsystemet kan till exempel kombinera data om felaktiga inloggningar med data från programkörningar på en dator för att upptäcka användare som kör kommandon som de inte brukar köra. Signaturbaserade IDS:er är oftast implementerade som en yngre generation av expertsystem.

2.4.2 Agentbaserade modeller

En agent är en fristående modul som arbetar utan att vara beroende av andra agenter [18]. I fallet med en IDS kan det betyda att man kör två olika system samtidigt för att dubbelkontrollera data och alltid ha ett backupsystem om det ena fallerar.

I en renodlad agentbaserad modell bygger IDS:en på att olika delar av arbetet tas omhand av olika agenter. De olika agenterna kan också tas bort och läggas till efter behov för att lägga till eller ta bort funktionalitet utan att behöva störa övriga agenter. Det betyder att man kan uppgradera en liten del av IDS:en samtidigt som övriga delar kör vidare utan störningar. Det ger ett stabilt och anpassningsbart system.

2.4.3 Datautvinnande modeller / Självlärande system

Den här sortens IDS:er samlar data och skapar statistik över olika händelser [11]. Den gör också kopplingar mellan de olika händelserna. Den databas som sedan byggs upp använder IDS:en för att klassificera händelser och avgöra ifall användarna är de användare de utger sig för att vara och om de arbetar med saker de ska göra. Likheten mellan datautvinnande modeller och självlärande är slående. Egentligen kan man säga att datautvinnande modeller är en delmängd av självlärande system.

Inom självlärande system hittar man system som lär sig genom att man börjar med att köra systemet i en säker miljö och sedan spela upp träningstrafik för den. Andra system fungerar genom att de börjar med en liten signaturbas för att sedan varna för allt som verkar misstänkt. När administratören konfirmerar det som en felaktighet eller avslår det som en tillåten aktivitet så lägger systemet upp detta i en databas för att inte behöva fråga nästa gång. Skillnaden mellan dessa två modeller är att den datautvinnande modellen enbart börjar med att logga systemet. Den ska alltså i teorin inte behöva någon systemadministratör som matar den med träningsdata.

2.4.4 Artificiella neurala nätverk / Immunsystemsbaserade

Det här är den mest intressanta sorten av IDS:er. Dessa är också självlärande system, men utvecklas mer mot att efterlikna naturliga system som återfinns i till exempel människor.

Artificiella neurala nätverk [9] är en kombination av datautvinnande och självlärande som man spelar upp träningstrafik för. Trafiken lagras i ett neuralt nätverk. Denna information använder den sedan för att klassificera senare data. Den plockar också in senare data i det neurala nätverket och blir på så sätt 'smartare' hela tiden.

Immunbaserade IDS:er [10] ska arbeta på samma sätt som kroppens naturliga immunförsvar. En detektor ska skapa mönster av alla händelser, och lagra dessa i en databas. När ett mönster inte återfinns i databasen ska IDS:en gå in i ett larmläge. Om systemadministratören markerar det som ett godkänt mönster så markeras det som godkänt för framtida bruk. Mönstren i databasen ska också ha en livslängd som ökar om mönstret ofta uppträder. Det ska leda till att mönster som sällan uppkommer klassificeras som något mer misstänkta än mönster som ständigt påträffas.

Skillnaden på den immunbaserade modellen och andra modeller vi tagit upp, ska vara att den på ett intelligent och naturligt sätt ska efterlikna kroppens immunsystem och på grund av det ha mindre felfrekvens. Tyvärr verkar det som det mesta inom detta område är ren forskning, och vi har inte funnit några fullt implementerade IDS:er som använder några av dessa modeller.

2.5 Det senaste på forskningsfronten

Vad vi hittat som det senaste på forskningsfronten handlar mycket om att göra IDS:er mer självlärande och 'intelligenta'. Immunbaserade modeller, artificiella nätverk och liknande självlärande system är det som dominerar.

Röd tråd genom forskningsfronten är att få ner antalet falska larm utan risk för att missa nya typer av felaktigheter. DARPA-testerna visade att de bästa IDS:erna klarade av att detektera ungefär 70% av alla attacker [12]. Även om IDS:er har utvecklats mycket efter det har det också tillkommit många nya attacker som lätt förvirrar en IDS. Ett annat mål är att skapa "install and forget"-system som själva ska kunna hålla sig uppdaterade och moderna genom olika självinlärningsmodeller. De ska alltså kräva minimalt med installations- och konfigurationstid.

2.6 Testdata

För att kunna testa och träna en IDS behövs en datamängd som både innehåller normal data och data från attacker. Formatet på datamängden beror på vad det är för typ av IDS som används, men vanliga format är nätverkstrafik som är inspelad med programmet tcpdump [35], data från Solaris BSM [17] och data från Windows NT audit logs.

Om datamängden ska användas för att träna upp en självlärande IDS behöver den återspegla verkligheten på ett bra sätt. Där kan automatgenerad data bli problem då det blir allt för statistisk korrekt så att den lärande IDS:en sedan ger alarm varje gång en användare gör ett vanligt fel som inte fanns med i det automatgenererade träningsdatamängden.

Det finns forskningsgrupper som har försökt skapa bra automatgenerad data genom att först utföra en statistisk analys på riktig data och sedan med hjälp av analysen skapa träningsdata. Den genererade datamängden har de sedan analyserat på samma sätt som den riktiga datamängden och funnit att statistiskt sett ger samma resultat [7].

Tyvärr finns det få publika datamängder att tillgå. Detta beror på att det är svårt att erhålla data som går att använda efter att man har anonymiserat det. Anonymiseringen krävs för att inga känsliga personuppgifter, lösenord och login eller annan känslig information läcker ut som kan äventyra säkerheten hos de datasystem och nätverk vars trafik och loggar man erhållit [2].

En annan svårighet är att veta vad man ska spela in och vilka punkter i ett nätverk eller datorsystem som ger bäst kvalité på datamängden. De publicerade datamängder som man kan använda för testning och träning gås igenom nedan.

2.6.1 DARPA 1998 1999 intrusion detection evaluations data sets

Datamängderna [28] kan delas upp i två delar, ett off-line och ett on-line. Off-line-mängden bestod av 7 veckors data som var klassiferade i två olika klasser: normal- och attack-sessioner. Detta skulle användas av utvecklarna för att kunna träna och utvärdera sina IDS:er. On-line-mängden bestod av två veckor data som skulle användas för själva testandet. Formaten på datat var tcpdump [35], Solaris BSM [17] och Windows NT audit logs [1].

Off-line-mängden spelades upp på det testnätverk där man körde de IDS:er man ville testa. Själva uppspelandet gjordes med ett verktyg som heter tcpreplay [36]. Resultaten från IDS:erna analyserades och presenterades.

Eftersom detta var det första riktiga försöket till att skapa bra testdata, så fanns det inte några tidigare försök att ta lärdom ifrån. Det första problemet DARPAs utvecklare ställdes för var hur man skulle skapa datamängden. Man kom fram till två olika alternativ: insamling av data från riktiga företag och syntiserad data genererad i ett test-nätverk. Ovan har vi belyst varför det första alternativet inte gick att genomföra så därför valde de att syntetiskt generera data i ett testnätverk.

För att få den genererade datamängden så verklig som

möjligt så började man med att analysera 4 månaders data från Hanscom Air Force Base. Man fann att följande protokoll var intressanta att generera: HTTP, X Windows, SQL, SMTP, DNS, FTP, POP3, Finger, Telnet, IRC, SNMP och Time. Man genererade sedan data för ett nätverk med 1000-tals maskiner och flera olika användargrupper så som kontorspersonal, programmerare och fabriksarbetare. På så sätt fick utvecklarna realistisk bakgrundstrafik till de båda delarna av datamängden. Sedan så lade de till 120 olika attacker från 38 olika kategorier till bakgrundstrafiken. Tabell 1 visar på den stora spridningen av olika attacker.

Då datamängderna från 1998 fokuserade på att få fram tekniska sätt att testa IDS:er så flyttades fokuset för datamängderna som kom ut 1999 till att fokusera mera på att testa hela IDS-system och hitta orsaker till varför de missade nyare variater av attackerna från 1998. Andra större förändringar var att man lade till Windows NT till listan över attackerade operativsystem. 1998 hade man bara använt sig av Solaris och Linux som operativsystem. Man lade också till en analys där man tittade på hur många falsklarm de testade IDS:erna genererade. Det finns många begränsningar med datamängderna från DARPA, men den största är att de gjordes 1998 och 1999, och sedan dess har många nya attacker och tekniker för att inte bli upptäckt av en IDS utvecklas.

2.6.2 LARIAT

År 2000 så släpptes LARIAT [15], vars motivation är att skapa en grupp av verktyg för att testa och utveckla brandväggar och IDS:er. LARIAT simulerar en liten organisation som är kopplad mot internet med attacker mot både UNIX- och Windows-maskiner och nätverkstrafik i hög hastighet. I LARIAT så väljer först användaren mellan olika typer av bakgrunds- och attacktrafik. Sedan initierar LARIAT automatiskt både maskinen som ska generera all trafik samt initierar de maskiner som ska utsättas för attackerna. LARIAT försöker vara så enkelt som möjligt att automatisera, konfigurera och administrera. De attacker och normal trafik som används är lika de som användes i DARPA 1999 och 1999, dock har man lagt till ett attack-API så att användaren lätt ska kunna lägga till nya attacker och modifiera gamla.

LARIAT är ett bra försök till att uppdatera attackerna från DARPA 1998 och 1999. Till exempel har man utvecklat ett scenario där en attackerare attackerar flera maskiner för att sedan installera en DDOS klient och börja attackera andra maskiner med den. Det verkar som LARIAT fortfarande utvecklas men ingen publik version har släppts. Man kan skicka ett mail till utvecklarna[1] för bli notifierade om de skulle släppa LARIAT publikt i framtiden.

[1] lariat@sst.ll.mit.edu

2.6.3 Capture the Flag

I väntan på bättre datamängder så har många utvecklare av IDS:er börjat använda sig av data från olika hackertävlingar. En av de mest populära är Capture the Flag [32] på datorpartyt Defcon [20] i USA. Där tävlar flera lag om vem som kan knäcka de andra lagens datorer så snabbt som möjligt, samtidigt som de ska skydda sina egna datorer från de andra lagen. Genom att spela in denna trafik får man en stor uppsjö av olika attacker samtidigt som man kan hitta nya opublicerade attacker. Tyvärr finns det många nackdelar med att använda denna trafik för tester, vi ska här lista de största.

Klassifiering Då lagen inte har tid eller möjlighet att dokumentera varenda attack de utför får man ingen bra klassifiering på vad som är ren bakgrundstrafik och vad som är sofistikerade attacker. Detta gör att man måste lägga ner mycket arbete på att först utskilja vad som är attacker och vad som är bakgrundstrafik samtidigt som det kan vara näst intill omöjligt om någon av lagen använder sig av en opublicerad attack.

Bakgrunds- och attacktrafik Fördelningen mellan bakgrundstrafik och attacktrafik är uppenbarligen väldigt snedfördelad då de flesta lagen attackerar mer än vad de till exempel läser mail och surfar på websidor.

Pågrund av dessa nackdelar är data från sådana tävlingar egentligen bara bra för att skapa sig en idé om hur sofistikerade attacker en IDS kan behöva klara av.

2.7 Verktyg

2.7.1 Congestant

Congestant är utvecklat av en person vid namn Horizon och var först publicerad i Phrack nr 54 [8]. I artikeln tas flera olika typer av tekniker för att komma undan IDS:er, till exempel fragmentering av IP-paket, felaktiga IP- och TCP-checksums och paket med olika TTL värden.

När artikeln och programmet skrevs 1997 var det inte många IDS:er som klarade av att till exempel följa en hårt fragmenterad ström av paket, men de flesta av dagens IDS kan man anta klarar både fragmenterade paket och paket med felaktiga checksums. Däremot är TTL-problem ett av de problemen som är svåra att lösa för en IDS utan att ha en detaljerad bild av hur nätverket ser ut.

2.7.2 IDSwakeup

IDSwakeup [25] är utvecklat av Stephane Aubert och består av två delar – ett shellscript innehållande olika attacker som skickas iväg med hjälp av Hping [24] och ett

	Solaris	SunOS	Linux
DOS-attacker (11 typer, 43 instanser)	Back, Neptune, Ping of Death, Smurf, syslog, Land, apache 2, Mailbomb, Process table, UDP storm	Back, Neptune, Ping of Death, Smurf, apache 2, Land, Mailbomb, Process table, UDP storm	Back, Neptune, Ping of Death, Smurf, Teardrop, Land, apache 2, Mailbomb, Process table, UDP storm
Intrångsattacker (14 typer, 17 instanser)	Ordlistsattack, ftp-write, guest, phf, http-tunnel, xlock, xsnoop	Ordlistsattack, ftp-write, guest, phf, http-tunnel, xlock, xsnoop	Ordlistsattack, ftp-write, guest, imap, phf, named, http-tunnel, sendmail, xlock, xsnoop
Root-attacker (7 typer, 38 instanser)	Eject, ffbconfig, Fdformat, ps	Loadmodule, ps	Perl, xterm
Analys / tester (6 typer, 22 instanser)	Eject, nmap, Port Sweep, Satan, mscan, saint	Eject, nmap, Port Sweep, Satan, mscan, saint	Eject, nmap, Port Sweep, Satan, mscan, saint

Tabell 1. Olika typer av attacker i DARPA -98

program, IWU, skrivet i C-kod som IP-paket innehållande valfri data.

Eftersom senaste versionen av IDSwakeup släpptes i Oktober 2000 så är de flesta attackerna gamla. Dock visar Stephane Aubert hur flexibelt och kraftfullt Hping är. Syftet med IDSwakeup är att testa hur känslig den testade IDS:en är för paket som ser ut som välkända attacker men ändå inte är det, det vill säga hur många falska positiva larm IDS:en ger.

2.7.3 Fragrouter

Fragrouter [23] implementerar många av de fragmenteringstekniker som tas upp i Secure Networks "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detectionfrån Januari 1998 [14] för att lura en IDS.

Fragrouter i sig kan inte generera paket med attacker i utan istället lyssnar den på ett interface efter trafik att paketera om efter de specifikationer man har valt vid start. Sedan skickas de iväg till en förutbestämd IP-adress. Med hjälp av fragrouter kan man ta ett program som utför en enkel attack och direkt få attacken att bli ännu mera avancerad att upptäcka genom att låta paketen gå igenom fragrouter.

Detta kan vara väldigt effektivt då olika operativsystem hanterar fragmenterade paket på olika sätt. Det medför att IDS:en måste antingen veta vilket operativsystem som körs på maskinen som attacken är riktad mot eller analysera paketet på alla de olika sätt fragmenten kan sättas ihop på av den attackerade maskinen. Dagens IDS:er är duktiga på att analysera fragmenterade paket, så detta verktyg är inte längre så effektivt som det var 1999 då det släpptes.

2.7.4 Hping

Hping [24] är ett verktyg för att lätt kunna bygga ICMP-, UDP- och TCP-paket genom att använda några command-line-växlar till hping. I standardläge fungerar hping som det vanliga UNIX-kommandot ping, men det kan också utföra traceroutes med TCP-paket, paket med förfalskad avsändare och många andra saker.

I motsats till de andra verktygen vi har tagit upp så är utvecklingen av detta verktyg fortfarande aktivt, och någon gång under 2005 kommer Hping version 3 släppas. Där har de implementerat ett stöd för TCL så att användaren enkelt ska kunna skriva avancerade paketdefinitioner. Samtidigt hjälper Hping användaren med saker såsom checksums, paketlängder och underliggande paketlager.

Målet med detta TCL stöd är att det skapas ett bibliotek på Internet med små TCL-script för till exempel så kallade Proof of Concepts. Proof of Concepts är ett bevis på att ett säkerhetsproblem i ett program går att utnyttja för att knäcka programmet. Som exempel på kraftfullheten i Hping version 3 ska det vara enkelt att implementera de flesta av portscanning-teknikerna som finns i Nmap enligt Hpings författare, Salvatore Sanfilippo [4]. Förnärvarande finns Hping version 2 för Linux, FreeBSD, OpenBSD, NetBSD och Solaris.

2.7.5 Nmap

Nmap [29] är gjort för att scanna av större nätverk. När man scannar tar man reda på vilka IP-addresser som används för tillfället och vilka portar som är öppna och vilket operativsystem som körs på maskinerna. Nmap har många olika typer av scanningtekniker där vissa är svårare för en IDS att upptäcka. Det är viktigt för en IDS att upptäcka dessa scanningar för det är goda tecken till början av en attack.

Nmap kan också användas för att ta reda på om en brandvägg är rätt konfigurerad och spärrar rätt portar. För närvarande finns nmap både i en textversion och grafisk version till många olika operativsystem, till exempel Windows, Linux, Mac OSX, FreeBSD, NetBSD, OpenBSD och Solaris.

Utvecklingen av Nmap är fortfarande aktiv och databasen över olika operativsystems fingeravtryck (används vid operativsystems identifikation) uppdateras kontinuerligt. Eftersom Nmap både berättar vilket operativsystem och vilka tjänster som körs på maskinerna, och ibland också vilken version av tjänstens programvara, så ger det en attackerare en mycket bra startpunkt då de vet mera exakt vilka buggar de kan utnyttja och kan förfina attacker som behöver utföras.

2.7.6 Tcpreplay

Tcpreplay [36] är ett verktyg för att återuppspela trafik som man har spelat in med hjälp av tcpdump eller ethereal [22]. Med tcpreplay kan man ta den inspelade trafiken och ändra på paketen på nivåerna 2-4 för att sedan spela upp dem i valfri hastighet utan att behöva bygga upp det nätverk som den inspelade trafiken representerar.

Med en maskin som har tcpreplay så kan man både simulera attackeraren och offret, samtidigt som den IDS som ska testas avlyssnar trafiken. Tyvärr hanterar inte tcpreplay TCP-sessioner mer än att den kan spela upp en inspelad session. Flowreplay som kommer med tcpreplay ska lösa det problemet så att man kan återskapa och modifiera en TCP-session. Senaste utvecklingsversionen av tcpreplay släpptes 19 april 2005 och tcpreplay fungerar på de flesta operativsystem som stödjer libpcap [35] och libnet [27].

3 Praktik

3.1 Valda IDS:er

För att kunna svara på frågan i vår frågeställning angående skillnaden mellan IDS:er från tre olika områden – forskning, Open source och kommersiella – bestämde vi oss att välja tre olika IDS:er från de tre områdena, vilket visade sig vara lättare sagt än gjort. Här nedan kommer vi ta upp vad vi hittade för IDS:er när vi letade i de olika områdena. Som vi skrev i stycke 1.5 om avgränsning så kommer vi enbart testa den nätverksbaserade delen av IDS:erna.

3.1.1 Open source

De flesta känner till IDS:en SNORT [34], men det finns också en IDS som heter Prelude [31] som börjar bli mer och mer populär. Vi valde SNORT då den fortfarande är en av de mest använda och populära IDS:erna, det hade varit mycket intressant att jämföra Prelude och SNORT men det fanns inte tillräckligt med tid för att genomföra en sådan utvärdering.

SNORT är en signaturbaserad nätverksbaserad IDS med ett textgränssnitt. Konfiguration sker i konfigurationsfiler och resultatet hamnar sedan i en SQL-databas. Till detta sätter man lämpligen ett grafiskt gränssnitt för att ändra konfigurationen, uppdatera signaturer och övervaka SNORT. Ett av de mest populära verktygen för övervakning är BASE [19], vilket också kommer att vara det verktyg vi kommer att använda oss av, men det finns många andra.

För att regelbundet hålla sin signaturdatabas uppdaterad finns till exempel programmet Oinkmaster [30] som laddar ned nya signaturer från SNORT:s hemsida och installerar dessa, om nya signaturer tillkommer. Den visar också exakt vilka regler som ändrats mellan olika uppdateringar, vilket gör att man får bra överblick över hur signaturdatabasen förändras över tid. Oinkmaster är ett perl-skript och kan köras under nästan alla system, som till exempel Linux, Windows, Solaris och VMS

3.1.2 Kommersiella

När vi letade efter kommersiella IDS:er fann vi att många företag säljer enbart sin IDS-lösning som ett hårdvarupaket. För att kunna utvärdera deras produkt behöver man få tillgång till deras hårdvara som ibland kan vara ett problem, speciellt om man inte är en större kund till företaget. Detta gör att man kan bli tvungen att köpa ‘grisen i säcken’ och inom IDS området är det en dålig idé.

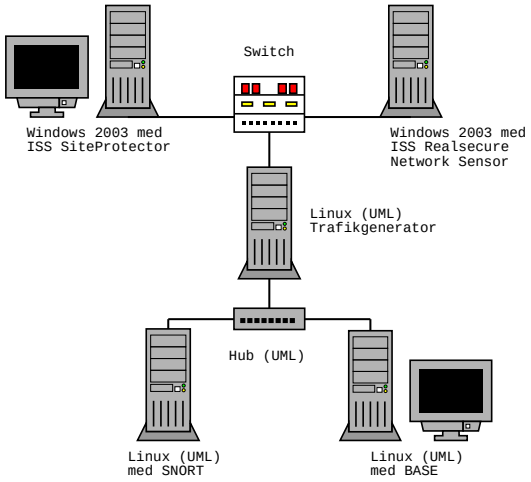
De större tillverkarna som till exempel Cisco har en IDS lösning, men där behöver man förutom en speciell hårdvara för huvudpunkten också speciella add-on-kort för varje router eller switch man vill övervaka. Detta gör det ännu svårare och dyrare att utvärdera deras system i helhet. Ett av de få företag som lät oss ladda ner en testversion var Enterasys [21] som har produktfamiljen Dragon, som är en signaturbaserad IDS lösning. Tyvärr kunde vi inte utföra något bra test av denna produkt då det visade sig att de bara levererade 1/5 del av signaturdatabasen med den evalueringsversion vi hade fått.

Till sist så tittade vi på ISS:s [26] IDS-lösning vid namn Real Secure. Deras lösning kan man både få paketerad i en specialiserad hårdvarulösning eller som ett program man installerar på sin befintliga hårdvara. Real Secure är en av de populäraste kommersiella IDS:erna. De påstår sig klara trafik i 1 Gbit men då behöver man köpa deras speciella hårdvara.

Den del av Real Secure vi valde att använda är Network Sensor. Den installeras på en maskin, där den samlar information från nätverket. Till detta använder vi SiteProtector som är ett övervakningsprogram som samlar in data från Network Sensor, för att sedan kunna ge administratören möjligheter att se händelser, ge varningar och larm om misstänkta intrångsförsök och även kunna automatiskt stänga ned attackerade system om man så vill.

3.1.3 Forskning

Vi hade mycket svårt att hitta en IDS som var nyutvecklad från forskningsområdet. Detta beror på att IDS-området



Figur 4. Testnätet som använts

i sig är relativt nytt och de Open source-alternativ som finns lockar till sig många forskare, som testar sina idéer genom att implementera det som till exempel en plugin till SNORT eller Prelude. Som exempel finns det en anomaly detection plugin för SNORT, vid namn SPADE [5]. Många av de artiklar vi har läst har tyvärr inte sagt något om de har gjort någon implementation av sina teorier det får man nog se som en stor brist i området som helhet.

3.2 Testdata och testprogram

Vi har valt att spela upp trafik från en Capture The Flag-liknande tävling där en Linux-maskin sattes upp på Internet och hela världen fick försöka knäcka den inom fyra dygn [33].

Genom att använda oss av tcpreplay [36] kommer vi att spela upp trafiken som till största del består av olika sorters attacker och jämföra hur de olika IDS:erna reagerar. Då trafiken inte är klassificerad och attackerna inte namngivna så kommer vi istället att försöka skapa oss en bild av trafiken genom att jämföra vad de olika IDS:erna rapporterar för attacker.

3.3 Testnät

Figur 4 visar hur vårt testnät kommer att vara uppbyggt. De maskiner som kör operativsystemet Linux implementerar vi som User-mode Linux (UML -instanser och Windows 2003 körs på separata maskiner. Trafikgeneratorn kommer köras under User-mode Linux.

3.4 Resultatet från test av IDS:er

Vi spelade upp ungefär 100 Mb testdata på trafikgeneratorn som såg till att trafiken kom fram till maskinen som körde Real Secure Network sensor och SNORT samtidigt. Här följer en redovisning av den information man kunde få ut ur de olika IDS:erna.

3.4.1 ISS Real Secure

Real Secure klassifierade alarmen i tre olike klasser *High*, *Medium* och *Low*. De flesta alarm som klassifieras som *High* är olika former av säkerhetshål i webbapplikationer som till exempel i PHPbb som maskar har använts sig av för att sprida sig. Portscanningar detekteras i *Medium* och förmodligen är det därför som just den klassen fick så höga siffror. I klassen *Low* är det rätt blandat med typer av alarm med allt från gamla säkerhetshål som *land packet* till enkla DOS-attacker (Ping floods).

Klassifieringen av de olika alarmen kan ses lite underlig ibland då till exempel en attack som kan krasha Cisco routers klassifieras som *Low* samtidigt som ett försök att hitta maskiner körandes Back Orifice klassas som *High*. För varje detektering kan man se datum och IP-data samt en sida innehållande information om vilka system som är känsliga för den typen av attack, samt länkar till webbplatser innehållande mer informatiom attackerna och verktygen som används vid generering av attackerna.

3.4.2 SNORT

Gränssnittet som vi valde till SNORT heter BASE. BASE är också ett Open Source-projekt som bygger på systemet ACID som CERT har utvecklat. BASE använder sig av en MYSQL-databas som SNORT lagrar insamlat data i. BASE klassifierar inte de olika alarmen på samma sätt som ISS Real Secure utan istället har de ett flertal kategorier baserat på typ av alarm, till exempel web-application, non-standard-protocol och network-scan. Användaren får här själv inse hur allvarligt själva larmet är utifrån informationen som SNORT levererar. För varje larm så föreslår BASE på samma sätt som ISS Real Secure ett flertal olika hemsidor där man kan läsa mer om vad larmet innebär. Man har också möjlighet att se en sparad kopia av paketet i tcpdump-format för att kunna analysera det mera ingående.

Under de första testerna visade det sig att SNORT bara detekterade ungefär 50% så många larm som Real Secure gjorde. Då vi undersökte saken närmare visade det sig att SNORT tappade ungefär 50% av alla paket vi skickat. Vad detta beror på är vi inte helt säkra på, men det härrör sig från att vi kör SNORT på en UML-maskin. Om det sedan är UML som tappar paket, eller om det är SNORT som inte är kompatibelt med UML vet vi inte, men de två fungerar inte

	ISS	SNORT
Antal alarm	2831	5921
Antal Kategorier	95	191
Antal käll-IP	65	43

Tabell 2. Resultat från test

tillsammans. Vi gjorde om alla tester, och använde oss av en funktion i SNORT som gör att man kan mata den med en fil med inspelad trafik direkt istället för att låta den avlyssna nätverket.

3.4.3 Jämförelse mellan ISS och SNORT

Vår ursprungliga tanke var att jämföra det utdata vi fick från de två olika IDS:erna. Tyvärr visade det sig då vi genomfört testerna att det är i princip omöjligt att jämföra sådant data. En anledning är att de olika IDS:erna inte presenterar sitt data på samma vis. Real Secure presenterar sina alarm grupperat efter typ av attack, och SNORT presenterar sina alarm grupperat efter vilken typ av trafik det är.

Ända sättet att jämföra resultaten från de två IDS:erna är att manuellt jämföra varje alarm. Sök- och filtreringsfunktionerna är dock dåliga i både Real Secure och BASE, vilket försvårar arbetet betydligt. Det är helt enkelt mer eller mindre omöjligt att få ut data ur de båda IDS:erna på ett sådant sätt att de är maskinellt jämförbara med varandra. Både SNORT och Real Secure klarar av att exportera insamlad data till en fil i CSV-format, men det möjliggör inte en analys eftersom till exempel CVE-nummer för de olika attackerna inte tas med.

För att undersöka vad den stora skillanden på mängden larm i tabell 2 berodde på, så valde vi slumpmässigt ut 6 stycken IP-addresser från de larm som Real Secure hade registrerat och jämförde vilka larm som hade registrerats från SNORT och Real Secure.

- IP-adress: 4.14.187.57
  - Real Secure: *TELNET auth attack*
  - SNORT: Inget larm.
  - Kommentar: För detta paket så registrerade Real Secure det som en misslyckad telnet-inloggning med användarnamnet “guesty”. SNORT registrerade inget larm alls för denna IP-adress. Vi analyserade trafiken från denna IP-adress och såg inte något paket med strängen “guesty” i så förmodligen var detta ett felaktigt larm från Real Secure.
- IP-adress: 12.25.27.51
  - Real Secure: 6 st *Ping Flood*
  - SNORT: Inget larm.
  - Kommentar: Vi tror att Real Secure har gränsvärden för ICMP- paket per sekund som är satt mycket lägre än i SNORT.
- IP-adress: 12.146.178.5
  - Real Secure: 3 st *Email relay*
  - SNORT: 9 st *Open Port 25*
  - Kommentar: Varför Real Secure gör en bättre analys här beror på att den har flera regler för just SMTP trafik medans SNORT har enklare SMTP-regler som enbart kan detektera utnyttjande av säkerhetshål i olika SMTP-programvaror.
- IP-adress: 12.218.7.244
  - Real Secure: 2 st *TCP Port Scan*
  - SNORT: Inget larm.
  - Kommentar: Detta beror på att modulen i SNORT som ska detektera portscanningar är som standard satt till en låg känslighet, eftersom den annars ger för mycket felaktiga larm.
- IP-adress: 24.18.161.126
  - Real Secure: 17 st *TCP Port Scan*
  - SNORT: 2 st *ICMP Ping*, 10 st *SNMP*, 4 st *TCP Port Scan*, 32 st *Open Port*
  - Kommentar: Analysen av denna adress gav en antydan till varför snort hade larmat för för flera attacker då SNORT inte är lika duktig på att sammanställa flera paket till ett larm. Istället blev det en spridd skur av olika typer av larm, vilket beror på att modulen som sköter detektering av portscanning är sämre än motsvarande i Real Secure.
- IP-adress: 24.121.10.84
  - Real Secure: 2 st *TCP port scan*
  - SNORT: Inget larm.
  - Kommentar: Detta är ännu ett tecken på att Real Secure har en bättre modul för detektering av portscanningar.
- IP-adress: 216.75.191.122
  - Real Secure: 379 st *Ping flood*, 42 st HTTP-attacker, 1 st *TCP port scan*, 2 st SMTP-attacker, 25 st *Portmaster Reboot*

- SNORT: 4410 st *ICMP Large Packet*, 12 st *Open port*, 65 st HTTP-attacker

- Kommentar: Då 76% av larmen var för IP adressen 216.75.191.122 antog vi att denna adress kunde visa på vad det var som gjorde snort så duktig på att larma. Här har vi räknat samman attackerna som var HTTP-baserade. Vi kunde inte någon klar anledning till varför antalet HTTP-attacker som SNORT hade detekterat var en tredje del större än Real Secure men gissar på att det kan bero på att SNORT har flera regler för HTTP-attacker. Ovan ser man också varför SNORT genererade ett så stort antal larm. För varje stort ICMP-paket som IP-adressen skickade, så genererades ett larm i SNORT, medan Real Secure istället valde att skicka en tiondel färre larm om *Ping flood*. Real Secure har här ett mycket bättre sätt att larma som är mer resistent mot en överlastningsattack. Dock redovisar inte Real Secure hur många ICMP-paket varje *Ping flood*-larm bestod av och det får anses som en större brist. Denna IP-adress hade också skickat 25 stycken paket som Real Secure detekterade som *Portmaster Reboot*[2], SNORT detekterade inget. Detta beror på att SNORT inte har någon regel för just den attacken.

Sammanfattningsvis kan man säga att Real Secure har mera genomarbetade moduler som ser till att det inte skapas så många enskilda larm för varje paket utan försöker utifrån kontexten ge ett larm för varje typ av attack, till exempel ett larm för en *Ping flood*, där SNORT larmar för varje ICMP paket, och ett larm för varje *TCP port scan*, där SNORT larmar för varje försök till anslutning på en port. Regeldatabasen hos de båda IDS:erna verkar vara i stor sett lika utvecklad men man kan finna vissa svagheter hos SNORT som till exempel analys av SMTP-trafik. SNORT har inte någon specifik modul för analys av SMTP-trafik utan använder enbart regler för analys, medan Real Secure verkar använda sig av både regeldatabasen och en modul för analysering av SMTP-trafiken. En stor fördel med SNORT är att man själv kan utveckla moduler för analys av olika typer av protokoll, denna möjlighet har man inte med Real Secure på grund av den stängda källkoden.

4 Slutsats

4.1 Svar på vår frågeställning

4.1.1 Vilka utvärderings- och inlärningsdata finns tillgängliga och vad håller de för kvalité?

De utvärderings- och inlärningsdata som finns idag är framförallt DARPA 98/99. Det är de enda datamängder vi hittat som är kategoriserade över typer av attacker och liknande. De håller en hög kvalité, men är tyvärr till åren komna och börjar bli för gamla. LARIAT är ett projekt för att skapa en nyare och mer lättanvänd version av DARPA, men tyvärr har inte LARIAT släppts till allmänheten ännu. Förutom dessa finns också olika former av Capture-the-flag-data. Dessa är sällan kategoriserade, men går bra att använda för att utvärdera olika systems svar på attackdata.

Vi har kommit fram till att testdatat i DARPA är för gammalt för att kunna användas i praktiken idag. Attackerna i testdatat är simpla med dagens mått mätt, medans bakgrundstrafiken inte är riktigt lika föråldrad. Människor surfar på ungefär samma sätt som för sju år sedan. Det man saknar i bakgrundstrafiken är till exempel P2P-trafik för nedladdning av stora mängder filmer och musik som blivit mycket vanligare idag jämfört med 1998/99.

LARIAT är nyare, men innehållet är trots det fyra år gammalt, så det är också oanvändbart. LARIAT har varit färdig länge, men vi har inte sett några tecken på att det någon gång kommer att bli tillgängligt utanför utvecklingsgruppen. Detta gör att personer som vill göra nya testverktyg som använder samma testmetodik som LARIAT inte kan lära sig av LARIAT. Vi tror också att om LARIAT hade släppts när det var nytt och aktuellt så hade det kunnat fortleva till idag på grund av de verktyg som ingår i LARIAT för att lägga till nya data.

Det enda moderna testdata som finns tillgängligt enligt oss är Capture-the-flag-data. Nackdelarna är att det är mycket svårt att få någon struktur på testerna eftersom datamängden inte är strukturerad. CTF-data innehåller också en högre densitet av attacker, vilket medför att man inte får tillräckligt med bakgrundstrafik för att till exempel simulera höghastighetstester. Det är orimligt att spela upp CTF-data i en gigabit per sekund då 90% av trafiken kommer att innehålla attacker. Sådan trafik är orimlig i verkligheten och testar egentligen bara IDS:ens förmåga att hantera överlastningsattacker.

4.1.2 Vad finns det för testprogram och testmetoder för att undersöka kvalitén hos en IDS?

För att testa en modern IDS behöver man moderna verktyg, men de flesta verktyg som refereras till i forskningsrapporter är mycket gamla eller har en långsam utvecklingsgång. Många verktyg är inte heller publikt tillgängliga utan finns bara för en begränsad skara forskare inom IDS-området. Dock så finns det några Open Source-verktyg som har en aktiv utveckling, till exempel Nmap, Hping och Tcpreplay som vi beskrivit i rapporten. Tcpreplay har vi använt i vårat praktiska test och funnit att det är en mycket kraftfullt verktyg att använda för manipulering av inspelad trafik. Tyvärr hann vi inte utföra några tester med Hping, men det är ett mycket intressant verktyg som kom-

[2] ett specifikt genererat paket för att crasha en viss typ av OS

mer att hjälpa utvecklare av IDS att skapa test-scripts för att utvärdera sina IDS:er.

Den metodik som användes vid DARPA- och LARIAT-evalueringen anser vi vara fullt fungerande. Om nyare attacker och bakgrundstrafik skulle genereras idag på samma sätt, samtidigt som man skulle inkludera hjälpverktyg så som LARIAT har, så skulle det utgöra en bra grund för att utföra tester av IDS:er.

4.1.3 Var ligger forskningsfronten idag?

Vår uppfattning är att det finns ett stort glapp mellan forskningsfronten och faktiskt implementerade system. Forskningen verkar röra sig mycket inom ett teoretiskt drömland där alla högt svävande idéer alltid går att implementera på magiskt vis. Vissa forskningssystem har på något sätt implementerats, oftast som en modul till SNORT eller som ett litet testprogram. De är dock starkt begränsade och skulle inte klara av att hantera ett riktigt datorsystem med flera användare användare och stor mängd trafik.

De faktiskt användbara IDS:er som vi funnit är alla signaturbaserade. Vi tycker det är tråkigt att ingen storskalig forskning verkar göras inom området signaturbaserade IDS:er, eftersom det är där som den faktiska användaren och reella problem kommer in. Det vore också intressant om någon kommersiell leverantör eller annan aktör ville implementera några av forskningsmodellerna som involverar självlärande system eller liknande.

4.1.4 Hur bra står sig IDS:er från Open Source-aktörer, kommersiella företag och forskningsdomänen mot varandra?

Forskningsdomänen står sig mycket dåligt, eftersom vi inte hittat några implementerade modeller som fungerar i ett modernt test med större datamängder. Det finns ett fåtal äldre implementationer men dessa har inte något större värde då de slutade utvecklas i slutet på 90-talet. Om det hade funnits flera testimplementationer publikt att tillgå så kanske kommersiella företag hade tagit vara på de nya teknikerna snabbare och på så sätt tror vi att utvecklingen inom området hade gått snabbare.

Inom det kommersiella området är det egentligen bara de större tillverkarna av nätverksutrustning som till exempel Cisco som har byggt in IDS-funktionalitet i sina produkter. Det finns också många företag som har tagit SNORT och utvecklat egna gränssnitt för konfiguration och övervakning för att sedan paketerat det hela i speciell hårdvara för att klara av högre hastigheter på trafiken. Detta gör att det finns många snarlika produkter som inte ger någon spridning av olika tekniker inom området och det anser vi vara en stor nackdel.

Ett av de få företag som inte verkar ha utgått från SNORT är Internet Security Systems (ISS) med deras Real Secure. Deras Hybrid IDS är signaturbaserad med ett välutvecklat gränssnitt och en distribuerad arkitektur med sensorprogramvara för både Windows, Solaris och Linux som både kan lyssna på nätverkstrafik och systemloggar. Efter ha använt Real Secure i vårt praktiska test så tycker vi det verkar vara en väl genomarbetad produkt, och då ISS lämnade ut evalueringsversion utan krav på speciell hårdvara tycker vi att det var lätt att få en uppfattning om hur bra den fungerade.

Den Open Source-IDS vi testade var SNORT kombinerat med gränssnittet BASE. SNORT är precis som Real Secure en signaturbaserad NIDS, och som analysator av trafik så är den både snabb och fungerar bra. Signaturdatabasen är också välutvecklad med ständiga uppdateringar som går att ladda ner gratis från deras hemsida. Tyvärr finns det inte många andra Open Source-IDS:er än SNORT, vi har nämnt Prelude i vår rapport men det är också en av de få andra som utvecklas aktivt.

En nackdel med SNORT är att det inte ingår något gränssnitt i produkten, utan användaren är tvungen att skriva långa regler i konfigurationsfiler och övervaka via konsollen. Det finns dock flera verktyg för att skapa grafisk övervakning och konfiguration för SNORT, som till exempel BASE, men dessa utvecklas av helt andra grupper, och att hitta information om att de finns kan vara svårt. Dessa gränssnitt känns inte heller lika lättanvända och välutvecklade som ISS Real Secure, utan kräver ingående kunskap om hur olika protokoll och attacker fungerar.

4.1.5 Hur svårt är det att genomföra ett praktiskt test av olika IDS:er?

Inom det kommersiella området är det mycket svårt att få ut evalueringsprogramvara som innehåller en full signaturdatabas. Den enda vi hittade är ISS Real Secure. Många kräver också speciell hårdvara. Det gör det mycket svårt att utvärdera om de kommersiella produkterna verkligen fungerar som de utgör sig för att göra. På Open Source-området är det lättare att sätta upp testsystem, men då kommer man snart till nästa problem, och det är att få tag på bra testdata, vilket just nu inte finns. Till sist är det problem med att jämföra resultaten från ett test på ett mer ingående vis än att bara jämföra sifferstatistik. Detta på grund av att olika program har olika namn på samma attack, och det går därmed inte att automatisera jämförelseprocessen på något sätt.

4.2 Nuvarande problem inom området

Det vi sett som största problemet inom området att konstruera en IDS verkar vara att datorer inte innehåller någon intelligens. Om man satte en person att hela tiden övervaka systemet så skulle personen själv kunna fatta beslut om vad

som är ett intrångsförsök och vad som inte är det. En dator däremot måste programmeras för varje fall.

Det finns många rapporter idag som beskriver funktionalitet hos olika 'smarta' IDS:er. Det finns allt från neurala nätverk som ska vara självlärande till system som kopierar det mänskliga immunförsvaret. Vi har dock inte hittat någon IDS som faktiskt implementerar några av dessa 'smarta' funktioner. Kanske är bara datorkraften för svag idag för att klara av att köra dem, men vi misstänker också att flera rapporter inte är skrivna av personer som faktiskt tänkt på om det är implementationsbart eller inte.

Som vi skrivit i kapitel 2.5 så klarar de bästa systemen av att detektera ungefär 70% av alla attacker i ett test. Många system, speciellt de som är signaturbaserade, är föråldrade redan när de kommer ut på grund av den oerhörda hastighet med vilken nya hål i programvara upptäcks.

4.3 Fortsatt arbete

En intressant fråga vi stötte på är hur Prelude, som vi skrev om i kapitel 3.1.1, står sig mot andra IDS:er i ett test. Vi skulle gärna se en undersökning av Prelude. Prelude är precis som ISS Real Secure en hybrid-IDS, och ett jämförande test av de värdbaserade delarna skulle kunna vara intressant. För att göra ett liknande test rekommenderar vi användning av riktig hårdvara. Användning av User-mode Linux resulterar i maskiner som är alltför klena för att användas till den här typen av resurskrävande applikationer såsom SNORT.

Det testdata som finns tillgängligt idag är inte lämpat för den här typen av tester. Man vill ha färskt testdata som är kategoriserat efter typ av attack och har bra verktyg för att skicka egna sekvenser av attacker. Detta i kombination med normal trafikdata ger testet en korrekt bild av hur IDS:erna fungerar i praktiken. Ett förslag på framtida arbete inom området är att uppföra nya kategoriserade testdatamängder, och vi tror att LARIAT kan vara en bra grund att fortsätta bygga på.

En annan möjlighet är att försöka låna hårdvara från tillverkarna för att utvärdera den. Detta kombinerat med bra testdata skulle vara mycket intressant att se. Om någon skulle utveckla en output-modul till SNORT som alarmerar i format enligt RFC 3067 eller annan standard så vore det välkommet, speciellt om andra tillverkare också skulle börja använda sig av samma standard.

Referenser

[1] N. Athanasiades m.fl.: *"Intrusion Detection Testing and Benchmarking Methodologies"* First IEEE International Workshop on Information Assurance (IWIA'03) (2003)

[2] E. L. Barse: *"Logging for intrusion and fraud detection"* PhD Thesis. Chalmers University of Technology, Goteborg, Sweden (2004)

[3] E. L. Barse, E. Jonsson: *"Setting the scene for intrusion detection"* Technical report 04-05, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden (2004)

[4] F. Biancuzzi: *"Network Tool Development with hping3"* `http://www.onlamp.com/lpt/a/5236` 2005-04-27 (2004)

[5] S. Biles: *"Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE)"* Technical Report TR2004-485, Department of Computer Science, Dartmouth College, Hanover, USA (2003)

[6] M. Bishop: *"Introduction to Computer Security: Chapter 22"* `http://nob.cs.ucdavis.edu/book-intro/slides/22.pdf` 2005-04-27 (2004)

[7] R. Chinchani, A. Muthukrishnan, M. Chandrasekaran: *"RACOON: Rapidly Generating User Command Data For Anomaly Detection From Customizable Templates"* 20th Annual Computer Security Applications Conference (ACSAC 2004) (2004)

[8] horizon: *"Defeating Sniffers and Intrusion Detection Systems"* `http://www.phrack.org/phrack/54/P54-10` 2005-04-24 (1998)

[9] W. Jing-xin, W. Zhi-ying, D. Kui: *"A Network Intrusion Detection System based on the Artificial Neural Networks"* ACM International Conference Proceeding Series, Proceedings of the 3rd international conference on Information security (2004)

[10] Z. Junzhong, H. Houkuan: *"An Evolving Intrusion Detection System Based on Natural Immune System"* TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering (2002)

[11] W. Lee, S. J. Stolfo: *"Data Mining Approaches for Intrusion Detection"* 7th USENIX Security Symposium (1999)

[12] W. Lee, D. Xiang: *"Information-Theoretic Measures for Anomaly Detection"* Security and Privacy Proceedings. 2001 IEEE Symposium (2001)

[13] P. A. Porras, P. G. Neumann: *"EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances"* Proc. 20th National Information Systems Security Conference (1997)

[14] T. H. Ptacek, T. N. Newsham: *"Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection"* Technical Report, Secure Networks, Inc. (1998)

[15] L. M. Rossey m.fl.: *"*LARIAT*: Lincoln Adaptable Real-time Information Assurance Testbed"* DARPA Information Survivability Conference and Exposition II, volume 1, IEEE (2001)

[16] B. Song, M. Ye, J. Li: *"Intrusion Detection Technology Research based High-Speed Network"* Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003 (2003)

[17] Sun Microsystems: *"Solaris 8 System Administrator Collection – SunSHIELD Basic Security Module Guide"* `http://docs.sun.com/app/docs/doc/806-1789` 2005-04-27 (2000)

[18] D. Zamboni m.fl.: *"An Architecture for Intrusion Detection using Autonomous Agents"* Proceedings of the 14th Annual Computer Security Applications Conference (1998)

[19] Basic Analysis and Security Engine (BASE) project `http://secureideas.sourceforge.net/` 2005-05-04

[20] Defcon `http://www.defcon.org/` 2005-05-01

[21] Enterasys Intrusion Defence `http://www.enterasys.com/products/ids/` 2005-04-27

[22] Ethereal: A Network Protocol Analyzer `http://www.ethereal.com/` 2005-05-01

[23] FRAGROUTER (8) manual page `http://packetstorm.widexs.nl/UNIX/IDS/nidsbench/fragrouter.html` 2005-04-27

[24] hping security tool – home page `http://www.hping.org/` 2005-04-27

[25] IDSwakeup `http://www.hsc.fr/ressources/outils/idswakeup/` 2005-04-27

[26] Internet Security Systems – Proventia Enterprise Protection `http://www.iss.net/products\delimiter"026E30F_services/enterprise\delimiter"026E30F_protection` 2005-04-27

[27] The Libnet Packet Construction Library `http://www.packetfactory.net/Projects/Libnet/` 2005-05-01

[28] MIT Lincoln Laboratory – DARPA Intrusion Detection Evaluation `http://www.ll.mit.edu/IST/ideval/` 2005-05-01

[29] Nmap – Free Security Scanner For Network Exploration & Security Audits `http://www.insecure.org/nmap/` 2005-04-27

[30] Oinkmaster `http://oinkmaster.sourceforge.net/` 2005-05-04

[31] Prelude-IDS – The Hybrid IDS framework `http://www.prelude-ids.org/` 2005-04-27

[32] The Schmoo Group – Capture the RootFu `http://www.shmoo.com/cctf/` 2005-05-01

[33] Server Break-in Challenge `http://www.linuxense.com/challenge/` 2005-05-01

[34] SNORT – the de facto standard for intrusion detection/prevention `http://www.snort.org` 2005-05-01

[35] TCPDUMP Public Repository `http://www.tcpdump.org` 2005-04-27

[36] Tcpreplay: Pcap editing and replay tools for *NIX `http://tcpreplay.sourceforge.net` 2005-04-27