Parametrar för intrångsanalys

MARTIN KARRESAND

FOI
Totalförsvarets forskningsinstitut
Ledningssystem
Box 1165
581 11 LINKÖPING

Tel: 013-37 80 00
Fax: 013-37 81 00

www.foi.se

FOI-R--1831--SE
ISSN 1650-1942

Teknisk rapport
December 2005

Ledningssystem

Martin Karresand

Parametrar för intrångsanalys

FOI-R--1831--SE
ISSN-1650-1942

Ledningssystem
Teknisk rapport

December 2005

Utgivare	Rapportnummer, ISRN	Klassificering
Totalförsvarets Forskningsinstitut – FOI Ledningssystem Box 1165 581 11 LINKÖPING	FOI-R--1831--SE	Teknisk rapport
	Månad år December 2005	**Projektnummer** E 7091
	Forskningsområde Ledning, informationsteknik och sensorer	
	Delområde Ledning med samband, telekom och IT-system	
	Delområde 2	

Författare	Projektledare
Martin Karresand	Mikael Wedlin
	Godkänd av Martin Rantzer Chef, Ledningssystem
	Tekniskt och/eller vetenskapligt ansvarig Johan Allgurén IC, Institutionen för Systemutveckling och IT-säkerhet
	Uppdragsgivare/kundbeteckning FM

Rapporttitel
Parametrar för intrångsanalys

Sammanfattning

Den här rapporten analyserar 11 konferensbidrag som behandlar anomalibaserad intrångsdetektering från Recent Advances in Intrusion Detection (RAID) åren 2000-2004, samt 9 konferensbidrag från IEEE Symposium on Security and Privacy (S&P) åren 2000-2005. Analysen görs med avseende på de parametrar som respektive algoritm använder som bas för att detektera intrång. Dessutom presenteras och analyseras ytterligare fyra bidrag som undersöker om det går att använda bytefrekvensfördelning hos ett binärdatafragment för att avgöra vad det tillhör för filtyp.

Nyckelord
intrångsanalys, intrångsdetektering, parametrar

Övriga bibliografiska uppgifter
Omslagsbild: Martin Karresand, 2005

ISSN	Antal sidor	Språk
ISSN-1650-1942	33	Svenska

Distribution	Pris
Enligt missiv	Enligt prislista
	Sekretess Öppen

Issuing organisation	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Command and Control Systems P.O. Box 1165 SE-581 11 LINKÖPING	FOI-R--1831--SE	Technical report
	Month year December 2005	**Project number** E 7091
	Research area code C^4ISTAR	
	Sub area code C^4I	
	Sub area code 2	

Author(s)	
Martin Karresand	**Project manager** Mikael Wedlin
	Approved by Martin Rantzer Head, Command and Control Systems
	Scientifically and technically responsible Johan Allgurén Head, Systems Development and IT Security
	Sponsoring agency FM

Report title

Parameters for Intrusion Analysis

Abstract

This report presents an analysis of 11 papers treating anomaly based intrusion detection from Recent Advances in Intrusion Detection (RAID) published between 2000 and 2004, and 9 papers from the IEEE Symposium on Security and Privacy (S&P) published between 2000 and 2005. The analysis is performed with the aim to find what parameters are used as a basis to detect intrusions from. In addition to that four more papers are presented and analysed. The papers are treating the question of whether it is possible to use the byte frequency distribution of a data fragment to correctly classify it as belonging to a specific file type.

Keywords

intrusion analysis, intrusion detection, parameters

Further bibliographic information

Cover photo: Martin Karresand, 2005

ISSN	Pages	Language
ISSN-1650-1942	33	Swedish

Distribution	Price
By sendlist	According to price list
	Security classification Unclassified

Innehåll

1 Inledning

I och med att interaktionsmöjligheterna i datorsystemen blir allt mer avancerade, tillsammans med att fler och fler noder ingår i nätverken gör att detektion av intrång och missbruk av systemen blir svårare att upptäcka. Samhällets och militärens ständigt ökande beroende av sina datorsystem gör att behovet av att kunna detektera intrång blir allt större. Tyvärr medför den ökade komplexiteten hos systemet som interaktionsmöjligheterna och systemstorleken innebär att det i motsvarande grad blir svårare att korrekt och snabbt kunna upptäcka intrång.

En viktig, om inte till och med den viktigaste faktorn som avgör kvaliteten på intrångsdetekteringen är de parametrar i form av sensordata som utgör grunden för detekteringen. Oavsett vilken algoritm som används för att detektera intrång blir aldrig detektionsgraden bättre än vad parametrarna tillåter.

De parametrar som är relevanta för intrångsdetektering är också användbara vid intrångsanalys. Dessa två områden är nära besläktade och den enda skillnaden är egentligen att intrångsanalys inte har samma möjlighet att använda flyktiga data från till exempel RAM-minne.

Ett område som bedöms vara mycket intressant är det som behandlar tekniker för att kunna kategorisera fragment av binärdata enbart utifrån strukturen på fragmenten. Därför har en fördjupning gjorts inom detta område genom att tre extra konferensbidrag, samt ett utkast till en teknisk rapport har analyserats i ett separat kapitel.

1.1 Bakgrund

Den här rapporten är milstolpe för 2005 års arbete i projektet “Strid i IT-domänen”, beställd av Försvarsmakten inom ramen för FoT.

Meningen är att rapporten ska utgöra underlag för nästa års arbete då “Intrångsanalys i praktiken” förhoppningsvis kan genomföras. Delar av det arbetet har redan utförts och kommer på något sätt att ingå i nästa års rapport.

1.2 Avgränsning

Rapporten har avgränsats till att omfatta en beskrivning av de parametrar som använts för anomalidetektering. Ett urval av parametrar har gjorts genom att studien omfattar 11 konferensbidrag från Recent Advances in Intrusion Detection (RAID) [1] åren 2000–2004, samt 9 konferensbidrag från IEEE Symposium on Security and Privacy (S&P) [2] åren 2000–2005.

De parametrar som använts är ofta inte beskrivna i detalj och i de flesta fall saknas någon djupare motivering av varför just de parametrar som använts har valts. Detta begränsar delvis djupet på rapporten i den del som behandlar ett urval av de vanligare

parametrarna som använts för närvarande. Det går därför ej heller att göra mer än kvalificerade gissningar av skälen till valen av parametrar.

Vad gäller den fördjupade studien av binärdatakategorisering har en begränsning av omfattningen gjorts utifrån den inriktning och teknik som bidragen använder. Endast bidrag med direkt inriktning mot att kategorisera binärdatafragment har tagits med. Likaså finns det ännu ej publicerat, men till en konferens inskickat, material som på grund av reglerna för konferenspublicering inte kan redovisas i denna rapport. Materialet berör dock utveckling och tester som fyller igen vissa hål som bidragen om binärdatakategorisering lämnar.

2 Parametrar

Det här kapitlet presenterar först kort de viktiga delarna ur de konferensbidrag som studerats och sedan beskrivs parametrarna, indelade i tre huvudkategorier. Kategorierna utgår från om parametrarna använts för nätverksbaserad intrångsdetektering, systembaserad intrångsdetektering eller tillämpningsbaserad intrångsdetektering. Huvudområdena, som presenteras i varsitt avsnitt, har sedan ytterligare brutits ner i underområden.

2.1 Konferensbidrag

Bidrag från två stora konferenser, RAID och S&P, har använts för studien. RAID är ansedd som den främsta konferensen inom intrångsdetekteringsområdet och S&P är en välrenommerad konferens med långa anor inom IT-säkerhetsområdet i allmänhet. Varje bidrag presenteras i ett eget avsnitt och i kronologisk ordning med början med det äldsta.

Samma urval av bidrag har även använts i två ej publicerade rapporter [3, 4] från Linköpings universitet. Dessutom har ett bidrag från S&P 2005 tagits med.

2.1.1 Logic Induction of Valid Behavior Specifications for Intrusion Detection

Calvin Ko skriver i ett bidrag från S&P [5] om en metod att automatiskt skapa specifikationer för hur program får uppföra sig. Det han främst inriktar sig på är att skapa vad han kallar “valid access specifications” [5, sid. 9] som är uttryck av första ordningens logik. De kan representera sådan operationer som läsning och skrivning av filer, eller att skapa i-noder i filsystemet.

Detektering av anomalier sker på systemanropsnivå och Ko betonar vikten av att även använda information om processattribut, till exempel process ID och processägarens användaridentitet. Likaså bör objektattribut användas och då nämner han som exempel rättighetsinställningar. Också information i form av systemanropsargument, sökvägar och vem som skapade ett objekt är användbar information enligt honom.

2.1.2 Adaptive, Model-Based Monitoring for Cyber Attack Detection

Valdes och Skinner från SRI skriver i ett konferensbidrag från RAID 2000 [6] om en metod att skapa modeller av attackklasser. De beskriver inte parametrarna som används i detalj, men i sammanfattningen nämner de att det är TCP-pakethuvuden som ligger till grund för detekteringen. I bidraget står det sedan att det är TCPdump-

data[1] som används, vilket kan innebära att det är TCPdump i grundinställning vilket ger de första 68 byten av paketen.

2.1.3 A Real-Time IDS Based on Learning Program Behaviour

Det här konferensbidraget av Ghosh et al [8] från RAID 2000 är inriktat på att beskriva tre algoritmer från machine-learningområdet och hur dessa använts för intrångsdetektering. Algoritmerna har utvärderats med hjälp av data från DARPA 1999. Eftersom Ghosh et al inriktar sig mer på algoritmerna än vilka parametrar de använder är beskrivningen av parametrarna av förklarliga skäl knapphändig. Det som beskrivs är att de använder data från Basic Security Module (BSM)[2] för detekteringen.

De tre algoritmer som används är ett Elman Recurrent neuralt nätverk, en strängtransducer och en ändlig tillståndsmaskin. Ett Elmannätverk är byggt enligt feedforwardprincipen, men har dessutom ett antal noder som tar värden från en dold nod och sedan återkopplar värdet till alla noder i det dolda lagret.

Författarnas strängtransducer tar en serie symboler som indata och tittar på sannolikheten för olika utdatasymboler. De använder n-gram av BSM-sekvenser som indata. Algoritmen ger sedan olika l-gram av BSM-data från indatamängden, där $l < n$. Genom att räkna antalet olika l-gram som förekommer får författarna fram sannolikheten för de olika l-grammen.

Den tredje algoritmen är en vanlig ändlig tillståndsmaskin som tränas med normala data och sedan trimmas så att den detekterar avvikelser, men fortfarande tolererar nya ofarliga sekvenser av data.

2.1.4 Flexible Intrusion Detection Using Variable-Length Behaviour Applied to CORBA Objects

Marrakchi et al [9] beskriver ett intrångsdetekteringssystem på tillämpningsnivå där de tittar på CORBA-objekt. De parametrar de använder är baserade på klientsidans funktionsanrop till CORBA-objekt på serversidan. Både funktionsanropet i sig och de argument det tar används vid detekteringen. Genom att skapa modeller med toleransnivåer för argumenten blir algoritmen mer flexibel.

De exakta parametrarna som används är implementationsspecifikt och beskrivs därför inte. Varje session mellan klient och server betraktas som en sekvens och är modellerad i form av ett träd där roten är uppkopplingen och löven är nedkopplingen av sessionen.

2.1.5 Intrusion Detection via Static Analysis

Wagner och Dean [10] har inte någon mer detaljerad beskrivning av de parametrar de använder för sitt systembaserad intrångsdetekteringssystem. De parametrar som nämns är systemanrop i form av 2-gram. För att förenkla modellen tar de bara hänsyn till sekvenser av systemanrop och bortser från innehåll i lokala variabler, datastrukturer och andra dataflödesrelaterade parametrar.

När de sedan beskriver implementationen använder de mer än bara systemanropssekvenser eftersom de även tar hänsyn till de signaler som en process kan få ta

[1] TCPdump[7] är ett program för insamling av hela eller delar av nätverkspaket. Det finns versioner för både Unix/Linux och Windows.

[2] BSM är en övervaknings- och loggningsfunktion för processer i operativsystemets kärna. BSM kommer från SUN Solaris, men finns även implementerat för Linux.

emot. Det höjer precisionen i detekteringen enligt författarna. De diskuterar även att ta med de argument som systemanropen tar och nämner att de fått lovande resultat när de gjorde ett rudimentärt test där de tog hänsyn till systemanropsargument.

2.1.6 Data Mining Methods for Detection of New Malicious Executables

Det här bidraget från S&P 2001 av Schultz et al [11] beskriver hur data mining kan användas för att upptäcka elakartad kod. Schultz et al använder tre grupper av parametrar, Dynamic Link Library (DLL)-relaterade data, sekvenser av skrivbara tecken och hexadecimala bytesekvenser.

Den första gruppen av parametrar gäller bara för filer i Portable Executable (PE)-format, det vill säga exekverbara filer. Författarna använde exekverbara filer från Windowsmiljö och extraherade data ur dem med hjälp av funktioner i Linux (GNU Bin-Utils). Tre typer av data från PE-filerna extraherades för datamängden som bestod av

1. en lista med DLL:er som användes av binären,
2. en lista med DLL-funktionsanrop binären gjorde och
3. antalet funktionsanrop som binären använde från varje DLL.

Datamängden kom i sin helhet från PE-huvudet. Parametrarna omvandlades sedan till vektorform, där varje position beskrev om en parameter användes eller ej, eller antalet funktionsanrop. Sedan jämfördes vektorerna med beteendet hos systemet i realtid för att upptäcka anomalier, det vill säga att varje program som kördes kontrollerades så att enbart de DLL-relaterade parametrar som beskrevs av vektorerna användes. De här metoderna övergavs dock eftersom de inte gick att applicera på hela mängden av elakartad kod.

Nästa parametergrupp bestod av specifika strängar av skrivbara tecken som var unika för elakartade binärer, respektive ofarlig kod. Författarna påpekar att den här metoden redan används av antivirusprogramtillverkare. Några exempel på typer av strängar de hittade i binärerna var fragment av återanvänd kod, signaturer från den eller de som gjort programmet, filnamn och systemrelaterad information.

Den sista parametergruppen de använde fick de genom att extrahera unika hexadecimala sekvenser ur binärerna, liksom för fallet med strängar antog de att det fanns unika sekvenser för elakartad kod respektive ofarlig kod. Det här är också det en metod som redan används av anti-virusprogramtillverkare.

2.1.7 Information-Theoretic Measures for Anomaly Detection

I det här konferensbidraget undersöker Lee och Xiang [12] om det går att använda metriker från informationsteoriområdet för att detektera intrång. Det gör att de parametrar de använt får en nedtonad betydelse och enbart används för att utvärdera metrikerna. Parametrarna som används kan delas in i tre grupper, systemanrop, BSM-data och TCPdump-data.

Systemanropen som används i undersökningen kommer från *sendmail*. De formateras som n-gram där $n \in \{3, 4, \ldots, 19\}$ och sedan mäts olika former av entropi.

Enligt Lee och Xiang har BSM-datamängden den fördelen jämfört med systemanropen att den bland annat också innehåller information om verklig och faktisk

användare och användargrupp, namnet på det objekt som anropas och de argument som används vid anropet.

Den tredje gruppen av parametrar kommer från TCP-dump och framställs med hjälp av en modifierad version av Bro [13]. Den resulterande datamängden består av sessioner och innehåller en tidstämpel, längden på sessionen, avsändarport och -adress och ett kombinerat fält kallat service som innehåller mottagarport och protokoll. Dessutom ingår mottagaradress, antal bytes som skickats från avsändare till mottagare och tvärt om, samt de flaggor som används i paketen.

2.1.8 A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors

Sekar et al [14] presenterar ett systembaserat intrångsdetekteringssystem som använder systemanrop och anropsadressinformation. Adressinformationen för varifrån ett systemanrop görs får de genom att spara värdet på programräknaren vid varje systemanrop.

De använder sig även av ytterligare information som är tillgänglig från processstacken. Sekar et al förklarar att strukturen på de *activation frames* som stacken består av är standardiserad vad det gäller struktur och fältposition. Därför kan författarna också använda parametrar såsom returadress, processattribut och lokala processvariabler. Förutom information från processstacken använder de även (hårdvaru-)registerinnehåll.

Deras system klarar av att hantera dynamiskt laddade bibliotek. Genom att kontrollera om en påträffad länkning är statisk eller inte och om den inte är det ta nästa adress från stacken och upprepa kontrollen kan de arbeta sig fram till ett statiskt anrop.

2.1.9 CDIS: Towards a Computer Immune System for Detecting Network Intrusions

Williams et al [15] presenterade ett förslag på nätverksbaserat artificiellt immunförsvar för datorsystem vid RAID 2001. Systemet använder de första 40 byten i IP, TCP, UDP och ICMP-paket för att detektera intrång. Skälet till att inte de inte använder även datadelen i paketen var att detektionsalgoritmen då bedömdes bli för komplicerad. De skriver dock att detta planeras ingå i framtida forskning.

Antalet parametrar anges till 28 för TCP-paket och 16 för UDP respektive ICMP. Parametrarna från IP-delen av pakethuvudet är alla gemensamma för de olika pakettyperna. Detta gäller även för andra delar av pakethuvudena, till exempel räknas varje individuell flagga i TCP-huvudet som en separat parameter, men även det faktum att alla flaggor är satta räknas som en parameter. Det är även värt att notera att varje del i IP-adresserna som används betraktas som en separat parameter, det vill säga en hel IP-adress består av fyra parametrar. Det gör att det faktiska antalet parametrar som används sjunker. Tabell 2.2 visar de parametrar som används för varje pakettyp.

Från dessa parametrar plockas mellan 2 och 7 slumpvis valda parametrar ut. De får även slumpvis valda värden, så länge dessa ligger inom de tillåtna områdena. Parametrarna används sedan för att skapa virtuella antikroppar som tillsammans bygger upp mängden av alla otillåtna pakethuvuden. Med andra ord tillämpas en omvänd anomalidetektering genom att det är mängden av alla otillåtna tillstånd som modelleras.

Tabell 2.1: Paketrelaterade parametrar för respektive pakettyp

Typ	Parameternamn	Giltiga värden
IP	Protocol type	TCP, UDP, ICMP
IP	Identification Number	0 till $2^{16} - 1$
IP	Time To Live	0 till $2^{8} - 1$
IP	Flags	0 till $2^{16} - 1$
IP	Overall Packet Length	0 till $2^{16} - 1$
IP	Source Address (A.B.C.D)	Giltig IP adress
IP	Destination Address (A.B.C.D)	Giltig IP adress
TCP	Source Port	0 till $2^{16} - 1$
TCP	Destination Port	0 till $2^{16} - 1$
TCP	Sequence Number	0 till $2^{32} - 1$
TCP	Next Sequence Number	0 till $2^{32} - 1$
TCP	Acknowledgement Number	0 till $2^{32} - 1$
TCP	All Flags	0 till $2^{8} - 1$
TCP	Flags (PCWR, Echo, Urgent, Ack, Push, Reset, Syn, Fin)	0 till 1
TCP	Packet Size	0 till $2^{16} - 1$
UDP	Source Port	0 till $2^{16} - 1$
UDP	Destination Port	0 till $2^{16} - 1$
UDP	Data Length	0 till $2^{16} - 1$
ICMP	Type	0 till $2^{8} - 1$
ICMP	Code	0 till $2^{8} - 1$
ICMP	Data Length	0 till $2^{16} - 1$

2.1.10 Monitoring Anomalous Windows Registry Access

Apap et al [16] bygger sin intrångsdetektor på antagandet att de flesta virus, maskar och annan elakartad kod på något sätt använder Windows Registry, hädanefter kallat registret, för sitt Registry Anomaly Detection (RAD)-system. Enligt författarna finns det två nycklar [16, sid. 40] i registret som alltid eller nästan alltid används av elakartad kod. Den första nyckeln, *HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\processName* används alltid. Varje gång en tillämpning körs kontrollerar Explorer processnamnet via den nyckeln. Även i *auto-run*-delen av registret, i nyckeln *HKLM\Software\ Microsoft\Windows\ Current Version\Run*, är det många virus, maskar och trojaner som skriver information.

Metoden utnyttjar också hur installation av nya program går till. Utförs installationen utan inblandning av Install Shield Wizard är det ett tecken på en anomali.

En annan form av anomali som också används är på vilket sätt registret används. Apap et al konstaterar att registret i normalfallet används på ett regelbundet och strukturerat sätt. De har därför en kontroll av när och vid vilka intervall olika nycklar skapas eller ändras. Enligt dem verkar vissa nycklar bara användas vid installation av operativsystemet och ska alltså inte röras efter det.

Utifrån sina observationer av viktiga nycklar i registret skapar Apap et al en datamodell för RAD, modellen innehåller fem parametrar med direkt koppling till registret [16, sid. 42].

Process: Namnet på den process som använder registret, används för att hitta processer som inte användes i träningsdatamängden.

Query: Typen av operation som utförs på registret, bara ett fåtal av de som finns används normalt, till exempel `QueryValue`, `CreateKey`, and `SetValue`.

Key: Namnet på den nyckel som operationen berör, används för att hitta registernycklar som inte förekom i träningsdatamängden.

Response: Status för operationen när den är utförd, till exempel `success`, `not found`, `buffer overflow`, och `access denied`.

Result Value: Resultatet av operationen, används för att upptäcka ovanliga resultatvärden, vilka kan vara tecken på intrång eller missbruk av systemet.

Med hjälp av denna datamodell kan processer som imiterar en legitim process upptäckas genom att de har ett avvikande namn. Författarna ger exempel på *aim.exe* och *aimrecover.exe* som båda läser lösenordet för AOL Instant Messenger-tillämpningen, men där den andra är illegitim.

2.1.11 Detecting Anomalous Network Traffic with Self-organizing Maps

Ramadas et al beskriver i en artikel [17] från RAID 2003 hur de använt Self-organizing Maps (SOM) för att detektera anomalier i nätverkstrafik. SOM:en matas med data som innehåller de 64 första byten av IP-, TCP- och UDP-paket. Om pakethuvudet inte är 64 byte fyller metoden ut resten av de 64 byten med nollor (*zero padding*).

I och med att de fyller ut byte som inte tillhör pakethuvudet använder de i praktiken de första 28 byten för ett UDP/IP-paket och de 40 första byten för ett TCP/IP-paket, vilket omfattar hela pakethuvudet om inte tillvalsfältet används. Det gör också att de parametrar de använder i mångt och mycket överensstämmer med vad som anges i Tabell 2.2.

2.1.12 Detecting Self-propagating Email Using Anomaly Detection

Gupta och Sekar [18] presenterar en algoritm för att detektera e-postmaskar i en artikel från RAID 2003. De tittar på en intern e-postservers handlingar i form av sändning och mottagning av mail för att detektera att en mask håller på att sprida sig i ett nätverk.

För att kunna se avvikelser från normalfallet skapar de frekvensfördelningar av hur ofta klienter sänder, respektive tar emot mail generellt för hela systemet. De skapar även statistik för varje enskild klient. Statistiken skapas över varierande tidsrymder för att kunna detektera även långsamma maskar. Längderna på tidsrymderna är geometriskt fördelade.

De tester av metoden som författarna gjort är alla baserade på simuleringar, vilket de pekar ut som en svaghet. Gupta och Sekar skriver dock att de planerar att utvärdera algoritmen i skarp miljö någon gång i framtiden.

2.1.13 Improving HMM-based Intrusion Detection

Cho och Han [19] utnyttjar Basic Security Module (BSM) för att hämta de parametrar som de använder för att detektera intrång med. I Tabell 2.2 presenteras de specifika parametrarna.

Tabell 2.2: Paketrelaterade parametrar för respektive pakettyp

Grupp	Parameter
Systemanrop	ID, returvärde, returstatus
Process	ID, IPC ID, IPC-rättigheter, avslutningsvärde, avslutningsstatus
Filläsning/skrivning	läs/skrivläge, sökväg, filsystem, filnamn, argumentlängd

Eftersom program i många fall tillfälligt exekveras med högre rättigheter än de användaren har erbjuds en angripare ett ypperligt tillfälle för angrepp när denna rättighetsförändring äger rum. I normalfallet återgår rättighetsnivån till de som användaren har efter att programmet utfört de systemanrop som krävde högre rättigheter. Vid till exempel en buffertöverskrivningsattack sker aldrig denna återgång och en avvikelse inträffar. Denna avvikelse är vad Cho och Han utnyttjar för intrångsdetektering. Metoden klarar även att detektera förändringar i användargruppers rättigheter.

2.1.14 An Empirical Analysis of Target-resident DoS Filters

Detta bidrag [20] till S&P av Collins och Reiter behandlar problemet med Denial of Service (DoS)-attacker. De föreslår fyra olika metoder för att filtrera bort trafik från angriparen utan att störa den normala trafiken i alltför hög grad.

Metoderna kan delas in i två grupper, en som använder sig av avsändaradressen på IP-nivå och en som använder olika former av beskrivningar av de vägar paketen tagit över Internet.

Den första gruppen skapar kluster av tillåtna IP-adresser. Det ena sättet författarna föreslår är att klustren ska bestå av de prefix enligt CIDR som kan fås ur BGP-tabellerna. Det andra sättet är att använda en statisk storlek på de prefix som klustren består av. Författarna förkastar dock metoden med statiska kluster eftersom den inte korrekt kan modellera de administrativa relationerna mellan olika nätverk.

Metoden att använda time-to-livefältet (TTL) i pakethuvudet för att beräkna antalet hopp mellan sändare och mottagare är inte felfri, författarna påpekar att det värde som fältet initieras med skiljer sig mellan olika plattformar. Förutom TTL-fältet används också de tre första oktetterna i IP-adressen för att modellera antalet hopp.

Den fjärde metoden kräver förändringar i de routrar som används i nätverket. Varje router sätter ett värde i ett dedicerat fält i IP-huvudet, värdet bygger på routerns adress och den föregående routerns adress. Eftersom utrymmet för det fält Collins och Reiter planerar använda är begränsat kan det inträffa att två olika vägar över Internet får samma signatur med den här metoden. Sannolikheten för att det ska inträffa beror på storleken på det fält som används och antalet bitar som varje router sätter i fältet.

Dessa fyra metoderna använder data i Cisco:s NetFlow-format som samlas in med hjälp av System for Internet-level Knowledge (SiLK) systemet. SiLK konverterar datamängden till ett utrymmessnålt format som gör det möjligt att samla in data under lång tid. På så sätt är det möjligt att samla in DoS-data i en skarp miljö i och med att det går att vänta tills någon gör en DoS-attack, vilket kan innebära en lång väntan.

2.1.15 Formalizing Sensitivity in Static Analysis for Intrusion Detection

Feng et al [21] använder systemanrop från stacken tillsammans med anropens returadresser för att detektera anomalier. En metod går ut på att det program som ska övervakas förändras och speciell kod förs in så att det går att extrahera programmets körstatus.

Huvudinriktningen på konferensbidraget är olika tillståndsmaskiner för att kunna göra detekteringen. Därför beskrivs inte de parametrar som används i någon större utsträckning. Författarna inriktar sig på att skapa deterministiska modeller, såväl för stacken som för programmets exekvering i stort.

2.1.16 Fast Portscan Detection Using Sequential Hypothesis Testing

Jung med kollegor försöker detektera portskanningar [22] genom att titta på de uppkopplingsförsök som lyckas, misslyckas och blir obesvarade. De data de använder för utvecklingen av algoritmen kommer från loggar producerade av Bro [13], som är ett nätverksbaserad intrångsdetekteringssystem. De innehåller en tidstämpel för när en uppkoppling initierades (både in- och utgående), hur lång tid uppkopplingen varade, den slutgiltiga statusen för uppkopplingen, nätverksprotokoll, mängden data som överförts och IP-adressen för de inblandade parterna.

Algoritmen för att detektera portskanningar bygger på sekvensiell hypotestestning och kallas för *Threshold Random Walk*. I korthet bygger den på att uppkopplingsförsök från en dator följs och den betingade sannolikheten för att någon av två hypoteser (portskanning eller ej) är sanna beräknas kontinuerligt. Ligger sannolikheten inom ett förutbestämt intervall fortsätter “vandringen”, annars meddelas resultat.

2.1.17 Anomalous Payload-Based Network Intrusion Detection

Wang och Stolfo [23] presenterar en metod för att kunna upptäcka anomalier i trafik riktad till en specifik port genom att studera bytefrekvensfördelningen för datadelen i nätverkspaket. De använder 1-gram, det vill säga enskilda byte för att detektera avvikelser, bytefrekvensfördelningen kan därför beskrivas i form av histogram. Resultaten från den utvärdering de gör av metoden varierar stort, men för trafik till port 80 (http) är detektionsgraden nära 100 % och andelen falsklarm omkring 0,1 %.

För att kunna jämföra nya, okända datadelar med en modell skapar de vad de kallar för en *centroid*. Den består av två vektorer med medelvärdet respektive standardavvikelsen för varje bytekod. Nya paket jämförs sedan med centroiden genom att avståndet i till den beräknas med hjälp av en variant av Mahalanonbi-avstånd. I praktiken kan dock avståndsmetriken liknas vid ett Manhattan-avstånd viktat med standardavvikelsen för varje bytevärde i centroiden.

Den formel som används för att beräkna det förenklade Mahalanonbi-avståndet visas i 2.1, där $\vec{x}$ är bytefrekvensfördelningen i vektorform för det som ska kategoriseras och $\vec{y}$ är centroiden i vektorform. Standardavvikelsen för bytefrekvens i betecknas med σ_i och α är en utjämningsfaktor som tar hand om de fall då $\sigma_i = 0$.

$$d\left(\vec{x}, \vec{\bar{y}}\right) = \sum_{i=0}^{n-1} \left(|x_i - \bar{y}_i| / \left(\bar{\sigma}_i + \alpha\right)\right) \tag{2.1}$$

För att åtgärda problemet med att medelvärde och standardavvikelse är beroende av storleken på den datamängd som används för att göra beräkningarna skapar de grupper av centroider genom klustring. Sedan jämför de nya paket med den grupp som ligger närmast enligt ett vanligt Manhattan-avstånd.

De använder även centroiden för att skapa något som de kallar en *z-sträng*. Det är helt enkelt bytekoderna i centroiden sorterade i fallande frekvensordning. Wang och Stolfo föreslår att Z-strängen används som en rudimentär signatur för nya attacker och elakartad kod.

2.1.18 Seurat: A Pointillist Approach to Anomaly Detection

På RAID 2004 presenterade Xie et al [24] en metod för att upptäcka intrång genom att titta på förändringar av filsystemet i en dator. Med hjälp av korrelering av filsystemsförändringarna både spatialt och temporalt kan de även detektera anomalier i nätverk.

De mer exakta parametrarna de använder är inte beskrivna, men de nämner att deras metod också beaktar förändringar i storlek, innehåll och rättigheter som rör filer. Det gör att det med stor säkerhet är MAC-tidsstämplarna som är de primära parametrarna. Förändringar av innehållet kontrolleras via någon slags hashning, de beskriver det som “content digest” [24, sid. 253].

Filer med likadana fullständiga sökvägar men i olika datorer i nätverket betraktas som samma fil. Om sökvägarna skiljer sig åt, men innehållet i filerna är detsamma, betraktas filerna som olika.

2.1.19 Anomaly Detection Using Layered Networks Based on Eigen Co-occurrence Matrix

Oka et al [25] använder sig av egenvärden och egenvektorer för att detektera intrång i nätverk. Deras generella källa för parametrar är sekvenser, i den implementation som presenteras i konferensbidraget från RAID 2004 har de valt att använda sekvenser av användarkommandon i Unix för att illustrera principen. Några exempel på sådana kommandon är *ls, grep, cd* och *emacs*.

Författarna skapar en *co-occurrence* matris som beskriver relationerna mellan olika inom ett visst avstånd från varandra. Ju högre frekvens av kommandopar eller kortare avstånd mellan två kommandon i ett par, desto högre korrelation mellan dem. Matrisen komprimeras sedan genom att dess egenvärden och egenvektorer beräknas och de egenvektorer med de N högsta egenvärdena används för en ny co-occurrence matris.

2.1.20 Efficient Intrusion Detection using Automaton Inlining

Gopalakrishna et al [26] använder de anrop som görs av processer till olika kodbibliotek för att detektera anomalier i ett system. Ett program fångar upp anrop till funktioner i bibliotek och jämför dem med en modell. Den består av en vektor som innehåller aktuella tillstånd och beräknar hela tiden vilka tillstånd som för stunden kan nås från vektorn. Om det inte finns några nya tillstånd att nå innebär det en anomali och ett larm ges.

Författarna påpekar att det är mer kostsamt ur minnes- och exekveringshänseende att använda biblioteksanrop och inte bara de systemanrop de ger upphov till. De

skriver att det enda tillfället det skulle vara mer kostsamt att använda systemanrop vore om varje biblioteksanrop gjorde flera systemanrop. I verkligheten är det dock så att det snarast är normalfallet, eftersom biblioteksanropen används för att förenkla rutinartade programavsnitt för programmeraren.

Anledningen till att de använder biblioteksanrop och inte systemanrop är att det är arbetsamt att analysera den exakta användningen av systemanrop, vilket skulle behöva göras för att hitta de relevanta systemanropen. Skulle de gjort det hade de behövt spåra alla de systemanrop som varje biblioteksanrop ger upphov till, vilket kan bli komplext för dynamiskt länkade bibliotek.

Gopalakrishna et al planerar dock att åtgärda problemet med spårning av systemanrop i framtiden. Dessutom ville de undersöka om det gick att använda biblioteksanrop i stället för systemanrop eftersom de tidigare arbeten inom området de känner till bara använder sig av statiskt länkade program och därmed undviker stora delar av den arbetsintensiva analysen av dynamiskt länkade bibliotek.

2.2 Nätverksbaserade parametrar

Nätverksbaserade parametrar innebär i praktiken att olika delar av nätverkspaket har använts. I Tabell 2.3 har därför en uppdelning på pakethuvuden och paketdatadelen gjorts. Principiellt finns det naturligtvis ett alternativ till, att titta på hela paket. Det alternativet går dock att formulera genom att inkludera båda de kategorier som används här.

Tabell 2.3: Nätverksbaserade parametrar från de konferensbidrag som använts i rapporten

Parameter	Referens
Pakethuvuden	[6, 12, 15, 17, 20]
Paketdatadelen	[23]
Paketflöden	[22]

Tabellen visar att parametrar vanligtvis tas bara från pakethuvudet. Det finns flera skäl till det, bland annat krav på snabb detektion (nära realtid) och förmåga att kunna fungera även i nätverk med stor bandbredd. Likaså innebär standardiseringen av pakethuvuden att det är relativt enkelt att modellera ett normalläge, det finns ett begränsat antal tillåtna utseenden på pakethuvudena. Dessutom utförs vissa typer av attacker genom att pakethuvudet manipuleras. Ett annat skäl till att utesluta datadelen av paketen är att den personliga integriteten eventuellt skyddas genom att inte användardata läses. Det är dock inte helt sant, för att till exempel information om vilka webbplatser användaren besöker och hur ofta och länge kan lätt ställas med hjälp av pakethuvuden.

Att istället använda datadelen av paketen innebär att belastningen på intrångsdetekteringssystemet ökar när den större mängden data ska behandlas. Likaså är inte datadelen standardiserad på samma sätt som pakethuvudet, vilket gör det mer komplicerat att detektera avvikelser. Å andra sidan utförs många attacker genom *buffer overrun*, vilket sker via datadelen av paketen. Därigenom går dylika attacker bara att detektera genom att datadelen analyseras, att använda pakethuvudet är lönlöst.

Detektion av intrång via datadelen av nätverkspaket medför också en risk för att den personliga integriteten hos användarna kan påverkas. Beroende på vilken typ av

algoritm som används innebär det dock ett större eller mindre problem. Algoritmer som på något sätt sätter samman flera paket och tolkar innehållet i dem kan jämföras med att till exempel läsa den information som användaren hämtar på Internet, eller skickar via e-post. Om i stället enskilda datadelar används är det mindre risk att de data som algoritmen tar del av ska kunna innehålla tillräckligt mycket känslig information. Dessutom bygger till exempel [23] på att datadelens struktur och inte dess innehåll som sådant analyseras. Det är dessutom skillnad på om paketen analyseras manuellt eller maskinellt, maskinell analys kan anonymisera data innan de presenteras för användaren, medan vid manuell analys filtreras inte datamängden och operatören får tillgång till rådata.

Ett konferensbidrag tittar på en högre nivå av nätverkstrafik genom att använda hela flöden av paket i stället för enskilda paket. Att använda paketflöden för intrångsdetektering har ett smalare användningsområde än pakethuvuden och paketdatadelar och har ungefär samma för- och nackdelar som pakethuvuden, men kräver mer processorkraft och minnesutrymme att hantera. På en högre nivå kan paketflöden betraktas som ett specialfall av att använda pakethuvuden.

Paketflöden såsom de används av Jung och kollegor [22] innebär egentligen att det är handskakningsprocessen som övervakas. Med andra ord är det ett fåtal paket i början och slutet av en session som används, resten negligeras. Det innebär också att datadelen av paketen inte används, utan det är enbart några fält i pakethuvudet som ingår i algoritmen. Fördelen är att det går snabbt att hantera den nödvändiga informationen, men att beroende på belastning av den eller de servrar som övervakas kan ett stort antal sessioner behöva hanteras samtidigt. Används bara den inledande delen av handskakningen kan intrångsdetekteringsalgoritmen kasta de handskakningar som klarats av i sin helhet. Däremot måste de handskakningar som av någon orsak inte färdigställts hållas i minnet tills de fått status OK eller tillräckligt lång tid förlöpt för att de ska klassas som portskanningar.

2.3 Systembaserade parametrar

Systembaserade parametrar används för systembaserad intrångsdetektering, vilket också framgår av namnet. De innehåller fler och mer disparata undergrupper än de nätverksbaserade parametrarna. Den allra största gruppen av systembaserade parametrar är systemanrop som står för 45 % av alla parametrar som används i de konferensbidrag som studerats i den här rapporten. I Tabell 2.4 visas de klasser av parametrar som jag identifierat.

Tabell 2.4: Systembaserade parametrar från de konferensbidrag som använts i rapporten

Parameter	Referens
Systemanrop	[5, 8, 10, 11, 12, 14, 19, 21, 25]
Biblioteksanrop	[26]
Filer	[11, 24]
Windows-registret	[16]

Underguppen av systemanrop skulle egentligen kunna delas upp i ännu fler grupper eftersom till exempel processtackens användning och returadresser har fått ingå i

systemanropsgruppen. Nu har inte dessa parametrar förekommit ensamma i något av konferensbidragen, utan alltid använts i kombination med systemanrop.

Ett problem med att använda systemanrop är att det inte finns några generellt olämpliga anrop, utan varje anrops legitimitet måste kontrolleras i sin kontext. När till exempel ett ordbehandlingsprogram körs bör det vara tillåtet att skriva till filsystemet, men bara vid vissa tillfällen. Om i stället användaren läser mail är det inte lika självklart att det ska vara tillåtet göra filskrivning. Det gör att det är svårt att använda enskilda systemanrop som parameter för intrångsdetektering. Alla konferensbidrag som använder någon form av systemanrop gör det i form av sekvenser av anrop. I flera fall är det olika tillståndsmaskiner som används, vilka i princip modellerar hela exekveringsförlopp. Den kortaste sekvens som används är 2-gram [10].

Ett alternativ till att använda systemanrop är att i stället använda anrop till de kodbibliotek som är inlänkade i programkoden. Det innebär att det i de flesta fall finns fler anropstyper att titta på, det vill säga upplösningen blir högre. Det kan i vissa extremfall till och med vara så att funktionerna i ett bibliotek inte är kopplade till några systemanrop. Ett exempel som ges i [26] är *string*-funktionerna.

Användningen av biblioteksfunktioner medför tyvärr också i de flesta fall att belastningen på systemet ökar i jämförelse med om systemanrop skulle använts. Gopalakrishna med kollegor påpekar att det inte alltid är en många-till-enkoppling mellan biblioteksanrop och systemanrop. Vid till exempel namnuppslagning i nätverk via *resolve*-anrop görs flera systemanrop för varje biblioteksanrop. I det specifika fallet minskar alltså belastningen på systemet om biblioteksanrop används. Normalfallet är dock inte att det är en många-till-enkoppling mellan biblioteksanrop och systemanrop, eftersom biblioteksanropen används för att dölja komplicerade sekvenser av rutinartade systemanrop för programmeraren.

Ett annat problem som Gopalakrishna med kollegor undviker genom att använda biblioteksanrop är dynamiskt länkade bibliotek som döljer underliggande systemanrop. Genom användningen av dynamiska länkar är det svårt att få förutsägbarhet i exekveringen och därmed försvåras anomalidetektering avsevärt, vad som i ena stunden är en anomali är i nästa stund en normal sekvens av anrop. Därför bör användning av biblioteksanrop i möjligaste mån kombineras med användning av systemanrop. Ett sätt kan vara att ta hjälp av processtacken på det sätt som föreslås i [14].

Två av de studerade konferensbidragen använder sig av parametrar relaterade till filer. I det ena fallet handlar det om att utnyttja den information som finns i körbara filer. Genom att titta i huvudet på PE-filer kontrolleras när, var och hur DLL:er anropas av filen. Om någonting är förändrat i detta mönster innebär det en anomali. Metoden är mycket lik sättet systemanrop används på, i det här fallet tillämpas de idéerna på en högre nivå. Även två andra metoder som går ut på att skapa signaturer av olika körbara filers binära och textuella innehåll presenteras, men de är inte unika för körbara filer utan kan även appliceras på andra typer av filer.

En annan form av filrelaterade parametrar är att titta på förändringar av MAC-tider, hasher av innehåll och rättigheter (skrivning, läsning, m.m.). Det här är parametrar som ofta används vid intrångsanalys är relativt lätta att läsa av. Det finns dock skäl till att inte lita helt på den information som ges om parametrarna eftersom de helt eller delvis handhas av operativsystemet. En förändring av en fil kan döljas genom att parametrar på lägre nivå i operativsystemet manipuleras.

Att använda Windows registret för intrångsdetektering har liknande problem och fördelar som de filrelaterade parametrarna har, om inte annat därför att registret i praktiken består av en serie filer. Dessvärre är den heller inte ordentligt skrivskyddad

i alla delar eftersom den måste kunna hanteras av både kärnan och användare som installerar nya program.

Windows använder registret kontinuerligt, så det är mycket viktigt att välja rätt nycklar att kontrollera. Apap och hans kollegor pekar på att vissa nycklar i normalfallet bara används (skapas eller förändras) vid installation av operativsystemet. Det gör att det är ett enkelt sätt att detektera anomalier på, men att det också är relativt lätt att dölja förändringar för en skicklig angripare.

2.4 Tillämpningsbaserade parametrar

Att använda tillämpningsbaserade parametrar för intrångsdetektering är inte så vanligt. I och med att de befinner sig på en mycket hög nivå i ett datorsystem är de känsliga för angrepp på lägre nivåer. Å andra sidan erbjuder de möjligheter att detektera olika former av missbruk på användarnivå med hög upplösning. I Tabell 2.5 visas de konferensbidrag som använder någon typ av tillämpningsbaserade parametrar.

Tabell 2.5: Tillämpningsbaserade parametrar från de konferensbidrag som använts i rapporten

Parameter	Referens
CORBA-anrop	[9]
E-postserver	[18]

Att använda anrop till CORBA-objekt kan ses som en version av systemanrop på distans och något högre nivå. I och med att all kommunikation mellan klient och server i tillämpningen går via CORBA är det en bra plats att samla data på för intrångsdetektering. Likaså innebär den faktiska separationen av klienter och server att det är svårare att attackera lagren under CORBA, naturligtvis förutsatt att inte några andra tjänster körs på servern.

Den andra tillämpningen som används som parameterbas är en e-postserver. Där är det olika konstelationer av sändning och mottagning av mail som används som parametrar för detektering av maskutbrott. Det här konceptet liknar till viss del paketflödestypen som beskrivs i Kapitel 2.2. Fördelarna med den här parametertypen är att den är enkel att extrahera och förstå, samt inte belastar systemet i onödan. Tyvärr kan den ge upphov till falsklarm, till exempel i sådana fall där massutskick sker oregelbundet. Kombineras parametertypen med en parametertyp från en lägre nivå, eller med någon slags innehållskontroll av mailen minskar risken för falsklarm. Å andra sidan är innehållskontroll en mer exakt parametertyp, som dessutom redan finns implementerad i många av de anti-virusskydd som finns på marknaden.

3 Fördjupning

Konceptet med att försöka kategorisera binära datafragment enbart med hjälp av deras struktur är mycket intressant. Därför behandlar det här kapitlet ytterligare fyra konferensbidrag som rör kategorisering av datafragment med hjälp av deras struktur, förutom det som redan presenterats i Kapitel 2.1.17). Fyra av fem bidrag är således skrivna av forskargruppen kring Salvatore Stolfo. Det finns dock andra grupper som forskar om närliggande områden, men dessa ingår inte i den här fördjupningen eftersom jag velat visa på vad som gjorts specifikt inom datafragmentidentifieringområdet.

I ett bidrag [27] från West Point-konferensen presenterar Stolfos grupp hur de använt de metoder de skrev om i [23] för att i stället skapa vad de kallar *fileprints*. Dessa fileprints används sedan för att bestämma filtypen på filer där den vanliga identifieringen via till exempel filnamnsändelse inte fungerar. Principen för fileprints är i huvudsak densamma som i det första konferensbidraget där en modell av en viss datatyp jämförs med nya, okända datafragment. För att skapa modellen används fortfarande 1-gram, det vill säga enskilda byte. De experimenterar däremot med andra modeller för att framställa centroider på och jämför de nya modellerna med den ursprungliga enkel-centroidmodellen vad gäller hur bra de kan kategorisera okända filer.

Den första modellvarianten innebär att de använder K-means klustring av träningsfilerna. De kluster som blir resultatet av operationen används sedan som för att skapa centroider från. Det betyder att varje filtyp representeras av flera centroider. På så sätt ökar upplösning och precision vid kategorisering av nya filer eftersom de variationer i träningsfilerna som i enkel-centroidfallet gav upphov till hög standardavvikelse för vissa bytefrekvenser nu kan representeras på ett bättre sätt. Avståndsmetriken bygger för den här modellen på författarnas förenklade Mahalanobi-avstånd (se Ekvation 2.1). Den här modellen är bättre än den ursprungliga på att korrekt kategorisera filer.

Nästa nya variant av modell överger centroiderna och använder i stället N stycken slumpmässigt valda typfiler som modell för en filtyp. Genom att göra så slipper de beräkna medelvärde och standardavvikelse för varje bytefrekvens. Eftersom standardavvikelse saknas kan inte Mahalanobi-avstånd användas, utan de använder Manhattanavstånd i stället. Metoden visar sig vara den bästa av de tre på att korrekt kategorisera filer.

De nya modellerna kräver mer beräkningskapacitet vid själva kategoriseringen, men det kompenserar de genom att använda trunkerade data. Deras tester görs med de 20, 200, 500 och 1000 första byten av filerna och det jämförs sedan med kategoriseringsförmågan när hela filen används. Testerna visar genomgående att det är bättre att trunkera filerna.

Det här konferensbidraget [28] presenterades på RAID 2005 och är från Stolfos

grupp. Det behandlar en vidareutveckling av PAYL, som beskrevs i [23]. Förändringen innebär att i stället för en enkel-centroidmodell använder de nu den modell med flera klustrade centroider som beskrivs i [27]. Konferensbidraget är också mer inriktat på hur PAYL kan användas i distribuerade system för att kunna upptäcka attacker på ett mycket tidigt stadium.

Den teknik de använder är att låta PAYL fungera som ett in-gress/e-gress-filter och jämföra bytefrekvensfördelningen på ingående och utgående trafik. Idén är att maskar och annan elakartad kod som sprider sig gör det relativt snart efter att en dator blivit smittad. Genom att titta på om bytefrekvensfördelningen på ett utgående paket är lika med ett som nyss kom in kan de detektera smitta. De skriver att de har tagit hänsyn till *port forwarding*.

Tekniken med kontroll av bytefrekvensfördelningen används också för att skicka meddelanden och signaturer av misstänkta intrång mellan flera instanser av PAYL. Även om systemen är identiska och utför samma uppgifter kommer nätverkstrafiken till och från dem vara unik och därmed också de bytefrekvensfördelningar som PAYL använder för att detektera intrång. Genom att PAYL-instanserna utbyter signaturer i form av bytefrekvensfördelningar kan mängden falsklarm minskas, när två signaturer är lika räknas det som ett riktigt larm.

Wang med kollegor testar tre olika metoder för att göra jämförelsen av inkommande och utgående trafik. Den första metoden, som kallas *String eqaulity (SE)*, bygger på en exakt jämförelse av datadelen av inkommande och utgående paket. Metoden ger få falsklarm, men kan luras genom att en byte i datadelen av ett utgående paket byts ut förändras jämfört med inkommande.

Nästa modell heter *Longest common substring (LCS)* och den mäter längden på de delar av ett paket som är gemensamma mellan ingående och utgående. Om den längsta sådana delen är längre än ett tröskelvärde klassas den som ett möjligt intrång. Fördelen med den här metoden är att den kan hantera även fragmenterade paket, men den är något mer beräkningsintensiv än SE.

Den tredje metoden har namnet *Longest common subsequence (LCSeq)*. Den liknar LCS, men kräver inte kontinuerliga delsträngar, det vill säga delsträngar kan innehålla jokertecken. Den här metoden klarar av även polymorf elakartad kod, men kan även ge upphov till fler falsklarm.

Stolfo och hans grupp har även skrivit en teknisk rapport [29] om tekniken med att använda bytefrekvensfördelningen för datafragment. Rapporten är en fortsättning på [23, 27, 28], som beskriver intrångsdetekteringssystemet PAYL och principen för fileprints. I den här rapporten appliceras dessa principer på filer och målet är att detektera inbäddad elakartad kod. Författarna konstaterar genom empiriska försök att dagens antivirusprogram i många fall inte klarar av att upptäcka elakartad kod som bäddats in i till exempel PDF- och EXE-filer.

För att råda bot på dessa problem föreslår de att deras metod med kontroll av bytefrekvensfördelning hos filer ska användas. De använder tre metoder för att modellera en normalfil, enkel-centroid, klustrade centroider och typfiler.

En ny princip de testar är att använda även 2-gram för att skapa bytefrekvensfördelningar. Principen visar sig dock vara för outvecklad för att fungera tillfredsställande, men de tänker fortsätta utvecklingen av den. Det främsta skälet till att den inte fungerar tror de är att de använde för lite träningsdata när de tog fram normalmodellen. Eftersom metoden ger exponentiellt fler frekvenser att hantera kräver den också exponentiellt mer data, får den inte det blir frekvensfördelningsmodellen instabil.

Genom att addera elakartad kod till början eller slutet på en PDF-fil testar de

detektionsgraden hos sin metod och ett antivirusprogram som används som referens. När de använder 1-gram, det vill säga bytefrekvensfördelningsmodellen visar metoden sig vara bättre än antivirusprogrammet. När de i stället adderar den elakartade koden till mitten på filerna fungerar inte längre deras metod, anledningen uppger de är att bytefrekvensfördelningen för den elakartade koden är för lik bytefrekvensfördelningen för datadelen i PDF-filen.

Ett bidrag som inte kommer från Stolfos grupp är McDaniels och Heydaris *Content Based File Type Detection Algorithm* [30]. De använder 1-gram och föreslår tre olika metoder för att jämföra fingeravtryck för en viss filtyp med en fil. För att hantera stora differenser i bytefrekvensvärden föreslår de att en funktion som tyvärr inte anges i konferensbidraget ska användas. Funktionen ska i alla fall förstärka de lägre värdena och tona ner de större, men fortfarande ge en normalisering av bytefrekvensfördelningen till 1.

Den första metoden är mycket lik det som Stolfos grupp föreslagit med att titta på bytefrekvensfördelningen och vikta differensen mellan en fils bytefrekvensfördelning och en modell med hjälp av standardavvikelsen för varje bytefrekvens. Vad McDaniel och Heydari föreslår är att vikta med hjälp av korrelationen mellan de bytefrekvensvärden som används för att skapa ett filfingeravtryck. Likaså föreslår de metoder för att ta hänsyn till ordning av bytevärdena i filen.

För att skapa en filfingeravtrycksvektor används Ekvation 3.1. Det nya beräknade värdet i filfingeravtrycksvektorn anges med $NFPA$ och det gamla värdet med $OFPA$. PNF betecknar antalet filer som ingick i filfingeravtrycksvektorn innan det nya värdet beräknades och NA är det nya vektorvärdet.

$$NFPA = \frac{(OFPA \times PNF) + NA}{PNF + 1} \tag{3.1}$$

Korrelationsvärdet för varje bytevärde används sedan för att vikta differensen med. För att få en olinjär funktion för korrelationsvärdet, så att stora differenser straffas hårdare än små differenser använder de följande funktion för att beräkna korrelationsfaktorn $F(x)$:

$$F(x) = e^{\left(\frac{-x^2}{2\sigma^2}\right)}$$

Genom experiment kom de fram till att $\sigma = 0,0375$ [30, sid. 4] är ett lämpligt värde.

Nästa metod bygger på att de även tar hänsyn till korrelationen mellan enskilda bytevärden genom att beräkna korrelationen mellan bytepar. Återigen används Ekvation 3.1 för att beräkna filfingeravtrycksvektorn. Fortfarande beräknas också korrelationsvärden för varje bytefrekvens. De båda värdena för varje byte lagras i en 256×256 matris. En okänd fil genomgår sedan samma beräkningsprocedur som filfingeravtrycket och en 256×256 skapas även för filen som ska kategoriseras. Till sist jämförs filens matris med matriserna för de filfingeravtryck som den ska matchas med och den bästa överensstämmelsen blir kategoriseringen.

Den tredje metoden bygger på att den exakta formen på början och slutet av filer används för matchning. En matris med bytefrekvensfördelningen för varje position av de n första byten skapas. För att sedan jämföra en fil med ett filfingeravtryck används följande ekvation, där C_i betecknar korrelationen för byte i för den fil som ska kategoriseras och G_i betecknar korrelationen för motsvarande byteposition:

$$\bar{S} = \frac{C_1G_1 + C_2G_2 + \ldots + C_nG_n}{G_1 + G_2 + \ldots + G_n}$$

Metod nummer tre är den metod som ger bäst resultat med en detektionsgrad på 95,83 %. Den är också i stort sett lika snabb som den snabbaste metoden, som dock har problem med falsklarm. Författarna påpekar att den tredje metoden bör kombineras med andra eftersom inte alla filtyper har en karaktäristisk början eller slut.

4 Diskussion

Någonting som är förvånande, eller kanske snarare förfärande, är hur lite utrymme som ägnas åt att presentera de parametrar som används för intrångsdetektering i de konferensbidrag som jag gått igenom. I flera fall testades olika typer av algoritmer för att utröna om de var bra för detektering eller ej, men de parametrar som algoritmerna matades med användes enbart för att de ingick i en välkänd datamängd, DARPA ’99. Denna datamängd har för övrigt kritiserats för att ha flera brister [31].

I och för sig är det bra att samma datamängd används vid testning för det underlättar jämförelser av algoritmernas prestanda. Å andra sidan talar bara resultaten om hur pass bra algoritmerna detekterar intrång i en specifik datamängd. Att kunna jämföra algoritmer med hjälp av data från ett skarpt system vore bra, men i dagsläget är det svårt eftersom sekretess och skyddet av den personliga integriteten sätter hinder i vägen.

För att kunna dra några riktiga slutsatser från den här undersökningen krävs det att varje separat algoritm och dess parametrar kan utvärderas, helst praktiskt. Det innebär ett omfattande arbete och kommer bara att kunna tala om vilka av de parametrar som används som är bättre än andra för intrångsdetektering. Om vi istället närmar oss problemet från andra hållet och studerar metodiker från intrångsanalysområdet (IT-forensiska undersökningar) kanske det går att hitta bättre parametergrupper som ännu inte används inom intrångsdetekteringsområdet.

FOI har för närvarande ett litet informellt informationsutbyte med SKL och RKP på teknisk nivå, där olika praktiska aspekter av intrångsanalys och IT-forensiska undersökningar diskuteras. Resultatet av dessa utbyten har visat sig mycket givande med tanke på den ringa insatsen och är definitivt någonting som bör få fortgå och utökas.

Ett mycket intressant område som är värt att undersöka vidare är konceptet med att kunna bestämma typen av data i ett binärt fragment, vilket beskrivs i [23]. De nuvarande metoderna bygger till viss del på användning av metadata. Wang och Stolfo utnyttjar enbart bytefrekvensfördelningsdata när de använder metoden på nätverkspaket [23], men i ett senare konferensbidrag [27] tar de hjälp av metadata och kan därför i praktiken inte kategorisera ett datafragment enbart på dess struktur. De fortsätter sedan i ett konferensbidrag [28] från RAID 2005 att ytterligare utveckla PAYL, den här gången mot att kunna fungera i en nätverksmiljö. Deras senaste bidrag [29] behandlar problemet med att detektera elakartad kod som är inbäddad i ofarliga filer. Metoden de föreslår bygger på deras tidigare arbete och innehåller flera av de förbättringar som de föreslagit tidigare. De håller dock kvar vid att trunkera data och är på så sätt fortfarande beroende av filhuvudet för detektering.

5 Slutsats och framtida arbete

Det går inte att dra några riktiga slutsatser om vilka parametrar som är viktiga för en lyckad intrångsdetektering från en så begränsad undersökning som nu gjorts. Det som går att säga är att den klart största delen, 45 %, var systemanrop och därtill relaterade parametrar. Systemanropen åtföljdes dock alltid av någon eller några andra parametrar, såsom processtackens innehåll eller returadresser för subrutiner.

Likaså ingår pakethuvuden i mer eller mindre alla bidrag som behandlar nätverksbaserad intrångsdetektering. Även om de primära parametrarna inte berör pakethuvudet, som i fallet med paketflöden, kommer ändå pakethuvudet att vara med eftersom det är därifrån adresser, portar och flaggor för handskakning kommer.

Den här rapporten omfattar 20 konferensbidrag från två stora konferenser under 2000-talet. Trots det representerar de bara en bråkdel av alla de bidrag som publicerats under den tiden. Bara för att ett bidrag inte accepterats för publicering på dessa konferenser behöver dess innehåll vara dåligt. Därför skulle undersökningen mycket väl kunna utökas till att omfatta fler bidrag, kanske valda så att de representerar de olika forskargrupper som finns inom intrångsdetekteringsområdet.

Likaså omfattar undersökningen inte några analyser av de kommersiella intrångsdetekteringsprodukter som finns på marknaden. För att få en heltäckande bild av de parametrar som för närvarande används för intrångsdetektering skulle det vara önskvärt att även inkludera några av de större produkterna på marknaden. Ett problem är dock den sekretess som råder i den kommersiella världen på grund av konkurrenssituationen. Det gör att en analys av de kommersiella produkterna försvåras, eller i värsta fall omöjliggörs.

Litteraturförteckning

[1] A. Wespi and D. Zamboni, “RAID international symposium on recent advances in intrusion detection,” http://www.raid-symposium.org/, accessed 2005-11-24.

[2] IEEE, “IEEE symposium on security and privacy,” http://www.ieee-security.org/TC/SP-Index.html, accessed 2005-11-24.

[3] K. Burbeck, “Survey of recent work on learning-based anomaly detection,” 2005, not yet published.

[4] M. Karresand, “Survey of eight recent papers on learning based anomaly detection from the IEEE S&P symposium 2000-2004,” 2005, not yet published.

[5] C. Ko, “Logic induction of valid behavior specifications for intrusion detection,” in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, May 2000, pp. 142–153.

[6] A. Valdes and K. Skinner, “Adaptive, model-based monitoring for cyber attack detection,” in *Recent Advances in Intrusion Detection*, ser. LNCS, H. Debar, L. Mé, and F. Wu, Eds., vol. 1907. Springer Verlag, 2000, pp. 80–92.

[7] JWL, “tcpdump/libpcap,” http://www.tcpdump.org/, accessed 2005-12-19.

[8] A. Ghosh, C. Michael, , and M. Schatz, “A real-time intrusion detection system based on learning program behavior,” in *Recent Advances in Intrusion Detection*, ser. LNCS, H. Debar, L. Mé, and F. Wu, Eds., vol. 1907. Springer Verlag, 2000, pp. 93–109.

[9] Z. Marrakchi, L. Mé, B. Vivinis, and B. Morin, “Flexible intrusion detection using variable-length behavior modeling in distributed environment: Application to CORBA objects,” in *Recent Advances in Intrusion Detection*, ser. LNCS, H. Debar, L. Mé, and F. Wu, Eds., vol. 1907. Springer Verlag, 2000, pp. 130–144.

[10] D. Wagner and D. Dean, “Intrusion detection via static analysis,” in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001, pp. 156–168.

[11] M. Schultz, E. Eskin, E. Zadok, and S. Stolfo, “Data mining methods for detection of new malicious executables,” in *SP ’01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, May 2001, pp. 38–49.

[12] W. Lee and D. Xiang, “Information-theoretic measures for anomaly detection,” in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001, pp. 130–143.

[13] V. Paxson, “Bro: a system for detecting network intruders in real-time,” *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, Dec. 1999.

[14] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni, “A fast automaton-based method for detecting anomalous program behaviors,” in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001, pp. 144–155.

[15] P. Williams, K. Anchor, J. Bebo, G. Gunsch, and G. Lamont, “Cdis: Towards a computer immune system for detecting network intrusions,” in *Recent Advances in Intrusion Detection*, ser. LNCS, W. Lee, L. Mé, and A. Wespi, Eds., vol. 2212. Springer Verlag, 2001, pp. 117–133.

[16] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S. Stolfo, “Detecting malicious software by monitoring anomalous windows registry accesses,” in *Recent Advances in Intrusion Detection*, ser. LNCS, A. Wespi, G. Vigna, and L. Deri, Eds., vol. 2516. Springer Verlag, 2002, pp. 36–53.

[17] M. Ramadas, S. Ostermann, and B. Tjaden, “Detecting anomalous network traffic with self-organizing maps,” in *Recent Advances in Intrusion Detection*, ser. LNCS, G. Vigna, E. Jonsson, and C. Kruegel, Eds., vol. 2820. Springer Verlag, 2003, pp. 36–54.

[18] A. Gupta and R. Sekar, “An approach for detecting self-propagating email using anomaly detection,” in *Recent Advances in Intrusion Detection*, ser. LNCS, G. Vigna, E. Jonsson, and C. Kruegel, Eds., vol. 2820. Springer Verlag, 2003, pp. 55–72.

[19] S.-B. Cho and S.-J. Han, “Two sophisticated techniques to improve hmm-based intrusion detection systems,” in *Recent Advances in Intrusion Detection*, ser. LNCS, G. Vigna, E. Jonsson, and C. Kruegel, Eds., vol. 2820. Springer Verlag, 2003, pp. 207–219.

[20] M. Collins and M. Reiter, “An empirical analysis of target-resident DoS filters,” in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, May 2004, pp. 103–114.

[21] H. Feng, J. Giffin, Y. Huang, S. Jha, W. Lee, and B. Miller, “Formalizing sensitivity in static analysis for intrusion detection,” in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, May 2004, pp. 194–208.

[22] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, “Fast portscan detection using sequential hypothesis testing,” in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, May 2004, pp. 211–225.

[23] K. Wang and S. Stolfo, “Anomalous payload-based network intrusion detection,” in *Recent Advances in Intrusion Detection 2004*, ser. LNCS, E. Jonsson el al., Ed., vol. 3224. Springer-Verlag, July 2004, pp. 203–222.

[24] Y. Xie, H.-A. Kim, D. O'Hallaron, M. Reiter, and H. Zhang, "Seurat: A pointillist approach to anomaly detection," in *Recent Advances in Intrusion Detection*, ser. LNCS, E. Jonsson and et al., Eds., vol. 3224. Springer Verlag, 2004, pp. 203–222.

[25] M. Oka, Y. Oyama, H. Abe, and K. Kato, "Anomaly detection using layered networks based on eigen co-occurrence matrix," in *Recent Advances in Intrusion Detection*, ser. LNCS, E. Jonsson and et al., Eds., vol. 3224. Springer Verlag, 2004, pp. 223–237.

[26] R. Gopalakrishna, E. Spafford, and J. Vitek, "Efficient intrusion detection using automaton inlining," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, May 2005, pp. 18–31.

[27] W.-J. Li, K. Wang, S. Stolfo, and B. Herzog, "Fileprints: Identifying file types by n-gram analysis," in *Proceedings from the sixth IEEE Sytems, Man and Cybernetics Information Assurance Workshop*, June 2005, pp. 64–71.

[28] K. Wang, G. Cretu, and S. Stolfo, "Anomalous payload-based worm detection and signature generation," in *Recent Advances in Intrusion Detection*, ser. LNCS. Springer Verlag, 2005.

[29] S. Stolfo, K. Wang, and W.-J. Li, "Fileprint analysis for malware detection," Computer Science Department, Columbia University, New York, NY, USA, Tech. Rep., 2005, review draft.

[30] M. McDaniel and M. Heydari, "Content based file type detection algorithms," in *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*. Washington, DC, USA: IEEE Computer Society, 2003, p. 332.1.

[31] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, Nov. 2000.