FOI
Totalförsvarets forskningsinstitut
Ledningssystem
Box 1165
581 11 Linköping

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

www.foi.se

# Strid i IT-domänen

MARTIN KARRESAND, MATS PERSSON

FOI-R--2192--SE Användarrapport **Ledningssystem**
ISSN 1650-1942 December 2006

Martin Karresand, Mats Persson

# Strid i IT-domänen

| | | |
|---|---|---|
| **Utgivare**<br>Totalförsvarets Forskningsinstitut – FOI<br>Ledningssystem<br>Box 1165<br>581 11 LINKÖPING | **Rapportnummer, ISRN**<br>FOI-R--2192--SE | **Klassificering**<br>Användarrapport |
| | **Månad år**<br>December 2006 | **Projektnummer**<br>E 7091 |
| | **Forskningsområde**<br>Ledning, informationsteknik och sensorer | |
| | **Delområde**<br>Ledning med samband, telekom och IT-system | |
| | **Delområde 2** | |
| **Författare**<br>Martin Karresand, Mats Persson | **Projektledare**<br>Mikael Wedlin | |
| | **Godkänd av**<br>Johan Allgurén<br>IC, Institutionen för Systemutveckling och IT-säkerhet | |
| | **Tekniskt och/eller vetenskapligt ansvarig** | |
| | **Uppdragsgivare/kundbeteckning**<br>FM | |
| **Rapporttitel**<br>Strid i IT-domänen | | |

**Sammanfattning**

Den här rapporten behandlar de resultat som producerats i FoT-projektet Strid i IT-domänen. Den största delen av arbetet har gjorts med inriktning på intrångsanalys. Bland annat har en metod (Oscar) för att identifiera filtyp hos datafragment tagits fram. Möjligheten att använda mer tekniskt orienterade scenarion inom Försvarsmakten har studerats. Det har även gjorts en studie av hur program för radering av hårddiskar hanterar ATA-standarden. Programmen visade sig inte kunna hantera hela standarden, varför de heller inte kan garanteras kunna radera allt innehåll på en hårddisk.

**Nyckelord**

informationssäkerhet, parametrar för intrångsdetektering, ATA-gränssnitt, scenario

**Övriga bibliografiska uppgifter**

Omslagsbild: Martin Karresand, 2005

| | | |
|---|---|---|
| **ISSN**<br>ISSN-1650-1942 | **Antal sidor**<br>98 | **Språk**<br>Svenska |
| **Distribution**<br>Enligt missiv | **Pris**<br>Enligt prislista | |
| | **Sekretess**<br>Öppen | |

| | | |
|---|---|---|
| **Issuing organisation**<br>FOI – Swedish Defence Research Agency<br>Command and Control Systems<br>P.O. Box 1165<br>SE-581 11 LINKÖPING | **Report number, ISRN**<br>FOI-R--2192--SE | **Report type**<br>User report |
| | **Month year**<br>December 2006 | **Project number**<br>E 7091 |
| | **Research area code**<br>$C^4$ISTAR | |
| | **Sub area code**<br>$C^4$I | |
| | **Sub area code 2** | |
| **Author(s)**<br>Martin Karresand, Mats Persson | **Project manager**<br>Mikael Wedlin | |
| | **Approved by**<br>Johan Allgurén<br>Head, Systems Developement and IT Security | |
| | **Scientifically and technically responsible** | |
| | **Sponsoring agency**<br>FM | |
| **Report title**<br>Warfare in the IT domain | | |
| **Abstract**<br>This report presents the results from the project Strid i IT-domänen (Warfare in the IT domain). The main part of the work has been done in the intrusion analysis field. A method called Oscar to identify the file type of data fragments has been developed. The possibility to use more technically oriented scenarios within the Defence Forces has been studied. There has also been a study of how software to erase the contents of hard disks handles the ATA standard. The result shows that the softwares studied did not fully comply to the standard, hence they cannot be guaranteed to be able to erase all information on a hard disk. | | |
| **Keywords**<br>information security, intrusion detection parameters, ATA interface, scenario | | |
| **Further bibliographic information**<br>Cover photo: Martin Karresand, 2005 | | |
| **ISSN**<br>ISSN-1650-1942 | **Pages**<br>98 | **Language**<br>Swedish |
| **Distribution**<br>By sendlist | **Price**<br>According to price list | |
| | **Security classification**<br>Unclassified | |

# Innehåll

# 1 Inledning

Det här är resultatrapporten för FoT-projektet *Strid i IT-domänen (SiITD)*, som har pågått åren 2004–2006. Rapporten kommer att övergripande behandla de resultat och publikationer som framkommit inom projektet. De publikationer som gjorts på internationella konferenser och workshops har bifogats i sin helhet som bilagor sist i rapporten.

Det stora flertalet av de internationella publikationerna har gjorts under det sista året av projektet, eftersom de bygger på den forskning som gjorts under de två föregående åren. En av publikationerna har gjorts i samarbete med Institutionen för datavetenskap vid Linköpings universitet och i en har personal från Institutionen för Telekrigvärdering, FOI deltagit som projektmedlemmar. Övriga publikationer är framtagna internt av IT-säkerhetsgruppen på Institutionen för Systemutveckling och IT-säkerhet, FOI.

Arbetet är indelat i fyra delområden som alla har gemensamma beröringspunkter. Områdena är;

**scenarion** omfattar arbete (två rapporter) där försök till utformning av scenarion med ett mer tekniskt innehåll ur IT-säkerhetssynvinkel gjorts,

**virtuellt RödaKors-märke** för skydd av IT-infrastruktur hos samhällsnyttiga funktioner,

**testmiljö** för att emulera en trådlös miljö för utvärdering av IT-strid och

**intrångsanalys** med tyngdpunkten lagd på att identifiera de parametrar som är viktiga vid detektion och analys av intrång.

Den röda tråd som binder ihop de olika delarna kan ses som stammen på ett träd bestående av konceptet strid i IT-domänen med de fyra områdena som grenar. Det finns även en logisk koppling mellan dem på så sätt att scenariona beskriver olika situationer där intrångsanalys kan appliceras och ett virtuellt RödaKors-märke faller in som en del där militära och civila system blandas, vilket är mer verklighetsnära. Det hela avslutas med praktiska implementationer av de idéer som kan ses som frukterna på trädet.

## 1.1 Bakgrund

Projektet är en fortsättning på FoT-projektet *IT-vapen i en laborativ miljö* vars främsta syfte var att bygga upp en allmän kompetens inom området. Den kompetens som byggts upp har i det här projektet konsoliderats och fördjupats inom relevanta områden, med särskilt fokus på militära ledningssystem och deras speciella egenskaper.

Löptiden för projektet har varit tre år och det har under dessa år förändrats i takt med att forskningen gått framåt och därigenom visat på viktiga och intressanta aspekter att studera närmare. Förändringarna har också gjorts med hänsyn till planerade internationella samarbeten, allt i samråd med kund.

## 1.2 Syfte och omfattning

Syftet med den här rapporten är att redovisa de resultat som erhållits i projektet. Resultaten redovisas separat för respektive forskningsområde (scenarion, virtuellt RödaKors-märke, testmiljö och intrångsanalys). De mest djupgående resultaten finns att hämta inom intrångsanalysområdet och testmiljö.

Syftet med projekt SiITD har varit att svara på de frågeställningar som redovisas i kap. 1.3, samt att ge en ökad förståelse för hur IT-domänen är beskaffad ur ett militärt perspektiv. Projektet är ämnat att täcka hela området med särskild fördjupning inom intrångsanalys, IT-spaning och intrångsdetektering.

## 1.3 Frågeställning

De frågeställningar som legat till grund för arbetet inom projektet har varit:

- Kan man med IT-baserade vapen föra en framgångsrik strid och vilken samverkan behövs i så fall med konventionella vapen?
- Vilka är IT-hoten?
- Vilka metoder kan eller skall användas för IT-spaning och intrångsanalys och vilka krav ställer detta på egen struktur?
- Hur gör man i omvärlden?

Dessa frågor har i besvarats i varierande grad i och med slutförandet av projektet.

## 1.4 Inkluderade publikationer

Följande vetenskapliga publikationer är bifogade till rapporten i form av bilagor namngivna med versaler i enlighet med nedanstående lista. Referenser i den löpande texten är gjorda med de siffror som anges vid respektive publikation i listan.

A [10] David Lindahl och Mikael Wedlin. A CNA Demonstration. *Proceedings of the International Expert Conference on Computer Network Attacks and the Applicability of International Humanitarian Law 2004*, ss 80–84, 2005. Officershögskolan (FHS).

B [8] Martin Karresand och Nahid Shahmehri. Oscar – File Type Identification of Binary Data in Disk Clusters and RAM Pages. *Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006)*, ss 413–424, 2006. Springer Verlag.

C [6] Martin Karresand och Nahid Shahmehri. File Type Identification of Data Fragments by Their Binary Structure. *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop 2006*, ss 140–147, 2006. IEEE.

D [2] Erika Johansson och Mats Persson. Test Bed for Assessment of CNO and EW Against Emulated Wireless Ad Hoc Networks. *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2006*, ss 318–325, 2006. IEEE.

E [7] Martin Karresand och Nahid Shahmehri. Oscar – File Type and Camera Identification Using the Structure of Binary Data Fragments. *Proceedings of the 1st Conference on Advances in Computer Security and Forensics, ACSF*, ss 11–20, 2006. The School of Computing and Mathematical Sciences, John Moores University.

F [1] Claudiu Duma, Martin Karresand, Nahid Shahmehri och Germano Caronni. A Trust-Aware, P2P-Based Overlay for Intrusion Detection. *Proceedings of the 3rd IEEE International Workshop on P2P Data Management, Security and Trust (PDMST 2006)*, ss 692–697, 2006. IEEE Press.

G [9] Martin Karresand och Nahid Shahmehri. Oscar – Using Byte Pairs to Find File Type and Camera Make of Data Fragments. *Proceedings of the 2nd European Conference on Computer Network Defence, in conjunction with the First Workshop on Digital Forensics and Incident Analysis (EC2ND 2006)*, ss 85–94, 2007. Springer Verlag.

# 2 Verksamhet

Det här kapitlet beskriver kortfattat den verksamhet som bedrivits inom tre av de fyra inriktningarna i SiITD. Speciell tyngd har lagts på den mer praktiska verksamheten i form av implementationer och programutveckling, men även mer allmän information om tankar och idéer kring vad som åstadkommits finns med.

## 2.1 Virtuellt RödaKorsmärke

Tanken på ett virtuellt RödaKorsmärke för att skydda IT-system hos de funktioner och inrättningar som idag är fredade för angrepp vid konflikter dök upp efter en konferens där FOI deltog 2004. Arbetet med att utveckla ett dylikt märke har drivits under flera år och bland annat resulterat i en enkel demonstrator. Vidare har kontakt hållits och diskussioner förts med specialister inom folkrätts- och konfliktfrågor.

Ett flertal arbetsdokument har framställts för att utgöra underlag till ett framtida konferensbidrag. I väntan på lämpligt publiceringsforum har arbetet tills vidare lagts på is för att kunna fortsätt om eller vid positiv respons även internationellt.

## 2.2 Testmiljö

Idén till att bygga en testmiljö för trådlösa ad hoc-nätverk dök upp någon gång under 2005. År 2006 ägnades till att försöka förstå vilka begränsningarna var och att sedan räkna ut hur detta skulle kunna realiseras. Sedan tidigare fanns det en lämplig grafisk miljö som passade bra, nämligen Freke-Tavastsystemet. Dessutom hade vi redan tillgång till lämplig hårdvara för att bygga testmiljön, bestående av mer avancerade HP-switchar.

Testmiljön byggdes och programmerades ihop under våren. Det krävdes lite modifieringar av Freke-Tavast så att det kunde kommunicera med resten av systemet och samtidigt visa nätverket med dess datatakter på bildskärmen. Programkod för att styra och taktbegränsa nätkopplingarna fick skrivas i stort sett från grunden. Det fanns dock en del stöd för datataktsbegränsning och hantering av virtuella nät i Linuxkärnan, vilket gjorde det hela mycket enklare att implementera.

När testmiljön fungerade tillräckligt bra för enkla demonstrationer, ansåg vi att prototypen var klar. Resultatet var så pass bra att vi bestämde att systemetkonceptet skulle publiceras i form av ett konferensbidrag.

## 2.3 Intrångsanalys

Den största delen av arbetet i SiITD har samlats under det gemensamma namnet *intrångsanalys*. Verksamheten har framför allt varit inriktad på att från grunden sätta sig in i de problem och möjligheter som intrångsanalys erbjuder ur ett tekniskt perspektiv. Som ett led i detta arbete har bland andra MUST varit delaktiga, men även civila myndigheter såsom SKL, RKP, SÄPO och FRA.

Tanken med verksamheten har varit att hitta parametrar som är viktiga för att påvisa intrång, både pågående och redan genomförda och på kända eller ännu okända sätt. Dessa kan sedan bland annat användas som bas för att skapa bättre intrångsdetekteringssystem och säkerhetstillämpningar, optimera säkerhetsnivån i system, få en korrekt lägesbild vid intrång eller andra IT-relaterade incidenter, förbättra de grundläggande säkerhetsfunktionerna i framtida ledningssystem och öka kompetensen vad gäller CNO.

För att få en bättre förståelse för verkliga problem och möjligheter inom intrångsanalysområdet har arbetet präglats av tillämpad forskning med en hög grad av programutveckling för att kunna verifiera den teoretiska delen av arbetet. Genom publicering av delresultat på konferenser har vi fått en stor mängd återkoppling från internationella auktoriteter inom området och på så sätt bra förslag på fortsatt väg framåt. Konferensdeltagandet har också bidragit till att höja kvalitén på forskningen och gett oss en god insyn hur vi ligger till på den internationella akademiska forskningsfronten.

Likaså har mycket inspiration hämtats från kontakter med den operativa delen inom IT-brottsbekämpning, både hos polisen och SKL, men även från andra myndigheter. Dessa kontakter har främst bestått av studiebesök och tekniskt orienterade möten där vi fått inblick i de problem som kan uppstå i samband med intrångsanalys och utredning av IT-relaterade brott.

# 3 Sammanfattning av publikationer

Det här kapitlet sammanfattar de publikationer som producerats inom SiITD, både till internationella konferenser och workshops, men även FOI-rapporter och memon. De publikationer som presenterats på den internationella vetenskapliga arenan återfinns i sin helhet som bilagor sist i rapporten.

## 3.1 Scenarion

Inom spåret för scenarioutveckling har skrivits två FOI-rapporter [5, 14] där den första utgör underlag för den andra rapporten.

Rapporten *Scenarion och trender inom framtida informationskrigföring ur ett tekniskt perspektiv* [5] tar upp tre scenarion (två färdiga och ett förslag) som alla behandlar konflikter med inslag av IT-strid (Computer Network Operations (CNO)). Dessa scenarion har först sammanfattats utifrån ett IT-tekniskt perspektiv, för att sedan analyseras. Resultatet av analysen är att precisionen i scenariona är för dålig vad gäller IT-stridsbeskrivningar, de sägs utspela sig 10 till 15 år framåt i tiden (2015–2020), men är mycket övergripande beskrivningar av attacker mot dagens system. De anmärkningar som riktas mot de nuvarande scenariona kan sammanfattas i följande punkter:

- Det görs implicita antaganden om att de egna datorsystemen har så bra skydd att endast insider-hot är relevanta,
- det talas aldrig om huruvida de resurser i form av personal och hårdvara som krävs för att upprätthålla den implicit antagna säkerhetsnivån verkligen finns tillgängliga hela tiden,
- det talas inte heller om det faktum att IT-striden är asymmetrisk, det vill säga den med mest utbredd teknikalisering också är den mest sårbara,
- slutligen är inte eventuella IT-angrepp mot det eventuellt mer sårbara civila samhället medtagna i scenariona.

Rapporten avslutas med att konstaterande om att det fortsatta arbetet med att utveckla bra scenarior för IT-angrepp kommer att inriktas mot att utveckla metoder eller teorier för att koppla samman övergripande scenarior och teknik.

Rapport nummer två [14] i serien om scenarior behandlar två förslag på egna, tekniskt detaljerade scenarior som är tänkta att kunna inlemmas i de stora övergripande scenariorna. De tekniska scenariorna behandlar alla dagens teknik på grund av den mycket stora osäkerhet om den tekniska utvecklingen om 10–15 år. Tyngdpunkten har lagts på en ur IT-säkerhetssynpunkt korrekt beskrivning av möjliga attackvägar. Dessa har sedan klätts med relevanta kringhändelser som tillsammans bygger upp ett fungerande scenario. Kringhändelserna är dock enbart baserade på rapportförfattarens egen fantasi och generella kunskap om världspolitik och kan mycket väl behöva korrigeras innan de sätts samman med de stora scenariorna.

Det första scenariot baseras på upptäckten av en trojan i en dator och ger sedan tre olika alternativa spår med stegrande inblandning av offensiv verksamhet. Scenario nummer två behandlar injicering av trafik i ett koalitionsnät via manipulerade trådlösa noder. Liksom i scenarion nummer ett beskrivs ett antal varianter av händelseförloppet med stegrande grad av offensiv verksamhet.

I rapportens slutsatser tas ett par intressanta frågeställningar gällande det fortsatta arbetet upp. I korthet går de ut på att

- det finns svåröverbryggbara skillnader i koncepts- och föreställningsvärld mellan tekniska scenarior och övergripande typscenarior,
- skärningsytorna mellan tekniska scenarior och övergripande typscenarior misstänks vara små,
- tekniska scenarior kanske inte fyller sitt syfte, och
- det är därför kanske bättre med tekniska experiment på riktiga system i stället.

## 3.2 Virtuellt RödaKors-märke

Idén om ett virtuellt RödaKors-märke för civila funktioner såsom sjukhus och andra vårdinrättningar dök upp år 2004 efter en demonstration av nätverksattacker (Computer Network Attack (CNA)) vid en konferens om folkrätt och IT-säkerhet. FOI:s CNA-demonstation berörs i ett kapitel [10] i dokumentationen från konferensen.

I bakgrunden till demonstrationen pekar författarna på skillnader mellan konventionell krigföring och IT-krigföring, skillnader som innebär att IT-vapen är mer autonoma och därmed svårstyrda. Likaså kan de ej fatta egna beslut om någonting i förutsättningarna för attacken har förändrats, ett exempel är att en soldat som fått order om att slå ut en byggnad själv kan avgöra att någonting är fel och därmed undvika att icke-kombattanter dödas. Ett IT-vapen däremot slår blint och följer slaviskt de order det fått.

Ett annat problem är motsättningen mellan användbarhet och säkerhet. Meningen med att införa datorer är att underlätta informationshantering. Om säkerhetsnivån är för hög i relation till användbarheten faller nyttan med datorsystemet. Det ger i slutänden i stället en för låg säkerhetsnivå.

Ytterligare en orsak till dålig IT-säkerhet som anges är att det kostar pengar att skydda sig mot någonting som kanske inte ens händer. Dessutom är trenden att använda allt mindre specialanpassad hårdvara och i stället utnyttja vad som finns tillgängligt på den öppna marknaden. Det gör att system som egentligen har höga krav på säkerhet kanske inte ens har möjlighet att ge den säkerhetsnivå som krävs.

Det påpekas i konferensbidraget att en viktig orsak till de problem som finns är den brist på kunskap som genomsyrar hela IT-etablissemanget, från programmerare och utvecklare, via försäljare till slutkonsument. Det första ledet känner inte till behoven hos slutanvändarna, säljarna kan inte tillräckligt mycket om vare sig datorsystemen eller kundernas behov, och slutkonsumenterna vare sig bryr sig eller vill veta vad som pågår under huven på datorn.

Några av de nyckelområden som orsakar fel och intrångsmöjligheter i en dator är enligt författarna

- dold interaktion mellan programvaror och dolda funktioner i desamma,
- dålig kvalitetskontroll av programvaror med säkerhetshål som följd,
- okunskap om vikten av uppdaterad säkerhetsprogramvara och
- modifierad hård- och mjukvara.

Konferensbidraget sammanfattas med gissningen att på grund av ovan angivna orsaker kommer möjligheterna till CNA inte att minska i framtiden, utan snarare tvärt om.

## 3.3 Testmiljö

Testmiljön för IT-strid mot trådlösa ad hoc nätverk presenterades i [2]. Författarna skriver att deras testmiljö har flera viktiga fördelar, bland annat så är den oberoende av de fysiska nodernas konfiguration och kan användas för att utvärdera störningsutrustning och skydd utan risk för röjande signaler.

Enligt författarna är den främsta fördelen med deras verktyg att det kombinerar verkliga sändningsförhållanden (via geografisk information) med användning av fysiska noder utan att behöva använda radiogränssnittet. Dessutom kan virtuell positionering göras grafiskt med hjälp av de kartor som systemet använder. Allt detta ger en mer verklighetsnära test- och utvärderingsmiljö än liknande verktyg på marknaden.

Emulatorn består av fyra delar,

1. en ad hoc-nätverksemulator,
2. en datornätverksemulator,
3. en virtuell LAN-switch och
4. ett antal noder som utgör det testade nätverket.

Det går att flytta noder på kartan och på så sätt få ändrade förhållanden. Likaså är två olika antenner implementerade för att ge mer realistiska förhållanden. Det går att addera ytterligare antennmoduler för att bygga mer komplexa miljöer. De flesta parametrar som styr emuleringen går att ändra dynamiskt.

Ett enkelt test har utförts med gott resultat, men författarna påpekar att de har flera utökningar de vill göra för att ytterligare förbättra emuleringen. Förbättringarna omfattar bland annat inköp av bättre hårdvara och mer detaljerad styrning av transmissionsparametrarna.

## 3.4 Intrångsanalys

Den största delen av arbetet i projektet har lagts på intrångsanalysdelen, vilket också varit den mest forskningsnära delen. Det har medfört att 5 av 7 av de vetenskapliga artiklar som skrivits återfinns inom det här området. Likaså återfinns 4 av 6 av alla FOI-rapporter i projektet inom intrångsanalysdelen.

### 3.4.1 Rapporter

Arbetet med intrångsanalys började med en introduktion [3] till området med idéer om intressanta frågeställningar för vidare fördjupning. Fokus var militära nätverk, men i och med att en ökande andel av hård- och mjukvaran som används inom den militära grenen är standardprodukter, så kallade Commercial Of The Shelf (COTS), kom resultatet att kunna gälla lika väl för civila nätverk.

En genomgång av nuvarande forskning inom området görs för att ge en utgångspunkt för läsaren. De områden som inkluderats i genomgången är

- praktisk kriminalteknik vid IT-brott,
- spårning av intrång,
- formella metoder för IT-brottsutredning,
- tekniker för att dölja intrång,
- automatisk generering av intrångssignaturer,
- livslängd hos icke-permanent data,
- loggning och
- incidenthantering.

Sedan presenteras den praktiska undersökning av hur länge några av de vanligaste operativsystemen klarar sig utan skydd när de är kopplade mot internet. Det här experimentet gjordes med hjälp av den honungsfälla (honeypot) som implementerats i IT-krigslabbet på FOI. Vid undersökningen framkom det att en stor mängd av attackerna var riktade mot SSH-servrar, så en sådan sattes upp med ett lättgissat lösenord. Efter tre dagar blev den skannad och ett automatiskt intrång skedde, ytterligare några timmar senare gjordes två omgångar av manuella intrång.

Tyvärr var datamängden som inhämtades vid försöken med honungsfällan för liten för att några långtgående slutsatser om strukturen hos intrång skulle kunna dras.

I delen om framtida arbete påpekas vikten av god kännedom om de olika verktyg som används, samt den täta kopplingen mellan hackerverktyg och verktyg för intrångsanalys. De förslag för fortsatt verksamhet som lämnas omfattar därför en vidare studie av dessa verktyg, forskning gällande vilka parametrar, eller spår, som ett intrång lämnar, fördjupning inom teknik för att extrahera data och hur länge dessa data finns kvar på lagringsmediat. Två områden som enligt författarna inte får glömmas bort är de olika säkerhetskrav som system kräver och som på så sätt påverkar möjligheterna till intrångsanalys, liksom den icke-tekniska sidan med användarmanipulering.

Nästa rapport [12] om intrångsanalys behandlar filsystem och deras uppbyggnad, samt en kortare del om olika loggningsystem och metoder för att kontrollera riktigheten hos system och deras informationsinnehåll. Delen om filsystem behandlar *NTFS, FAT32, ext2* och *ext3* med avsikten att texten ska kunna fungera som manual för sådant arbete.

Loggningsdelen presenterar de problem som loggning medför, det vill säga mycket information är bra, men det kräver både plats och kraft av datorn. Dessutom går det inte att logga till exempel alla systemanrop, för det innebär att även loggningsanropen loggas och systemet börjar loopa. Rapportens loggningsdel redovisar också några vanliga loggningsfunktioner hos olika operativsystem.

Några olika programvaror för kontroll av riktighet hos information, det vill säga filer och filsystem presenteras kortfattat. Därefter följer ett stort antal obesvarade frågeställningar som framtida arbete. Rapporten avslutas med konstaterandet att om vi inte förstår grunden för hur ett datorsystem fungerar kan vi aldrig göra det säkert, samt att vi måste försöka hålla jämna steg med hackervärlden och ett sätt att göra det på är att studera spår efter ett intrång i en dator.

Den tredje rapporten [11] i intrångsanalysserien tar upp ett par av de frågeställningar som presenterades i föregående rapport om loggning [12]. Frågeställningarna rör diskkontrollern och då speciellt ATA-gränssnittet. Rapporten rönte stor nationell och internationell uppmärksamhet när den publicerades.

Rapporten innehåller en teoridel där olika aspekter av ATA-standarden och dess funktioner beskrivs. Därefter följer en kort utvärdering av några kända verktyg för radering av hårddiskar och ett hårdvarubaserat skrivskydd för hårddiskar. Varje teorikapitel avslutas med en del innehållande råd till databrottsutredare. Där beskrivs vad som kan gå fel och vilka åtgärder som bör vidtas för att minska risken för fel. De olika funktioner och problem som de kan innebära för en utredare behandlar följande områden

- förändring av rapporterad diskstorlek,
- användning av funktionen för Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T),
- skadehanteringsfunktionen,
- andra aspekter av ATA-standarden (främst lösenordsskydd av hårddiskar),
- ATA-baserade verktyg som kan vara till användning,
- HPA och DCO-test av hårddiskraderingsprogram och
- tester av ett hårdvarubaserat skrivskydd för hårddiskar.

I delen som behandlar radering av hårddiskar konstateras att inte något av verktygen klarade av alla de funktioner som ATA-standarden specificerar. Det innebär i praktiken att det finns en uppenbar risk att hårddiskar raderade med dessa verktyg fortfarande innehåller information efter raderingen. Med andra ord ett potentiellt säkerhetsproblem som måste beaktas.

Den avslutande delen av rapporten tar upp några av de frågor som lämnats obesvarade av rapporten, samt de nya som tillkommit under utredningen.

Den fjärde rapporten [4] om intrångsanalys tar upp frågan om vilka parametrar som är viktiga för intrångsanalys. Rapporten är menad som en introduktion till problemet och presenterar därför de parametrar som används i ett urval av konferensbidrag om anomalidetektering som publicerats vid Recent Advances in Intrusion Detection (RAID) under åren 2000–2004, samt IEEE Symposium on Security and Privacy (S&P) åren 2000–2005. RAID bidrar med 11 artiklar och S&P med 9. Dessutom ingår en fördjupning kring fyra artiklar om filtypskategorisering med hjälp av bytefrekvensfördelning.

Resultatet av analysen av de 20 artiklarna från RAID och (S&P) visar på att de allra flesta algoritmer för anomalidetektering analyserar hela eller delar av pakethuvudet vid intrångsdetektering i nätverk. Vid systembaserad intrångsdetektering används i de flesta fall systemanrop. Båda dessa resultat var helt i linje med vad som förväntades. Analysen försvårades dock av att det i många fall inte anges klart vilka parametrar som används i respektive algoritm.

Fördjupningen i rapporten görs inom vad som författaren anger vara ett intressant område och som dessutom inte har utforskats i någon högre grad. Området berör möjligheten att använda den information som kan extraheras direkt ur en dataström för att avgöra till exempel filtyp hos det som skickas i strömmen. Beroende på vad det är för filtyp kommer strukturen på binärdatat att se olika ut. Om vi bara får tag på ett fragment av binär information, till exempel en bit av en fil eller ett enda nätverkspaket, så är frågan hur mycket information som det går att få ut ur det vi hittat, utan att ha tillgång till någon slags meta-information. Några exempel på tillfällen när det här är extra intressant är vid nätverksspaning, databrottsutredning och återskapande av förlorad information.

De fyra artiklar som fördjupningen baseras på använder alla på ett eller annat sätt bytefrekvensfördelningen i ett datafragment för att avgöra vilken filtyp det tillhör. Det finns dock luckor i deras resonemang enligt rapportförfattaren, vilket har föranlett en fördjupning även inom SiITD-projektet.

Rapporten avslutas med att konstatera att 45 % av de parametrar som användes var systemanrop och därtill relaterade subparametrar, som till exempel stackinnehåll och returadresser för anrop. Även pakethuvudinformation kan ses som en meta-klass för de parametrar som används vid nätverksbaserad intrångsdetektering.

### 3.4.2 Vetenskapliga artiklar

Utifrån de idéer och uppslag till forskning som blev resultatet av rapporterna om intrångsanalys skapades flera vetenskapliga artiklar inom två spår. Det ena behandlade ett förslag till ett intrångsdetekteringssystem för en heterogen miljö med peer-to-peer (P2P) kopplingar mellan klienterna. Det kan liknas vid vad många hemanvändare upplever i dagsläget, men även den dynamiska miljö som det nya nätverksbaserade försvaret kommer att erbjuda. Oavsett vilket som är fallet befrämjar ett sådant system säkerheten i samhället och därmed totalförsvaret. Det här spåret kan ses som avslutat i och med publiceringen av artikeln. Ett andra spåret pågår fortfarande och har hittills resulterat i fyra vetenskapliga publikationer. Forskingen centreras runt den metod, Oscar, för identifiering av filtyp med hjälp av strukturen hos datafragment som blivit resultatet av arbetet med intrångsanalys.

P2P-artikeln [1] behandlar ett intrångsdetekteringssystem som utnyttjar redan befintlig infrastruktur och programvara genom att placera en demon mellan säkerhetsprogrammet och dess loggningsfunktion. Larm skickas sedan som meddelanden i ett P2P-nät om någon nod i nätet detekterar ett angrepp mot sig själv. De andra noderna i nätet kan du välja att lyssna på varningen eller inte. Valet görs utifrån den tillit som respektive nod har för de andra noderna i nätet. Genom att bara kunna varna för attacker (eventuellt övertagande) mot sig själv och därigenom inte kunna ange andra noder, har systemet eliminerat en klass av DoS-attacker. Systemet kräver inte någon särskild programvara, utan endast installation av en hjälpprogramvara.

Systemet testades genom att 36 noder sårbara för Slapper-masken utsattes för en maskattack. Mängden sårbara noder varierades och antalet noder som överlevde attacken, det vill säga blev varnade innan de själva blev smittade, mättes. Det visade sig att överlevnadsgraden vid 1 skyddad nod av 36 ökade med 34 % med det föreslagna systemet.

Uppslaget till spåret om filtypsidentifiering kom från en presentation [13] av Salvatore Stolfo vid RAID 2004. Där visade de på att olika filtyper har olika bytefrekvensfördelning. Forskningen som följde gav upphov till den första artikeln [8] om Oscar, en metod att utan hjälp av meta-data kunna utnyttja information dold i binära filfragment. Det konstaterades också att det inte gick att hitta några datorprogram som inte använde meta-information i form av filsystem eller filhuvuden för att identifiera filtyp hos okända datablock.

Artikeln använder bytefrekvensfördelningens medelvärde och standardavvikelse för att avgöra filtypen hos ett okänt fragment. Metoden mäter avståndet mellan en modell, centroid, av en viss filtyp och fragmentet som ska filtypsbestämmas med hjälp av en kvadratsumma. Detta för att kunna göra skillnad på ett fragment som avviker med ett fåtal stora bytefrekvensvärden och ett som avviker med flera små värden. Storleken på fragmenten är satt till 4 KiB eftersom det är en vanlig storlek på hårddiskkluster och fönster i RAM-minnet. Naturligtvis går det att använda andra storlekar också, den rekommenderade minimistorleken är 512 byte.

En modell för en viss filtyp skapas genom att bytevärdena räknas i sektioner om 4 KiB för en stor mängd data av den specifika filtypen. Medelvärdet och standardavvikelsen för varje bytevärde beräknas sedan med hjälp av resultaten från 4 KiB sektionerna. Genom att använda enskilda byte och inte hela sekvenser hålls minnes- och beräkningsbehovet nere och metoden blir snabb, samtidigt som den kan skilja flera filtyper åt med god precision.

För att visa metodens praktiska betydelse gjordes ett experiment där fragment av en .jpg-bild letades fram ur en klon av RAM-minnet i en dator. Sedan matchades varje fragment mot en databas med kända bilder för att påvisa att bilden förekommit på datorn. Samma experiment utfördes även med en kopia av swap-utrymmet i samma dator. Båda experimenten var framgångsrika.

Nästa artikel [6] om Oscar utvecklar idéen om bytefrekvensfördelningen och lägger till mätning av förändringsgraden hos byteströmmen. På så sätt tas även hänsyn till ordningen på bytevärdena i strömmen. Det är absolutbeloppet av skillnaden mellan två på varandra följande bytevärden som mäts och metoden ökar detektionsgraden med cirka 5 % jämfört med bytefrekvensfördelningen, utan att graden av falsklarm ökar. Om de två metoderna kombineras ökar detektionsgraden med cirka 10 % i stället, fortfarande utan ökning av falsklarmen.

För att kunna se inverkan från kvadratsumman vid avståndsmätningen gjordes även experiment med att använda en vanlig 1-norm, det vill säga summering av värdena. Skillnaden blev ett par procents detektionsgrad till kvadratsummans fördel, 1-normen är dock betydligt enklare att beräkna och om det finns krav på realtidsnära exekvering kan den kanske vara ett alternativ.

Den tredje vetenskapliga artikeln [7] i Oscar-serien utvärderar användning av 2-gram, det vill säga bytepar, för filtypsidentifiering. Användningen av 2-gram medför flera fördelar. För det första inkluderas både bytefrekvens och byteordning automatiskt i jämförelsen. För det andra kommer metoden att innebära att de bytesekvenser som antingen krävs eller som inte får förekomma i en viss filtyp kontrolleras per automatik. Centroiden kommer med andra ord att innehålla en enkel version av formatstandarden för en viss filtyp.

Det finns dock också några nackdelar med att använda 2-gram. I och med att det finns 65536 olika unika 2-gram och Oscar-metoden använder en fragmentstorlek på 4 KiB ger en teoretisk likformig fördelning av byteparen ett medelvärde mindre än 1 och metoden blir instabil, eftersom varje förekomst av ett speciellt bytepar motsvarar flera träffar i full skala. Dessutom innebär den ökade mängden enheter att räkna att exekveringstiden förlängs, liksom behovet av utrymme i RAM.

Trots eventuell instabilitet gav testerna med 2-gram en detektionsgrad på 100 % med 0,1 % falsklarm för .jpg-filer. Resultaten inspirerade till ett extra experiment för att se om det gick att se vilket kameramärke som en bild från en digitalkamera är tagen med. Experimentets resultat tydde på att det faktiskt fanns små skillnader mellan kameramärken.

I det fjärde konferensbidraget [9] gjordes testerna med kameraigenkänning om med en delvis förändrad implementation av Oscar-metoden. Förändringen gjordes för att öka robustheten och minska risken för instabilitet på grund av fragmentstorlek relativt antal möjliga 2-gram. Tyvärr visade det sig att skillnaderna som upptäckts vid det första experimentet förmodligen delvis berodde på just instabilitetfenomenet och därför inte gick att upprepa. Endast Fuji- och Olympuskameror går att skilja från de övriga kameratillverkare som användes i testet, men det beror på att Fuji och Olympus använder återstartsmarkörer i dataströmmen för ökad feltolerans.

# 4 Framtida arbete

Eftersom inriktningen mot intrångsanalys har varit framgångsrik både vad gäller uppnådda resultat och internationell uppmärksamhet ämnar vi fortsätta arbetet även i nästa FoT-projekt, som har namnet *Försvar av IT-system*. Trots det defensivt klingande namnet kommer studierna återigen göras ur en offensiv synvinkel, eftersom det är genom att känna vår motståndare vi kan försvara oss mot denne.

Det preliminära upplägget av det kommande arbetet har följande utseende:

1. Fortsätt och avsluta forskningen kring hur mycket och vilken information som går att få ut ur datainnehåll, utan tillgång till meta-information. Det inkluderar att se om det eventuellt går att foga samman identifierade fragment till en helhet igen.

2. Genomför forskning kring nätverkstrafik och dess informationsinnehåll enligt samma mönster och upplägg som för datainnehåll. Här undantas av naturliga skäl studier av datadelens innehåll.

3. Genomför forskning kring processinformation och stackinnehåll och deras informationsinnehåll enligt samma mönster och upplägg som för datainnehåll. Eventuellt görs en avgränsning av vilka operativsystem som ingår i forskningen, men det bör avgöras i ett senare skede när denna forskning väl kommit igång.

4. Beroende på hur fort forskningen fortskrider kan återgång göras till något av stegen för att fylla i luckor eller testa nya forskningsrön. Detta görs dock först efter en nogrann återkoppling och genomgång av erhållna resultat för att garantera att kraft och resurser sätts in på rätt ställe.

5. Fortsatt utveckling av testmiljön för emulerade nätverk för att utöka funktionaliteten och förbättra stabiliteten.

# Litteraturförteckning

[1] Claudiu Duma, Martin Karresand, Nahid Shahmehri och Germano Caronni. A trust-aware, P2P-based overlay for intrusion detection. I: *Proceedings of the 3rd IEEE International Workshop on P2P Data Management, Security and Trust (PDMST 2006)*, ss 692–697, Krakow, Polen, 2006. IEEE Press. `http://www.ida.liu.se/~cladu/pub/pdmst06.pdf`, accessed 2006-11-13.

[2] Erika Johansson och Mats Persson. Test bed for assessment of CNO and EW against emulated wireless ad hoc networks. I: *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2006*, ss 318–325, Piscataway, NJ, USA, 2006. IEEE.

[3] Martin Karresand. Intrusion analysis in military networks – an introduction. Teknisk rapport FOI-R–1463–SE, Ledningssystem, Systemutveckling och IT-säkerhet, Totalförsvarets Forskningsinstitut, december 2004. `http://www2.foi.se/rapp/foir1463.pdf`.

[4] Martin Karresand. Parametrar för intrångsanalys. Teknisk rapport FOI-R–1831–SE, Ledningssystem, Systemutveckling och IT-säkerhet, Totalförsvarets Forskningsinstitut, december 2005. `http://www2.foi.se/rapp/foir1831.pdf`.

[5] Martin Karresand, Mats Persson och David Lindahl. Scenarion och trender inom framtida informationskrigföring ur ett tekniskt perspektiv. Teknisk rapport FOI-R–1283–SE, Ledningssystem, Systemutveckling och IT-säkerhet, Totalförsvarets Forskningsinstitut, juni 2004. `http://www2.foi.se/rapp/foir1283.pdf`.

[6] Martin Karresand och Nahid Shahmehri. File type identification of data fragments by their binary structure. I: *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop 2006*, ss 140–147, Piscataway, NJ, USA, 2006. IEEE.

[7] Martin Karresand och Nahid Shahmehri. Oscar – file type and camera identification using the structure of binary data fragments. I: John Haggerty och Madjid Merabti, redaktörer, *Proceedings of the 1st Conference on Advances in Computer Security and Forensics, ACSF*, ss 11–20, Liverpool, UK, juli 2006. The School of Computing and Mathematical Sciences, John Moores University.

[8] Martin Karresand och Nahid Shahmehri. Oscar – file type identification of binary data in disk clusters and RAM pages. I: *Proceedings of IFIP*

*International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006)*, Lecture Notes in Computer Science, ss 413–424. Springer Verlag, 2006.

[9] Martin Karresand och Nahid Shahmehri. Oscar – using byte pairs to find file type and camera make of data fragments. I: Andrew Blyth och Iain Sutherland, redaktörer, *Proceedings of the 2nd European Conference on Computer Network Defence, in conjunction with the First Workshop on Digital Forensics and Incident Analysis (EC2ND 2006)*, ss 85–94. Springer Verlag, 2007.

[10] David Lindahl och Mikael Wedlin. A CNA demonstration. I: Karin Byström, redaktör, *Proceedings of the International Expert Conference on Computer Network Attacks and the Applicability of International Humanitarian Law 2004*, ss 80–84, Stockholm, Sverige, 2005. Officershögskolan (FHS).

[11] Arne Vidström. Computer forensics and the ATA interface. Teknisk rapport FOI-R–1638–SE, Ledningssystem, Systemutveckling och IT-säkerhet, Totalförsvarets Forskningsinstitut, februari 2005. `http://www2.foi.se/rapp/foir1638.pdf`.

[12] Arne Vidström, Mats Persson och Martin Karresand. Intrusion analysis in military networks – file systems and logging. Teknisk rapport FOI-R–1518–SE, Ledningssystem, Systemutveckling och IT-säkerhet, Totalförsvarets Forskningsinstitut, december 2004. `http://www2.foi.se/rapp/foir1518.pdf`.

[13] Ke Wang och Salvatore J Stolfo. Anomalous payload-based network intrusion detection. I: E. Jonsson el al., redaktör, *Recent Advances in Intrusion Detection (RAID) 2004*, band 3224 av *Lecture Notes in Computer Science*, ss 203–222. Springer-Verlag, juli 2004.

[14] Mikael Wedlin. CNA-scenarier ur ett tekniskt perspektiv. Teknisk rapport FOI-R–1620–SE, Ledningssystem, Systemutveckling och IT-säkerhet, Totalförsvarets Forskningsinstitut, mars 2005. `http://www2.foi.se/rapp/foir1620.pdf`.

# A Förevisning av CNA – IECCNIHL 2004

David Lindahl och Mikael Wedlin. A CNA Demonstration. *Proceedings of the International Expert Conference on Computer Network Attacks and the Applicability of International Humanitarian Law 2004*, ss 80–84, 2005. Officershögskolan (FHS).

# 6 A CNA Demonstration

David Lindahl and Mikael Wedlin*

*The following text provides comments on the demonstration on CNA, which was conducted by Mr. David Lindahl and Mr. Mikael Wedlin on the 17 November 2004.*

## 6.1 Background

The use of CNA is a new addition to the methods of war. Its effects, however, are not. The effect deriving from a successful CNA is directly related to

– What the attacked system is used for,
– What computers used to propagate the attack are used for.

If the computer system is used to physically control a device, such as a weapon, control of the weapon can be lost, and physical damage can be brought on the targets the weapon can engage. This does not significantly differ from the use of weapons captured in other manners, such as by commandos. If the computer in question is used to store or disseminate information, the information can be altered; causing decisions to be made on false grounds. The information can also be accessed. This does not significantly differ from the goal armed forces attempts to achieve by the use of camouflage, encryption or deliberate dissemination of false rumours or the normal gathering of intelligence.

But if these results are the same threats as always, the methods differ. By utilizing CNA access to information can be automated, thousands of computers infected by a virus in fractions of a second resulting in enormous amounts of information revealed or destroyed. Also, a mistake is much easier to make. A commando raiding a coast line and finding a hospital where his briefing has told him to expect a gun placement is unlikely to mistake the two. An automated program on the other hand will run its instructions through regardless of the consequences, bringing ruin and destruction to neutral and foe alike if the programmer is not very skilled or very lucky.

As a tactic, CNA is asymmetric warfare at its worst (or best) since an attacker in theory only has to have access to a single computer and a single point of com-

* David Lindahl and Mikael Wedlin are Research Scientists at the Swedish Defence Research Agency. The Powerpoint presentations from the lecture are found on the enclosed CD-ROM (eds. comment).

munication to the enemy's C4I-systems (Command, Control, Communication, Computer and Information). Conversely, if one side is not computerized, the more technically advanced side have no way of using CNA against them.

Why is this possible? Would it not be better to build the computer systems so that no one but the authorized persons could control them? Yes, it would, but it is normally not possible. First of all, one of the main advantages of using computer networks is the ability to quickly share or access data. So security measures must work within the needs of data availability. Often there is also a time constraint placed on the services of the system. The need for timely dissemination of possible targets of an air strike is obvious. This too places a burden on the security mechanisms.

Finally we have the bottom line. Computer systems cost money. Large scale production series reduce this cost. Hence, the temptation to use what has been known as COTS, or commercially 'off the shelf' available products instead of specially manufactured military products. These are often relatively cheap, fulfil the time and availability goals and are often add-ons to common programs such as word processors or spreadsheet programs. This means less procurement costs, less training costs and, since the same programs are used for administration in peace time, less support costs.

Unfortunately the COTS-products are designed for civilian office use. This means, in practice, that their security mechanisms are very bad indeed.

Also the use of COTS in many organisations promotes an attitude where security measures are seen as degrading the performance rather than enhancing security. Hopefully this will change over time as the technology matures.

## 6.2 CNA-lecture, an Abbreviated Summary

Computer systems can not easily be secured as an afterthought. This is due to the fact that they are not entirely technical; They are systems consisting of technical parts as well as the users the system interacts with. Also, very few computer systems are isolated from the rest of the world. They tend to be used for some kind of communication with other systems. This leads to a situation where the weakest link in the communication chain set the security level for the whole system of systems.

Users are not normally aware of how their computers work. Most of them view their computer systems as black-box-systems. That is, they never really consider the internal workings of the machine or networks. This is not really related to in what degree they (the users) are dependent on their machines. Many professionals like economists, designers, architects etc are in fact totally dependent on computers to be able to perform their duties. Strangely enough this has not led to any significant desire to understand their tools. As a result the users have great difficulty understanding what behaviour on their parts constitutes an unreasonable security risk.

This is also true of the salesmen and buyers of computer systems who can in fact be likened to 'the blind leading the blind'. The buyers, being reasonable decent persons, try to do their best in acquiring software at minimum cost. The salesmen, the same, try to do what they have in fact been hired to do, sell things. Neither really understands the merchandise and so the software requirements are seldom met. The authors would like to repeat an old parable, only half in jest that the difference between a used cars salesman and a computer salesman is that the car dealer knows when he is lying to the customer. Also he can usually drive a car.

Continuing, the developers of software have no real idea of under what circumstances their software will be used. As a result they develop extremely generalized pieces of software. This is of course also a sound business strategy since more potential customers generally translates to more actual sales.

The problem from a security standpoint is that (due to ignorance) no security concerns are voiced by the customers and so no security concerns are satisfied by the producers. When security concerns are belatedly raised it generally falls to security professionals who try to 'patch a leaking ship' by modifying details of a system already in use, paid for, and therefore not possible to exchange for something better.

## 6.3 Examples of Common Security Flaws in Software

### 6.3.1 Complex Interaction Between Functions

Every time a user interacts with the computer she is in reality accessing many different program files. A large program, such as Microsoft Office, can have several hundred different functions. The amount of different functions in software has been ever increasing over the years. This is in part due to the sales tactics of the software companies. In order to sell ever new versions of their software they use the argument that functionality has been increased, This, it is argued, will make users more productive. In reality few users indeed have ever tried even half the functions at their disposal. Most use only basic word processing, spreadsheet handling and emailing.

Apart from being unused, these different features interact with each other in ways that are very difficult for the average user to foresee. This can results in security breaches such as the use of quick save in the Microsoft Word word processor. If this option is enabled saving a document takes less time. But as a side effect, text erased from the document are not removed from the file but instead marked 'do not show on screen'. This information is easily retrieved by e.g. opening the file in an editor such as notepad. There have been several instances when classified documents have been 'declassified' in this manner (deleting text in a word processor) and then mailed to e.g. journalists.

The situation is not expected to improve in the foreseeable future as every new version of programs and operating systems get larger than the one preceding it. And since more efficient computers with larger memory banks and hard drives are available the customers continue to buy them, thus adding new and improved security flaws to the systems with every upgrade.

### 6.3.2 Inefficient or Non Existent Quality Control Regarding the Software Code

Many programs contain code that should never have been part of the program at all. Examples include flight simulators, games, back doors and other types of functionality. This is in itself sometimes a threat, as in the case with the back doors (remote access functionality included without the users knowledge, enabling an attacker to take control of the system); It is also a symptom of a greater problem: Namely that the normal status of quality control in software development is to determine that the features to be included work, more or less. No controls are normally performed to determine whether or not additional functionality is present. Neither are possible interaction between program components and the effects on the computer system as a whole considered. This is instead covered in the elaborate user agreements where any and all flaws become the problem of the buyer. *Caveat Emptor,* indeed!

### 6.3.3 Antivirus Programs

The use of antivirus programs is widespread; Unfortunately the awareness of how important it is to update the programs, is not. The protection an antivirus program offers can be summarized as the exact opposite of the way a human guard works. A human guard observes the comings and goings of personnel. Anyone not recognized is detained and questioned. An antivirus program works from a list of 'known bad guys'. Anyone not on the list is assumed to be authorized and is not stopped. This of course means that any new threat that was not included on the last update will successfully attack through the antivirus program. Antivirus programs also use a method known as heuristics in which they attempt to identify new virii by 'virus like behaviour'. This method is not working very well yet. The current situation is that with a well-updated antivirus program protection is adequate against virii classified at the time of the update but not against any versions spread after this point in time. As a side note this means that any weapon specially designed for a CNA are likely to be unclassified at the time of use.

### 6.3.4 Modified Hard- and Software

It is very easy for a programmer to modify the behaviour of a program so that it does things without the user being aware of it. The Operating System (e.g. Windows) has hundreds of files and many processes active at a given time. If even one of them can be modified then information can be harvested from the input to the computer and relayed to other computers via computer networks, or hidden in files that are being transported on diskettes or CD-ROM:s. This modification can be accomplished in many ways. Either through tricking the user into installing the modified files bundled with another program, or remote installing through some security hole in the communication software.

If access can be gained to the actual hardware it can be modified in several ways. Most computers are modular so that new accessories can be installed without difficulty. This means that specially modified hardware can be installed just as easy. Modified hardware can do anything from harvesting information, to detonating when a specific signal is received.

Armed forces run a serious risk from this sort of attack since they often buy in bulk. It may be worth the effort for an adversary to bribe someone to modify a component of e.g. keyboards or printers during a shipment or even at the factory in order to get that equipment inside military installations. The CNA team might then via radio remote control the computers and in that way bypass much of the security on the normal network access points. This is of course just one example of how such an attack might happen.

## 6.4 Conclusions

The use of COTS in semi- and high security systems will probably continue. There is no signs that computer users in general will want to learn more about the inner workings of the systems they take for granted. The number of security professionals in the field of computers increase, but not nearly as fast as the number of computers. It is therefore reasonable to assume that the possibilities of CNA will not decrease in the foreseeable future but will increase with the proliferation of computers in societies and in the armed forces of different countries.

# B Oscar – IFIPSec 2006

Martin Karresand och Nahid Shahmehri. Oscar – File Type Identification of Binary Data in Disk Clusters and RAM Pages. *Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006)*, ss 413–424, 2006. Springer Verlag.

# Oscar – File Type Identification of Binary Data in Disk Clusters and RAM Pages

Martin Karresand[1,2], Nahid Shahmehri[1]

[1] Dept. of Computer and Information Science
Linköpings universitet,
Linköping, Sweden
[2] Dept. of Systems Development and IT-security
Swedish Defence Research Agency,
Linköping, Sweden
{g-makar,nahsh}@ida.liu.se

**Abstract** This paper proposes a method, called Oscar, for determining the probable file type of binary data fragments. The Oscar method is based on building models, called centroids, of the mean and standard deviation of the byte frequency distribution of different file types. A weighted quadratic distance metric is then used to measure the distance between the centroid and sample data fragments. If the distance falls below a threshold, the sample is categorized as probably belonging to the modelled file type. Oscar is tested using JPEG pictures and is shown to give a high categorization accuracy, i.e. high detection rate and low false positives rate. By using a practical example we demonstrate how to use the Oscar method to prove the existence of known pictures based on fragments of them found in RAM and the swap partition of a computer.

## 1 Introduction

There are many examples of situations within the information security field where the ability to identify the file type of a data fragment is needed. For example, in network based intrusion prevention the structure of the packet payloads can be used to filter out traffic not belonging to a certain service, and in that way defend against new and yet unknown attacks [1].

Another example would be a military network where only encrypted traffic is allowed. By looking at the structure of the payloads of the data it is possible to determine if there are any services sending unencrypted data over the network, without having to rely on packet headers, or checking well-known ports. It is even possible to do so without interpreting the payload, thereby avoiding the potential security compromise associated with an in-transit decryption.

A third example can be found in the computer forensics field where a forensic examiner might have to do a manual search through several hundreds of gigabytes of unrecognisable data. A possible scenario would be a forensic examiner conducting an examination of a computer with several large hard disks, all deliberately corrupted by the owner, who is suspected of trafficking in child pornographic pictures. The examiner uses the available tools and manages to retrieve a lot of JPEG picture headers and specific markers, but no complete and viewable pictures can be found. In theory the examiner could start brute force matching of every disc cluster against a database of known child pornographic pictures. In reality such a solution would not be feasible since the case would be closed due to time constraints and legal implications long before the matching operation had finished.

As described in the scenario the existing tools (for example PC Inspector [2], Sleuth Kit [3], and The Coroner's Toolkit (TCT) [4], or the commercial Encase [5], File Scavenger [6], and Search and Recover [7]) would not be of much help, because they all need some remnants of a file allocation table or file header to be able to recover lost files, and even if there are headers to be found the data might still be fragmented, making the tools unable to identify more than the header part of the files.

Unfortunately it is not only corrupted hard disks that cause problems for a forensic examiner, the situation has become even more complicated since the introduction of Knoppix and related live CD distributions [8]. These distributions have made it possible to run a full-fledged computer system from a CD, without ever touching the hard disk. All necessary files are loaded into RAM, with a working read/write virtual file system, and hence no traces are left on the hard disk. There are also several methods [9,10,11] for remote execution of code in RAM only on a victim host. Both RAM and swap partitions are fragmented and unstructured by nature, due to the constant changes induced by the CPU's process scheduling. Hence, if one of these techniques is used for illegitimate purposes, finding evidence in the form of complete binary data or executables would be hard considering the limited abilities of the existing tools.

Consequently there is a need for a method making it possible to identify the type of a data fragment by its internal structure only, and to the best of our knowledge there is no tool available that can do that. Our claim is supported by the National Laboratory of Forensic Science (SKL) [12] in Sweden. They have, together with the National Criminal Investigation Department (RKP) [13] in Sweden, expressed an interest in the ability to automatically bring order to a large amount of unsorted binary data. This paper proposes a method for identifying the type of a binary data fragment and subsequently mapping out the contents of various storage media.

The proposed method, called the Oscar method or simply Oscar, has been implemented in a few proof-of-concept pieces of software. Together, these utilities form a toolbox, the Oscar toolbox, consisting of a JPEG data extractor, a byte frequency distribution extractor, a data fragment categorizer, and a search tool for hexadecimal strings. To evaluate the concept we have used the JPEG standard and extracted fragments of a JPEG file from a disk image file. We have also tested the toolbox using the computer forensics scenario mentioned earlier.

## 2 Proposed solution

Hard disks store data in *sectors*, which are typically 512 bytes long. The operating system then combines several sectors into *clusters*. Depending on the size of the partition on the hard disk clusters can vary in size, but currently the usual cluster size is 4 kB, which often also is the size of *pages* in RAM. Using data blocks that are the same size as clusters and RAM pages, the Oscar method builds vector models, called *centroids* [14, p. 845], of different file types based on statistics of their byte frequency distribution. The statistics are calculated using data in vector format, and by using the mean and standard deviation for each file type, unknown clusters or pages can be categorized. The similarity of the centroid and the sample vector to be categorized is then measured by calculating the distance between them in vector space. If the distance is less than a threshold, the sample is categorized as being of the same type as the centroid. To increase the detection rate and decrease the false positive rate, other type-specific features may also be used in the categorization process.

In Fig. 1 the histograms of the centroids of four different file types, .exe, JPEG, .ps, and .zip, are shown. As can be seen the histogram of the text-based .ps file is radically different from the binary-based JPEG, .exe, and .zip files. Please observe that the JPEG histogram is based only on the data part of the files used for the centroid. Even though the histograms of the binary files are fairly similar, they still differ, with the most uniform histogram belonging to the .zip type, then the JPEG, and last the .exe. All three binary file centroids also contain a higher number of 0x00 than other byte codes, probably due to padding. For the .exe the significantly higher rates of 0x00 and 0xff is worth noticing. The JPEG histogram shows a decrease in the higher byte codes, which might be a result of the RGB coding in combination with the lossy compression. Probably the cameras we used are tuned to what might be called a "normal" colour palette, and hence extreme values for one or more of the three parts of a RGB triplet is less likely in ordinary pictures. The lossy compression will also decrease the number of extreme values by reducing high frequencies in the pictures, i.e. rapid changes in colour and saturation. Additionally, the marker segment indicator 0xff is not to be used in the data part, apart from in a few special cases, and is therefore less likely to appear in the data.

As mentioned earlier the JPEG standard has been used to illustrate the method. To further enhance the detection capability of the Oscar method we use the fact that byte 0xff is only allowed in combination with a few other bytes in the data part of a JPEG file. If there is any pair of bytes representing a disallowed combination, the block will not be categorized as JPEG.

In the following subsections we present and discuss some of the main theoretical aspects of the Oscar method, together with some implementation specific features. Oscar is meant to be generalizable, but is at the moment mainly aimed at finding fragmented JPEG picture disk blocks. However, the metric used for categorization is also applicable to other types of data.

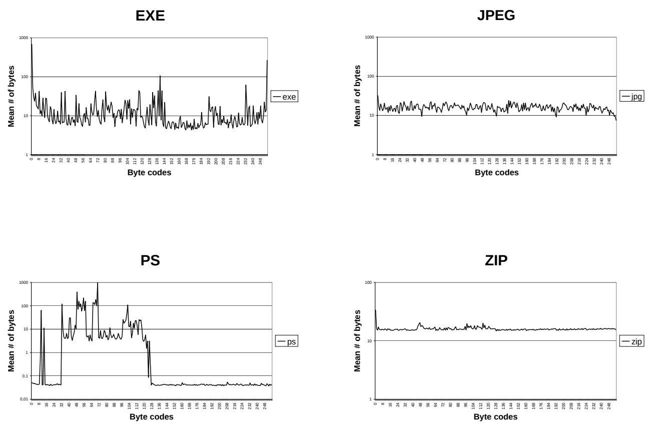


**Figure 1.** Histograms of the centroids of four different file types, .exe, JPEG, .ps, and .zip. Only the data part of the files was used when creating the JPEG centroid. A logarithmic scale is used for the Y-axis.

## 2.1 Metrics

There are many ways of measuring similarity between data samples. The approach chosen for Oscar is to measure the distance between a sample vector and a centroid [14, p. 845]. A weighted variant of a quadratic distance metric is used.

**Creation of Centroid.** The term centroid is defined as the average vector of a set of vectors belonging to a cluster [14, p. 845], i.e. the vectors being categorized as the same type. In our case the centroid consists of two vectors representing the mean and standard deviation of each byte value's frequency distribution.

To create the centroid for the JPEG type we used 255 pictures of normal family life. The pictures came from 5 scanned paper copy photographs and 250 digital photographs from three photographic sessions using two different digital cameras. The pictures' data parts were extracted and concatenated into one file. The resulting file was then used to create a matrix, which was fed to GNU Octave [15] for calculation of the mean and standard deviation. The result of the calculation constituted the centroid.

**Length of data atoms.** The data the Oscar method uses is in the form of 1-grams, i.e. bytes, the simplest form of an $n$-gram. An $n$-gram is an $n$-character long sequence where all characters belong to the same alphabet of size $s$, in our case the ASCII alphabet giving $s = 256$. The byte frequency distribution is derived by bin sorting the bytes into bins of size one. These bins then form a vector representing the sample to be compared to a centroid.

By using 1-grams we do not take the order of the bytes in the data into consideration. Considering the order of the bytes decreases the risk of false positives from disk clusters having the same byte frequency distribution as the data type being sought, but a different structure. Using larger $n$-grams increases the amount of ordering taken into consideration and would consequently be a nice feature to use. However, when using larger $n$-grams the number of bins, $b$ where $b \leq s^n$, also increases. The size of $b$ depends on whether the $n$-grams have a uniform probability distribution or not. The maximum value is required when the distribution is uniform, or nearly uniform and the fact that a JPEG file is compressed gives it an almost uniform $n$-gram probability distribution.

Since the alphabet used has size 256 and we use 4 kB large data fragments, the Oscar method is in practice limited to the use of 1-grams. Using larger $n$-grams requires $b > 4096$, which in this case gives a mean for the frequency of each $n$-gram less than one. Every sequence of characters longer than one byte will therefore have too large an impact on the calculations of the distance, and consequently the method becomes unstable and hence not usable.

**Measuring Distance.** The distance metric is the key to a good categorization of the data. Therefore we chose to use a quadratic distance metric, which we extended by weighting the difference of each individual byte frequency with the same byte's standard deviation. In Equation (1) the standard deviation of byte $i$ is represented as $\sigma_i$. A smoothing factor, $\alpha$, is used to avoid division by zero when $\sigma_i = 0$

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{n-1} (x_i - y_i)^2 / (\sigma_i + \alpha) \quad . \tag{1}$$

The advantage of using a more computationally heavy quadratic-based metric over a simpler linear-based metric is the quadratic-based method's ability to strengthen the impact of a few large deviations over many small deviations. Assuming the vector sums, i.e. $\|\boldsymbol{x}\|_1$ and $\|\boldsymbol{y}\|_1$, are constant, Equation (1) gives lower distance values for two vectors separated by many small deviations, than for two vectors separated by a few large deviations. A linear-based method gives the same distance, regardless of the size of the individual deviations, as long as the vector sums remain the same. Since we are using 4 kB blocks of data $\|\boldsymbol{x}\|_1 = \|\boldsymbol{y}\|_1 = 4096$ and we are, in reality, forced to use a quadratic distance metric to achieve decent categorization accuracy.

Some file types generate fairly similar histograms and it is therefore necessary to know which byte codes are more static than others to discern the differences in spectrum between, for example, an .exe file and a JPEG picture. We therefore have to use the more complex method of looking at the mean and standard deviation of individual byte codes, instead of calculating two single values for the whole vector.

## 2.2 Implementation of Tools

We implemented a toolbox in C containing a number of tools to support testing the Oscar method. The Oscar toolbox consists of a JPEG data extractor, *extr_ data*, a byte frequency distribution extractor, *extr_ 1-gram*, a categorizer, *type_ mapper*, and a hexadecimal string search tool, *hexgrep*.

The *extr_ data* tool steps through the markers in the header of a JPEG file until the data part starts and then extracts the data part to a file. The *extr_ 1-gram* tool is used for the creation of the centroid. It mainly sorts and counts the bytes in one 4 kB block at a time. The output of extr_1-gram is fed to GNU Octave [15], which is used to create the mean and standard deviation vectors constituting the centroid.

The *type_ mapper* tool is given an image file of a disk, RAM or swap partition. The tool then calculates the distance to the centroid for each 4 kB block of the file. At the same time it counts and filters some JPEG-specific markers, which we use to lower the number of false positives from other file types also having more or less uniformly distributed byte frequencies, for example .zip files and encrypted files.

The marker filters of the type_mapper tool are set to only accept the occurrence of Restart (RST) markers, 0xffdx, the End-Of-Image (EOI) marker, 0xffd9, and the marker 0xff00 in the data. Disk clusters where the RST markers do not loop through $x = \bmod 8$ in consecutive order are discarded. Likewise we filter out clusters where there are more than one EOI marker present. There is also a filter for 0xff00 set to require at least one such marker. The threshold is based on tests where we found the mean frequency of that marker to be 2365, with a standard deviation of 1686.1 for a 1 MB block of JPEG data. This corresponds to a mean of 9.2 and a standard deviation of 6.6 for a 4 kB block of data.

The *hexgrep* tool was created because we could not find an easy way to *grep* for hexadecimal data in a file. This would be useful when looking for JPEG Start-Of-Image (SOI) markers or other header related information. The tool could also be used for counting the number of blocks categorized as probable JPEG data in a type_mapper map.

## 3 Test Cases

Two evaluations of the Oscar method and toolbox were performed using two different setups. The first evaluation was done using a file containing several different types of data. The second evaluation was designed as a more practical experiment where two dump files, representing the RAM and swap partition of a computer, were searched for fragments of two JPEG files.

Although the detection rate of the Oscar method is important, the false positives rate is more important because of the intended use of the method. Therefore the test cases were focused on measuring the number of false positives generated in the different settings.

### 3.1 Evaluation Using Prepared File

The evaluation file for the first experiment was created by concatenating 53 files of 49 file types (see Table 1) into one. Three JPEG files were included to enable testing of the detection rate. Two of the pictures were taken by different digital cameras and one was produced by a scanner from an ordinary paper photograph. One of the digital camera JPEG files was previously unknown to the type_mapper tool. The scanner JPEG file contained RST markers (see Sect. 2.2) to test the filtering of markers.

**Table 1.** File types used for the evaluation

| | | | | | | |
|---|---|---|---|---|---|---|
| .ACM | .ENU | .NLD | .SVE | .cpl | .html | .sys |
| .BMP | .INF | .NLS | .TLB | .dll | .jar | .tgz |
| .COM | .INI | .OCX | .UCE | .doc | .mp3 | .tsp |
| .CPI | .ITA | .OLB | .ax | .dsk | .pdf | .txt |
| .DAT | .JPG | .RLL | .bin | .exe | .png | .vbs |
| .DEU | .MSC | .ROM | .bz2 | .gif | .ps | .xls |
| .DRV | .MSI | .SQL | .chm | .gpg | .rpm | .zip |

The files to be included in the evaluation were chosen based on their file size and picked subjectively from a laptop running Windows XP, SP 2. We wanted as large files as possible to decrease the impact of the file headers, because a header's structure of often differs significantly from the data part's structure. The total size of the evaluation file was 70.4 MB, giving an average size of the included files of 1,3 MB.

### 3.2 Evaluation of RAM and Swap Dump

The second experiment was based on the scenario described in Sect. 1, but changed from searching a hard disk to searching RAM and the swap partition of a computer. The evaluation was designed to measure the number of false positives generated when using the Oscar toolbox in a practical setting, and to see whether it was possible to apply the toolbox to such a scenario.

The test was performed by storing and viewing two JPEG pictures in a computer running Knoppix 3.9. Both pictures were taken from the set used to construct the centroid: one was from a digital camera and one was a scanned picture. The two dump files were created and searched for remnants of the two JPEG files. It is worth mentioning is that the swap file Knoppix used was also used by a Gentoo Linux installation on that computer, therefore the file could have contained data fragments from previous sessions.

The tools were applied in the following way:

1. the type_mapper tool was applied to the RAM and swap dump files containing fragments of the pictures,
2. the disk clusters marked as "probably JPEG data" were extracted, and
3. the hexgrep tool was used to find any matches between the extracted fragments and the two JPEG pictures used.

The work-flow described above could be used by a forensic examiner looking for pictures of child pornography. We used pictures portraying a pile of wood and a wedding, not child pornography, but since the algorithms of the Oscar method work on a binary level, the subject of the pictures does not affect the result of the evaluation.

## 4 Result

In this section the results of the two evaluations are presented. We also discuss whether it is possible to regard a match between a 4 kB data fragment and a complete picture as evidence of the complete picture once having existed on the storage media where the fragment was found.

### 4.1 Evaluation Using Prepared File

The evaluation file contained 17608 full 4 kB blocks, of which 476 should be categorized as JPEG. The algorithm categorized 466 of them correctly and produced 1 false positive (see Table 2).

**Table 2.** Result of the evaluation using the prepared file

| | # of 4 kB blocks | % of category |
|---|---:|---:|
| True pos. | 466 | 97.90 |
| True neg. | 17131 | 99.99 |
| False pos. | 1 | 0.01 |
| False neg. | 10 | 2.10 |

All false negatives were generated by the digital camera picture included in the centroid. The reason for the false negatives was that some parts of the picture lacked 0xff00 codes. We repeated the experiment with the 0xff00 filter disabled and got a 100% detection rate, but 20 false positives.

The false positive was generated by the *win32.sys* binary, which happened to have an almost perfectly symmetrical fragment at one place, giving a distance measure of 764.4. The mean distance of the JPEG data in our tests was approximately 1500. In this case using longer *n*-grams might have helped, but the current level of 1 false positive out of 17132 possible might be acceptable.

### 4.2 Evaluation of RAM and Swap Dump

There were 119 disk blocks categorized as possibly containing JPEG data in the RAM dump. When comparing them to the two JPEG files used, all matched. There were no false positives; the reason for this is not completely clear to us because we did not have full control of the contents of the RAM and swap, but we know that there were no other pictures present.

In the swap dump there were 126 fragments categorized as possibly JPEG data. All of them matched the pictures used, and consequently, there were no false positives in this dump either.

The fact that there were no false positives in this test, but a large number of true positives, is encouraging. The results also show that technically the method described in Sect. 3.2 for finding traces of child pornography is working.

There is, however, a legal aspect of the problem: evidence used in a trial must be trustworthy. Thus, if a certain number of fragments belonging to the same known child pornographic picture is found on a hard disk, the disk has probably contained that specific picture. What must be determined, then, is the number of matching fragments necessary to say with certainty that the picture once existed on the disk. While a 100% match cannot be achieved unless the entire image can be retrieved from the disk, it is possible to estimate the possibility that a partial match is equivalent to a complete match.

To find an approximate value for the probability of a partial match we have to start by figuring out the number of pixels stored in a disk cluster sized data fragment, i.e. the compression rate of JPEG. An estimation of the compression rate, $c$, can be made using the equation $c = \frac{3*x*y}{s}$, where 3 comes from the number of bytes used to encode RGB colour, $x$ and $y$ gives the size of the image in pixels, and $s$ is the size of the data part of the JPEG file in bytes. The compression rate depends on the diversity of the pixel values, but for normal pictures, such as those in Fig. 2, we found the compression rate to be in the range of 8 to 15.

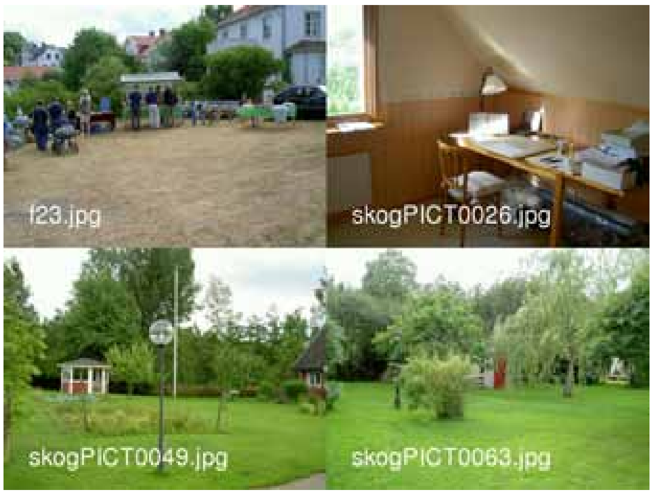


**Figure 2.** The four pictures that shared the same 12 to 22 first bytes in the data part of the files

At that compression rate, 4096 bytes of JPEG data approximately correspond to an area of between 10900 ($104^2$) and 20500 ($143^2$) pixels. Large areas of the same colour and saturation are compressed more by the JPEG algorithm than highly detailed areas where colour and saturation change rapidly. In other words, the entropy of a group of bytes is almost the same regardless of what part of the picture they represent. Therefore the larger part of the bytes of a fragment is made up of details of the picture. If two fragments are equal, it is likely that their details are equal too, and if so we can in reality draw the conclusion that the fragments came from the same source.

One interesting thing to notice is that of the pictures shown in Fig. 2, skogPICT0049.jpg and skogPICT0063.jpg have the first 22 bytes of the data part in common, and f23.jpg and skogPICT0026.jpg share their first 12 bytes. The file skogPICT0026.jpg also shares its first 14 bytes with skogPICT0049.jpg and skogPICT0063.jpg.

## 5 Related Work

Wang and Stolfo have written a paper [1] presenting PAYL, an intrusion detection system built on a method related to the Oscar method. Another paper [16] by Li, Wang, Stolfo and Herzog apply the fundamentals of PAYL to create so called *fileprints*, which are used for file type identification. Similarly to our method PAYL and the fileprints use a centroid modelling the mean and standard deviation of individual bytes. To speed up the file type identification Li, Wang, Stolfo and Herzog experiment with truncating the data and only using a certain number of bytes from the beginning of each data block.

To detect anomalies both PAYL and the fileprints make use of what is called a *simplified Mahalanobi distance* to measure the similarity between a sample and a centroid. The simplified Mahalanobi distance is described by Equation (2), where two vectors, called $x$ and $y$, represent the sample and the centroid respectively:

$$d\left(\boldsymbol{x}, \boldsymbol{y}\right) = \sum_{i=0}^{n-1} \left(\left|x_i - y_i\right| / \left(\sigma_i + \alpha\right)\right) \quad . \qquad (2)$$

The standard deviation of byte $i$ is represented as $\sigma_i$, the value of $\alpha$ is used as a smoothing factor to avoid division by zero when $\sigma_i = 0$. As can be seen this is similar to a linear-based distance metric between the sample and a model, weighted by the standard deviation of the byte frequency distribution.

In Sect. 2.1 we described the problem of a linear-based method giving the same result regardless of whether there is one large byte frequency deviation, or several small deviations. Since Equation (2) is linear-based its results depend heavily on the standard deviations, $\sigma_i$, of the bytes frequencies. If the standard deviation of the $\sigma_i$:s is low, the results of the equation will be almost equal, regardless of the distribution of the deviations of the sample vectors and the centroid. Our metric, where each byte frequency difference is squared, can in a better way handle byte frequency distributions where the standard deviation of the $\sigma_i$:s is low, because it favours many small differences over a few larger ones.

When Li, Wang, Stolfo and Herzog use truncated data for creating fileprints it makes them dependent on header data to be able to categorize files, which in turn requires the files not to be fragmented and preferably the file system to be intact. The Oscar method uses only the structure of a data fragment to identify its file type and therefore can work regardless of the state of the file system or degree of fragmentation.

Another method related to the Oscar method was presented by McDaniel and Heydari [17] in 2003. They use their method for determining the type of contents of different files, but use complete files or the header and trailer parts of files to create the fingerprints used as models. In order to incorporate byte order, to a certain extent, into the fingerprints the authors use what they call *byte frequency cross-correlation*. Most of the paper describes ways to improve the statistical features of the algorithms, enabling them to better handle discrepancies in the byte frequency of single files during creation of the fingerprints.

The main differences between the Oscar method and the method described in [17] is that the latter depends on the header and footer parts of files to create fingerprints.

Two of the more popular open source tools for forensic file carving are *lazarus* and the *Sorter* tool. The first one is part of the Coroner's Toolkit (TCT) [4] and the second tool is included in the Sleuth Kit [3]. Both of them depend on the Unix *file* [18] utility, which uses header information to categorize different file types. Consequently these two tools depend on header information for file type identification, which the Oscar method does not.

## 6 Conclusion and Future Work

We propose a method, called Oscar, for categorizing binary data on hard disks, in memory dumps, and on swap partitions. The method is based on creating models of different data types and then comparing unknown samples to the models. The foundation of a model is the mean and standard deviation of each byte, given by the byte frequency distribution of a specific data type.

The proposed method has been shown to work well in some smaller evaluations using JPEG files. The Oscar toolbox has been used to identify fragments of known pictures in RAM and memory dumps. The same methodology for using the tools can be put to immediate use by the police in the hunt for child pornography.

Future work will include further development of the tools, providing better integration between them as well as extended functionality. The number of centroids of different file types will be increased and eventually included in the type_mapper tool. Another example of planned improvements are the possibility to extract fragments directly to file when generating a map of the input data.

The idea of incorporating ordering in the centroid is interesting and thus the use of 2-grams and larger $n$-grams must be investigated further, as well as other ways of creating the centroid and measure the similarity of samples and centroids. We will also look further into finding a good method to recreate files from fragments found using the Oscar method.

## References

1. Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In E. Jonsson el al., ed.: Recent Advances in Intrusion Detection 2004. Volume 3224 of LNCS., Springer-Verlag (2004) 203–222
2. CONVAR Deutschland: Pc inspector. (`http://www.pcinspector.de/file_recovery/uk/welcome.htm`) accessed 2005-10-31.
3. Carrier, B.: The Sleuth Kit. (`http://www.sleuthkit.org/sleuthkit/index.php`) accessed 2005-10-25.
4. Farmer, D., Venema, W.: The Coroner's Toolkit (TCT). (`http://www.porcupine.org/forensics/tct.html`) accessed 2005-10-25.
5. Guidance Software: Encase forensic. (`http://www.guidancesoftware.com/products/ef_index.asp`) accessed 2005-10-31.
6. QueTek Consulting Corporation: File scavenger. (`http://www.quetek.com/prod02.htm`) accessed 2005-10-31.
7. iolo technologies: Search and recover. (`http://www.iolo.com/sr/3/`) accessed 2005-10-31.
8. Brand, N.: Frozentech's livecd list. (`http://www.frozentech.com/content/livecd.php`) accessed 2005-10-28.
9. grugq: Defeating forensic analysis on unix. Phrack **11**(59) (2002) `www.phrack.org/show.php?p=59&a=6`, last visited 2004-11-19.
10. grugq: Remote exec. Phrack **11**(62) (2004) `www.phrack.org/show.php?p=62&a=8`, last visited 2004-11-19.
11. Pluf, Ripe: Advanced antiforensics : SELF. Phrack **11**(63) (2005) `http://www.phrack.org/show.php?p=63&a=11`, accessed 2005-11-03.
12. Rhodin, S.: Forensic engineer, Swedish National Laboratory of Forensic Science (SKL), IT Group. (several telephone contacts during October and November 2005)
13. Ericson, P.: Detective Sergeant, National Criminal Investigation Department (RKP), IT Crime Squad, IT Forensic Group. (telephone interview 2005-10-31)
14. Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science **267**(5199) (1995) 843–848
15. Eaton, J.: Octave. (`http://www.octave.org/`)
16. Li, W.J., Wang, K., Stolfo, S., Herzog, B.: Fileprints: Identifying file types by n-gram analysis. In: Proceedings from the sixth IEEE Sytems, Man and Cybernetics Information Assurance Workshop. (2005) 64–71
17. McDaniel, M., Heydari, M.: 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) 332.1
18. Darwin, I.: file(1). (`http://www.die.net/doc/linux/man/man1/file.1.html`) accessed 2005-10-25.

# C Oscar – West Point 2006

Martin Karresand och Nahid Shahmehri. File Type Identification of Data Fragments by Their Binary Structure. *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop 2006*, ss 140–147, 2006. IEEE.

Proceedings of the 2006 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY, 21-23 June 2006

# File Type Identification of Data Fragments by Their Binary Structure

Martin Karresand, Nahid Shahmehri

*Abstract*— **Rapidly gaining information superiority is vital when fighting an enemy, but current computer forensics tools, which require file headers or a working file system to function, do not enable us to quickly map out the contents of corrupted hard disks or other fragmented storage media found at crime scenes. The lack of proper tools slows down the hunt for information, which would otherwise help in gaining the upper hand against IT based perpetrators. To address this problem, this paper presents an algorithm which allows categorization of data fragments based solely on their structure, without the need for any meta data. The algorithm is based on measuring the rate of change of the byte contents of digital media and extends the byte frequency distribution based Oscar method presented in an earlier paper. The evaluation of the new method shows a detection rate of 99.2 %, without generating any false positives, when used to scan for JPEG data. The slowest implementation of the algorithm scans a 72.2 MB file in approximately 2.5 seconds and scales linearly.**

## I. Introduction

Our military forces, as well as society as a whole, are becoming more and more computerized, and as the number of networked computers increases, the number of IT related crimes will increase and hence the importance of computer forensic examinations will follow suit. A forensic analysis often involves searching for information on media that has been corrupted or fragmented in some way, typically in network traffic, on crashed hard disks, and in RAM dumps. Identifying the type of an individual data block taken out of its context is hard. To the best of our knowledge, none of the currently available forensic tools can identify the file type of single data fragments without the use of header information or other meta data [1][2][3][4].

The following scenario describes the previously mentioned current tools and the problem of identifying the type of data fragment found on a hard disk. A computer containing material revealing detailed security plans of an ongoing G8 meeting is broken into by a professional hacker group financed by a terrorist organisation. The hackers extract the information they are looking for, and leave a program running which constantly creates, moves and deletes files on the hard disk. The break-in is detected only minutes after it has occurred, but due to the massive scrambling of the contents of the hard disk it has become heavily fragmented and most of the traces are destroyed. To be able to gain the upper hand against the terrorists the military and police urgently need to know what information was stolen and how, and that information is hidden within the unsorted binary data on the hard disk. Due to the limited abilities of the present tools the forensic examiners working on the case have to manually analyse the data fragments one by one.

As described in the scenario, time is certainly an issue in military and police operations. The tactics can be summarized as "continually reducing the amount of time it takes for our military to observe and respond to the enemy's actions so that the adversary's ability to react is outpaced by our military actions" [5, p. 2]. Military forces also desperately need to be able to detect data hidden within network traffic [5]. Also, detection and extraction of data hidden in transactions or streaming media is regarded as an area where more research is needed. This is also the case for tools enabling the forensic examiners to rapidly get an overview of the contents of seized media [5]. Hence the main requirements for a successful military operation similar to the one described in the scenario are:

- a quick response to minimize the damage,
- an overview of the affected media, and
- the ability to retrieve information regardless of the severity of the damage.

To contribute to the development of the computer forensic field we started working on the Oscar [6] method, which enables identification of the file type of data fragments solely by their internal structure, without the need of any meta data. The original principle [6] is to use the byte frequency distribution (BFD) of data fragments and calculate the mean and standard deviation for each byte value. These statistical measurements together form a model, called a centroid, of a chosen file type. The centroid is then compared to an unknown data fragment and the distance between the sample and the centroid is calculated. If the distance is lower than a predefined threshold the data fragment is categorized as being of the same file type that the centroid represents.

Currently the method is implemented with the identification of JPEG data in mind by using the BFD algorithm with a rule set created from the JPEG standard specifying

M. Karresand: Dept. of Computer and Information Science, Linköpings universitet, Linköping, Sweden and Dept. of Systems Development and IT-security, Swedish Defence Research Agency, Linköping, Sweden; *g-makar@ida.liu.se*
N. Shahmehri: Dept. of Computer and Information Science, Linköpings universitet, Linköping, Sweden; *nahsh@ida.liu.se*

the occurrence of some specific marker byte pairs in the data part of JPEG pictures. The principle of the method is, however, applicable to any kind of binary data.

The Oscar method allows a forensic examiner to get an almost instant overview of the contents of a hard disk, memory dump, or any other piece of evidence from a crime scene. The method can identify the file type of fragments of raw data that ordinary header based tools cannot.

This paper presents a new algorithm which extends the Oscar method. The algorithm works by calculating the rate of change (RoC), i.e. derivative, of the bytes in the data fragment and in that way incorporate the ordering of the bytes into the identification process. The algorithm is then implemented using two different metrics to measure the distance between an unknown sample and a centroid. The first distance metric builds on calculating the 1-norm of the rate of change in the data fragment. The second metric uses the frequency distribution of the rate of change of the fragment to be identified.

The RoC algorithm is compared to the original BFD algorithm of the Oscar method. To relate the comparison of the different algorithms to the military requirements described above we measure the execution time of the implementations, as well as their ability to identify the file type of fragmented data. To make the evaluation more useful we have combined the algorithms and distance metrics in different ways to try to find the best possible setup.

## II. Method

This section describes the algorithms used for the Oscar method. Two different approaches have been used; the original algorithm [6] which uses the byte frequency distribution of the data stream, and the new approach which uses the rate of change between consecutive bytes in a fragment. The rate of change method is also combined with two different distance metrics.

The byte frequency distribution method can be seen as a histogram of different byte values of a specific file type. We use the mean and standard deviation of each byte value count to model a file type. The mean values are then compared to the byte count of an unknown sample and the differences are squared and weighted by the standard deviations of the byte counts in the model. The sum of the differences is compared to a predefined threshold and the sample is categorized as being of the same type as the modelled file if the sum is less than the threshold.

The rate of change method looks at the absolute values of the difference in byte value between two consecutive bytes. The number of rates of change values of a model and a sample are compared using the same weighted sum of squares distance metric as the byte frequency method uses. We also experiment with an alternative distance metric using the 1-norm of the differences between a model and a sample.

### *A. Byte frequency Distribution*

The byte frequency distribution algorithm counts the number of occurrences of each byte value between 0 and 255 in a block of data. In this way a histogram of the data block is created. By measuring the mean and standard deviation of each byte value a model representing a specific data type is created. This model is called a centroid and is compared to a sample of an unknown data fragment. To measure the distance between a centroid and a sample a quadratic distance metric is used. Each difference value is weighted by the standard deviation of the byte value in question. The metric measures the difference between the mean value vector, $\vec{c}$, of the centroid and the byte frequency vector, $\vec{s}$, of the sample. The standard deviation of byte value $i$ is represented by $\sigma_i$ and to avoid division by zero when $\sigma_i = 0$ a smoothing factor $\alpha = 0.000001$ is used. The metric is described as

$$d\left(\vec{s}, \vec{c}\right) = \sum_{i=0}^{n-1} \left(s_i - c_i\right)^2 / \left(\sigma_i + \alpha\right). \qquad (1)$$

The size of the data blocks are currently 4 kB to match the commonly used size of RAM pages and disc clusters. The method does not require any specific data block size, but a block size of 512 bytes is recommended as a lower bound.

A special JPEG extension is used together with the byte frequency distribution algorithm. This extension utilizes some of the JPEG standard specifics regulating the use of special marker byte pairs, 0xFFxx, in JPEG files. These markers are used in the header of a JPEG file and also to some extent in the data part. By counting the number of occurrences of some markers we can significantly improve the detection rate and lower the false alarm rate of the algorithm.

The algorithm and the JPEG extension are described in more detail in [6].

### *B. Rate of Change*

We define the rate of change as the absolute value of the difference between two consecutive byte values in a data fragment.In this way the ordering of the bytes is to some extent taken into consideration, but the algorithm cannot tell what byte values give a specific rate of change, apart from a rate of change of 255, of course. Neither can the rate of change method tell whether the change is in a positive or negative direction, i.e. a positive or negative derivative of the byte stream.

The reason for using the absolute value of the byte value difference and not a signed value is that since the difference value range is bounded, the 1-norm distance metric will eventually become zero for long enough sequences. We therefore decided that the loss of resolution induced by the absolute value was acceptable compared to not being able

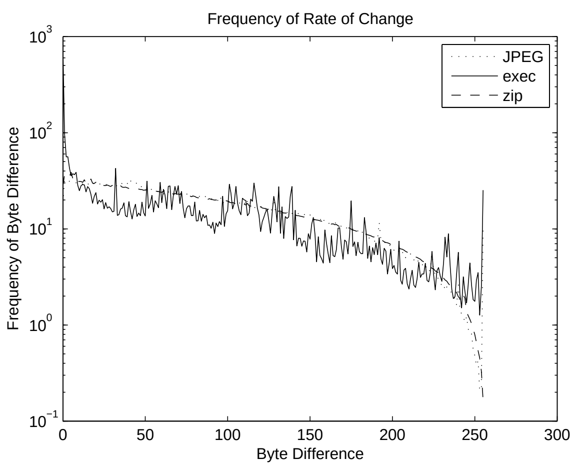


Fig. 1. The frequency distribution of the rate of change for three different file types. The Y-axis is plotted with a logarithmic scale.

to use the 1-norm distance metric. Another reason is that the higher resolution of a signed value difference is obscured by the statistical metrics used.

The frequency distribution of the rate of change is used in the same way as the byte frequency distribution in the first version of the Oscar method [6]. A centroid is created, containing a mean vector and a standard deviation vector. The centroid is then compared to a sample vector by measuring the quadratic distance between the sample and the centroid, weighted by the standard deviation vector of the centroid, as described in Equation 1.

Figure 1 shows the frequency distribution of the rate of change for three different file types. The figure indicate that it is possible to tell compressed and uncompressed file types apart by using this method. As can be seen in the figure, the slopes of the curves differ between the compressed (JPEG and .zip) files, and the executable Windows files. It is also possible to see that the .zip file has a smoother curve, i.e. lower standard deviation, than the JPEG file. The reason for the monotonically decreasing curves is the fact that, assuming a fully random and uniform byte distribution, the probability of difference $x$ is described by

$$p(x) = \frac{256 - x}{\frac{(256+1)\cdot 256}{2}}\ ;\ \ x = [0, 1, \ldots, 255].$$

This method is meant to be combined with the original byte frequency distribution method used for the Oscar method. It is possible to use the same quadratic distance metric for both methods and in that way make the algorithm simpler and easier to control. The combination is made as a logical AND operation, because our idea is to let the detection abilities of both algorithms complement each other, thus cancelling out the weaknesses, i.e. reduce the number of false alarms. Depending on the similarity of the true positive sets of the two methods, the improvement will be more or less significant. The detection rate decreases if one of the algorithms has a close to optimal true positive set, while the true positive set of the other algorithm is less optimal. If we prioritize the detection rate we should use a logical OR operation instead.

## III. Experiment

The experiments are meant to test the requirements of computer forensic tools, as illustrated by the scenario. Therefore, we have designed the tests resemble real life situations as much as possible and focused on measuring the detection rate in combination with the false positives rate, as well as the execution time of the algorithm implementations.

The detection rate and false positives rate of the BFD method was tested in [6] using a combination of many different file types. We used the BFD method with the JPEG extension to scan a file consisting of approximately 50 different file types concatenated into one and counted the number of correctly detected 4 kB fragments versus the number of false positives.

The same evaluation principle as for the BFD method, but with an updated test file, was used for the evaluation of the RoC method described in Sec. II-B. Due to the updated test file we also repeated the tests of the BFD method to enable a comparison of the two algorithms. The experiments measured the execution time, scalability, and detection rate versus the false alarm rate of different combinations of the two methods. The settings used were BFD with JPEG extension, BFD, RoC, RoC with JPEG extension, BFD in combination with RoC, and BFD with JPEG extension in combination with RoC.

The distance thresholds to be used for each algorithm and implementation were varied to allow us to find the best performance of the different algorithms. The threshold values for the BFD algorithm were incremented in steps of 256 from 768 to 2560 and then in steps equalling a 20 % increase of the previous value up to 11007. For the RoC quadratic distance metric threshold we used increments of 128 from 640 to 3584.

A file consisting of 57 individual files concatenated into one was used to evaluate the detection rate versus the false positives rate of the combinations of the two methods. The length of each file was adjusted to fill a multiple of 4 kB blocks by padding the files with zeros at the end. The adjustment of the size was applied to emulate the architecture of a unfragmented hard disk with 4 kB clusters. Due to the fact that we use fixed block sizes instead of a sliding window passing over the data in the storage media the amount of fragmentation of the test file does not matter. We therefore regard the test file as being a realistic representation of a hard disk, although very small compared to the sizes of present disks. The size of the concatenated evaluation file was approximately 72.2 MB. The included file types can be found in Table I. They were all taken from a standard office computer running Windows XP SP 2.

TABLE I
File types used for the evaluation

| | | | | | | |
|---|---|---|---|---|---|---|
| acm | cpl | exe | jpg | pdf | sve | vbs |
| ax | dat | gif | mp3 | png | sys | xls |
| bin | deu | gpg | msc | ps | tgz | zip |
| bmp | dll | html | msi | rll | tlb | |
| bz2 | doc | inf | nld | rom | tsp | |
| chm | drv | ini | nls | rpm | txt | |
| com | dsk | ita | ocx | scr | uce | |
| cpi | enu | jar | olb | sql | wav | |

The method combinations were tested with centroids for JPEG pictures, Microsoft executable format files, and .zip files created by the WinZip utility [7] using default settings. The executable files all started with the magic number "MZ", 0x4D5A. They were labelled as either "MS-DOS executable (EXE), OS/2 or MS Windows", or "MS-DOS executable (EXE)" by the *file-4.12* Unix utility. The files were chosen based on their size, which preferably should be at least 1 MB to give a large consecutive amount of similar data in relation to the amount of header data and zero padding.

The templates we used to measure the detection rate were produced by manually scanning the evaluation file and marking each 4 kB block as a hit or miss depending on the file type to match. Fragments of JPEG data were marked as hits, but if there were any disallowed header markers present in a fragment it was marked as a miss, even if the main part of the fragment consisted of data. This was done to mimic the desired behaviour of the algorithms. Regarding the templates for executable Windows files and .zip files we used the Unix file tool to categorize them and then marked all of the fragments belonging to the chosen file type as hits, even if parts of a fragment consisted of padding or header information. The ground truth used for the experiments regarding the number of hit marks in the template files can be found in Table II.

TABLE II
The number of hit marks for different file types in the template files forming the ground truth for the experiments

| Template file | HIT |
|---|---|
| JPEG | 1150 |
| Windows exec. | 4397 |
| Zip | 1426 |
| Other | 11510 |

The JPEG pictures included in the evaluation file were photographed using the same cameras as were used for the pictures forming the statistical base for the JPEG centroid. The pictures in the evaluation file were, however, excluded from the training data so as not to train the algorithm on the same data as was used to test it.

The following subsections describe each of the experiments in more detail.

## A. Algorithms Using a Quadratic Distance Metric

We used the BFD algorithm both with and without the JPEG extension in the tests. The BFD algorithm with JPEG extension is the algorithm used in the original Oscar method [6] and has therefore been evaluated before. The new test was performed to update the results to allow a comparison of the new algorithm to the original. For the reason of symmetry we also had to perform at test where the BFD algorithm was run without the JPEG extension, since the RoC algorithm is meant to run without the JPEG extension.

The test featuring the RoC algorithm with JPEG extension was performed to evaluate the possible improvement over the original BFD method of the detection rate by taking the ordering of the bytes into consideration. Although this algorithm does not allow detection of file types other than JPEG, the experiment allows us to make a fair comparison of the RoC against the BFD where the latter has previously given a detection rate of 97.9 % with 0.01 % false positives [6]. To make it possible to compare the detection rate of the BFD algorithm to that of the RoC frequency distribution on different file types, a test with the RoC algorithm in stand-alone mode was performed.

By combining the RoC and BFD algorithms we wanted to see if there was any increase in detection rate, and if that was the case, what amount of execution time was needed to give that increase. The two algorithms were implemented in the same loop in the program, and using the same read and write functions. The impact on the execution time of having two instead of one algorithm was therefore minimized. To give a worst case scenario regarding the execution time the program was compiled without any optimization.

Since the algorithms implemented with the JPEG extension were specialized at detecting JPEG data fragments we excluded these algorithms from the tests involving executable files and zipped files.

The two algorithms and their combination all used the quadratic distance metric presented earlier. The exclusion of the JPEG extension from one half of the experiments enabled testing using centroids for both JPEG, .zip, and Windows executable files.

## B. Algorithms Using 1-Norm as Distance Metric

These tests were performed to see whether it was possible to lower the burden on the processor by using the 1-norm as distance metric, without sacrificing the detection rate.

The algorithms used were the RoC frequency distribution in stand-alone mode as well as a combination of the BFD algorithm and the RoC algorithm. The combination was used to observe the effects on the detection rate and execution time. We did not use the JPEG extension, which enabled us to perform the experiments using three centroids representing JPEG, executable Windows files, and .zip files.

## C. Execution Time and Scalability

The execution times of the different combinations of the algorithms and distance metrics were measured using the *clock* C function. Each execution of one of the binaries reported the time it took to operate on the 72.2 MB test file. In that way we could measure the average execution time and thus diminish the effects of other processes in the computer. The clock function enabled us to measure with a precision of 0.01 second.

To evaluate the scalability of the methods the test file was used to form a file four times its original size by concatenating it to itself. The same tests as for the detection rate experiments were then run on the new evaluation file and the execution time was measured. Once again the *clock* function was used.

## IV. Result and Discussion

In this section we first present the results of the evaluations and then discuss them in a separate subsection. The results of the evaluation with JPEG specific implementations of the algorithms gave detection rates from 86.8 % and upwards, but at the same time some of the false positives rates exceeded 20 %. The executable file implementations gave detection rates in the range of 45 % to 85 % with false positives rates approaching 35 %. The zip file specific implementations gave detection rates that in some cases came close to 13 %, and false positives rates between 0.5 % and 1.9 %. The execution times for the different algorithms were approximately 1.5 seconds for the single algorithm implementations and twice as long for the double algorithm implementations. The algorithms scaled linearly.

## A. JPEG Centroid

Due to the fact that there was only one false positive in one of the tests involving the JPEG extension, we choose to present the results in a table instead of a graph. The results of the evaluation using the JPEG template and centroids can be seen in Fig. 2 and Table III. The RoC algorithm outperformed the BFD algorithm in all tests involving JPEG centroids, but combining the two algorithms did not give any additional effect on the detection rate and false positives rate.

If we compare the results of the stand alone implementations of the two algorithms with and without the JPEG extension we see that it increases the detection rate with

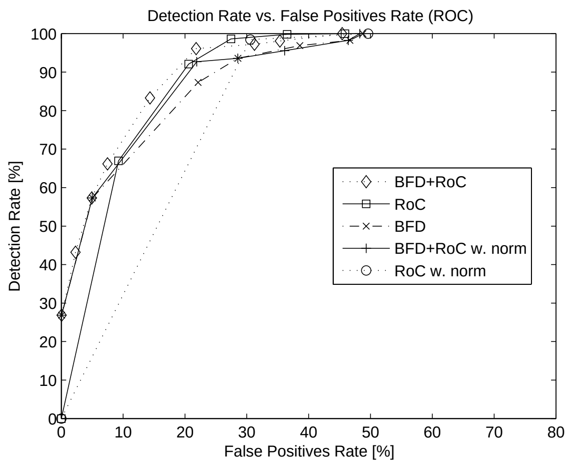


Fig. 2. The detection rate vs. the false positives rate of the five algorithm implementations for JPEG files, without the JPEG extension

almost 8 % for the RoC algorithm and almost 12 % for the BFD algorithm. The most striking effect is the fact that the JPEG extension almost eliminates the false positives, which in the worst case decrease from approximately 20 % to 0.005 %.

TABLE III
The detection rate and false positives rate of the algorithms for JPEG files, with the JPEG extension

| Algorithm | Thresh. | TP % | FP % |
|---|---|---|---|
| BFD JPEG | 1024 | 97.4 | 0.005 |
| RoC JPEG | 2304 | 99.2 | 0 |
| BFD/RoC JPEG | 6369/1536 | 99.0 | 0 |

## B. Windows Executable Centroid

The results from the experiments with the Windows executable centroid can be seen in Fig. 3. The BFD algorithm is more than twice as good at detecting data fragments of Windows executables as the RoC algorithm, although the RoC algorithm with the 1-norm distance metric has a more than 20 % better detection rate than the RoC with a quadratic metric. The RoC with 1-norm also decreased the amount of false positives almost four times compared to the RoC with a quadratic distance metric. The BFD algorithm has almost the same false positives rate as the RoC with the 1-norm distance metric.

Generally no algorithm achieved more than 12.6 % detection rate for the Windows executables, but the false positives rate never exceeded 1.9 %. As soon as the detection rate passed 12 % the false positives rate increased tremendously, exceeding the detection rate.

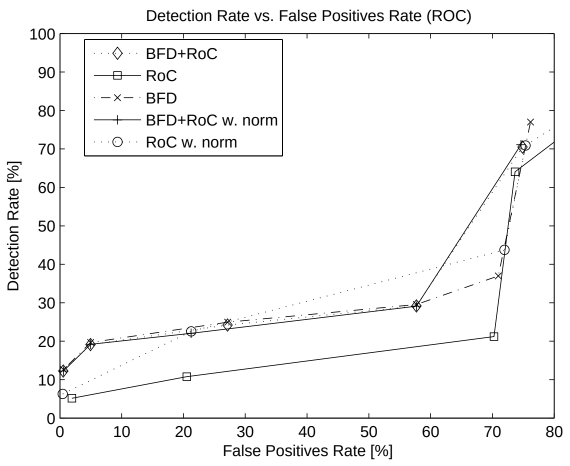


Fig. 3. The detection rate vs. the false positives rate of the algorithms for the Windows executable files

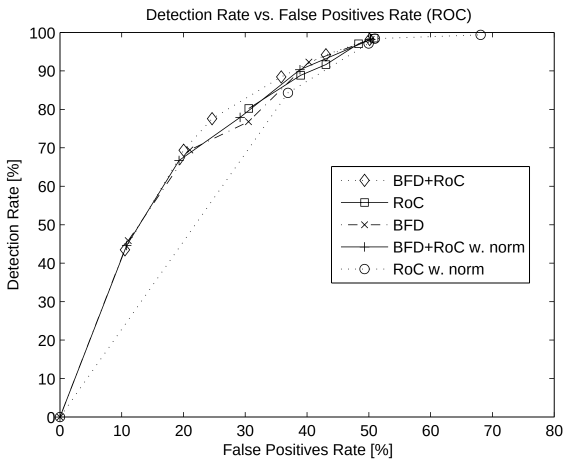


Fig. 4. The detection rate vs. the false positives rate of the algorithms for the .zip files

## C. Zip File Centroid

The results from the experiments with the .zip file centroid can be found in Fig. 4. As can be seen the detection rate lies between 46 % and 84 % which is much higher than for the Windows executable file experiments. However, the relatively high detection rate is followed by a high false positives rate, which is more than ten times higher than for the executable files and lies in the range of 11 % to 37 %. The higher detection rate values are given by the RoC algorithms, which are almost twice as good as the BFD based algorithms. On the other hand the BFD algorithms have a false positive rate three times lower than the RoC algorithms.

## D. Execution Time and Scaling

The approximate execution times for the different implementations can be found in Table IV. The values show that the JPEG extension slows the execution time down by a few hundredths of a second and that the 1-norm distance metric improves the execution time by between 5.9 % and 6.8 %. The compilation of the binaries was performed without any optimization flags set, hence it is possible to improve the results.

TABLE IV
The execution times for the different implementations of the two algorithms

| Algorithm | Exec. time [sec] |
|---|---|
| BFD JPEG | 1.27 |
| RoC JPEG | 1.75 |
| BFD | 1.20 |
| RoC | 1.70 |
| RoC norm | 1.60 |
| BFD/RoC | 2.50 |
| BFD/RoC norm | 2.33 |
| BFD/RoC JPEG | 2.48 |

The experiments testing the scalability of the algorithms showed that the execution time became four times longer when a four times bigger file was scanned. All 18 tests involving different algorithm combinations and distance metrics showed the same trend regarding the execution time. We therefore draw the conclusion that the algorithms scale linearly.

## E. Discussion

The single false positive we got when using the BFD algorithm with the JPEG extension came from a Windows executable file block containing a table with monotonically increasing byte values, which gave the block a saw-tooth form when plotting the byte values consecutively. Consequently the block had an almost uniform byte frequency distribution that made it fit the JPEG centroid very well. The saw-tooth form of the byte sequence also automatically avoided any disallowed JPEG marker combinations, since the hexadecimal value 0xFF was followed by 0x00. The RoC algorithm on the other hand detected the very smooth flow of the bytes in the block and categorized it as non-JPEG.

Low entropy, i.e. compressed, files like .zip and JPEG are hard to tell apart due to the their almost uniform and random byte distribution. The statistical values derived from them give centroids that have almost no significant features. But the lack of features makes them easy to distinguish from more ordered file types like executables and text files. It is also possible to distinguish JPEG files from .zip files due to the rather significant use of 0xFF00 to avoid disallowed byte pairs in JPEG files.

Measuring the distance between the centroid and a sample using the 1-norm did not produce good results for JPEG and .zip, but performed better than the quadratic distance metric when applied to the Windows executable centroid. The reason for that is the broad spectrum of Windows executables, which generate centroids having larger standard deviation values than the other two file types do. Executable files exhibit a larger number of very low and very high frequency changes than do JPEG or .zip files, making the quadratic distance metric less stable. The difference in frequency distribution of the three file types can be seen in Fig. 1.

One reason for the bad performance of the 1-norm distance metric regarding JPEG data fragments can be the fact that the norm is based on a summation of the parts of the vector of the rate of change. The norm can therefore not tell where the centre of gravity of the vector is placed and hence different samples can often get the same distance value. This is true even for samples not being of the same type as the centroids.

The execution time measurements shall all be seen as worst case scenarios, because we did not use any optimization of the code at compile time, nor did we actively optimize the algorithms to any larger extent. In a test related to what we present as future work we optimized the algorithms and used the -O3 flag when running gcc, resulting in an improvement of 5000 % regarding the execution time.

## V. Related Work

The intrusion detection system PAYL [8], [9], [10] is related to the Oscar method through the use of the byte frequency distribution. The system uses a distance metric called the *simplified Mahalanobis distance* to measure the similarity between a sample and a centroid. The metric is based on two vectors, called $x$ and $y$, representing the sample and the centroid respectively, and the standard deviation, $\sigma_i$, of each byte $i$ in the centroid. A smoothing factor $\alpha$ is used to avoid division by zero when $\sigma_i = 0$. As can be seen in Equation 2 this is similar to a linear-based distance metric between the sample and a model, weighted by the standard deviation of the byte frequency distribution.

$$d(\vec{x}, \vec{y}) = \sum_{i=0}^{n-1} \left( |x_i - y_i| / (\sigma_i + \alpha) \right) \quad . \tag{2}$$

PAYL has been implemented using different approaches for how to handle the centroid creation. The later implementations use k-means clustering in two steps to reduce the memory requirements and increase the modelling accuracy.

The main differences between the Oscar method and PAYL are the modelling of the data, the distance metric, and the centroid creation process. PAYL is based on the byte frequency distribution of the data stream, which does not take the ordering of the bytes into consideration. The new Oscar RoC method measures the rate of change in the data stream and can therefore distinguish between data blocks having similar byte frequency distributions, but different ordering of the bytes. The simplified Mahalanobis distance metric used by the PAYL intrusion detector suffers from an inability to separate samples having a few large deviations measured against the centroid from samples having several smaller deviations, in cases where the mean and standard deviation values of the centroid are almost uniform. This is the case when dealing with compressed or encrypted data, for example JPEG or .zip files. The Oscar method uses a quadratic distance metric which is better at handling almost uniform byte frequency distributions. Due to the weaknesses of the distance metric the PAYL method has to use a more complex and computationally heavy centroid creation process than the Oscar method.

The group that created PAYL has also used the byte frequency distribution method for creating *fileprints* [11], [9], which are used to determine the file type of unknown files, in the same manner as the first version [6] of the Oscar method. The fileprint method has borrowed its base from the byte frequency distribution of PAYL, together with the simplified Mahalanobis distance metric. The method is further improved by more advanced centroid creation methods, of which the two-layer method of PAYL is one. Furthermore the authors experiment with truncating the files to be categorized by the fileprint method. They find that by using only a few hundred bytes from the beginning of the files they get higher detection rates than when applying the method to full-length files.

Since the fileprint method and PAYL were developed in parallel the fileprint method also shares its main differences relative to the Oscar method with PAYL. In addition to the differences in the modelling of the data, the distance metric, and the centroid creation process, the fileprint method also differs from the Oscar method in the previous method's implicit dependence on file headers through the truncation used. The Oscar method is meant to do what the current tools cannot do, namely categorize data fragments without having to rely on header data and file blocks being stored consecutively, i.e. the Oscar method can work regardless of the state of the file system or degree of fragmentation. By using only the first few hundred bytes of a file the fileprint method in reality becomes a reinvention of the Unix *file* command.

McDaniel and Heydari [12] present a method which is also used for determining the type of unknown files. Their method uses the header and footer parts of files to create fingerprints of files in a similar fashion to the fileprint method presented above. In order to incorporate byte order, to a certain extent, into the fingerprints the authors use what they call *byte frequency cross-correlation*, which measures the amount of dependence between bytes in files. The authors give the html tag markers "<" and ">" as examples of bytes with strong correlation. Most of the paper describes different methods to improve the statistical features of the algorithms. By improving the statistical features McDaniel and Heydari try to better handle discrepancies in the byte frequency of single files during creation of the fingerprints.

The different methods used by McDaniel and Heydari for improving the statistical features of the algorithms improve the result, but make the method less general. This duality is similar to the benefits and problems involved in the use of the JPEG extension of the Oscar method. The main difference between the Oscar method and McDaniel's and Heydari's method is that the latter depends on the header and footer parts of files to create fingerprints, while the Oscar method does not.

Shamir and Someren [13] use the entropy of hard disk data fragments to locate the cryptographic keys stored on disk. They use a sliding window of 64 bytes and count the number of unique byte values in the window. A cryptographic key will generate a higher entropy than other binary data and in that way it is possible to differentiate keys from other data.

The Shamir and Someren method differs from the Oscar method in that it uses smaller data blocks and does not need any centroid to measure new, unknown samples against. Also the area of usage differs, although the Oscar method could be used for similar tasks. On the other hand, the Shamir and Someren method cannot be used for identifying the file type of an unknown data fragment and is specialized at detecting cryptographic keys.

## VI. Conclusion and Future Work

We have tested a new concept which enables the inclusion of the ordering of the bytes in a data fragment into the Oscar method used to identify the file type of data fragments. The principle on which the new algorithm is based measures the difference in byte value between two consecutive bytes and calculates the frequency distribution of the different rates of change. There is no upper bound on the size of the data fragments that can be used, but a recommended lower bound on 512 bytes.

The evaluation of the RoC method shows that it is slightly better than the byte frequency distribution method currently used in the Oscar method. The BFD method gives a detection rate of 87.3 % and a false positives rate of 22.1 %. When we use the RoC method the detection rate increases to 92.1 % with a false positives rate of 20.6 %. If we add the JPEG extension to the RoC method the detection rate increases to 99.2 % and the false positives rate decreases to 0 %.

Because of the promising results we will continue exploring the possibilities of using the ordering of the bytes to further improve the detection rate and lower the false alarms. One interesting area is the use of 2-grams [14]. Preliminary results suggest a possibility even to distinguish different digital camera makes from each other simply by looking at the structure of the data part of JPEG photographs.

We will also look into the area of reassembly of fragmented data. There are indications that the RoC method or the use of 2-grams can be helpful in finding the proper connection of the fragments found by the Oscar method.

To further evaluate the methods we will apply them to real data in cooperation with the police and military IT security forces.

## References

[1] I. Darwin, "file(1)." http://www.die.net/doc/linux/man/man1/file.1.html. accessed 2005-10-25.

[2] Air Force Office of Special Investigations and The Center for Information Systems Security Studies and Research, "Foremost - latest version 1.0." http://foremost.sourceforge.net/, 2005. accessed 2005-10-12.

[3] B. Carrier, "The Sleuth Kit." http://www.sleuthkit.org/sleuthkit/index.php. accessed 2005-10-25.

[4] D. Farmer and W. Venema, "The Coroner's Toolkit (TCT)." http://www.porcupine.org/forensics/tct.html. accessed 2005-10-25.

[5] J. Giordano and C. Maciag, "Cyber forensics: A military operations perspective," *International Journal of Digital Evidence*, vol. 1, Summer 2002. http://www.utica.edu/academic/institutes/ecii/publications/articles/A04843F3-99E5-632B-FF420389C0633B1B.pdf, accessed 19 Mars 2006.

[6] M. Karresand and N. Shahmehri, "Oscar – file type identification of binary data in disk clusters and ram pages," in *Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006)*, LNCS, p. To appear, 2006.

[7] WinZip International LLC, "The Zip file utility for Windows." http://www.winzip.com/. accessed 25 April 2006.

[8] K. Wang and S. Stolfo, "Anomalous payload-based network intrusion detection," in *Recent Advances in Intrusion Detection 2004* (E. Jonsson el al., ed.), vol. 3224 of *LNCS*, pp. 203–222, Springer-Verlag, July 2004.

[9] S. Stolfo, K. Wang, and W.-J. Li, "Fileprint analysis for malware detection," tech. rep., Computer Science Department, Columbia University, New York, NY, USA, 2005. Review draft.

[10] K. Wang, G. Cretu, and S. Stolfo, "Anomalous payload-based worm detection and signature generation," in *8th International Symposium on Recent Advances in Intrusion Detection, RAID 2005* (A. Valdes and D. Zamboni, eds.), vol. 3858 of *LNCS*, pp. 227–246, Springer Verlag, 2006.

[11] 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*, pp. 64–71, June 2005.

[12] 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), p. 332.1, IEEE Computer Society, 2003.

[13] A. Shamir and N. Someren, "Playing 'hide and seek' with stored keys," in *Financial Cryptography: Third International Conference, FC'99* (M. Franklin, ed.), vol. 1648 of *LNCS*, pp. 118–124, Springer-Verlag, 1999.

[14] M. Damashek, "Gauging similarity with n-grams: Language-independent categorization of text," *Science*, vol. 267, pp. 843–848, Feb. 1995.

# D WLAN testbed – West Point 2006

Erika Johansson och Mats Persson. Test Bed for Assessment of CNO and EW Against Emulated Wireless Ad Hoc Networks. *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2006*, ss 318–325, 2006. IEEE.

Proceedings of the 2006 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY

# Test Bed for Assessment of CNO and EW Against Emulated Wireless Ad Hoc Networks

Erika Johansson and Mats Persson

*Abstract*— **This paper describes a test bed for assessment of computer network operations (CNO) and electronic warfare (EW) against wireless ad hoc networks. The test bed allows real applications to exchange real traffic over the emulated wireless network. Examples of test bed use are presented, along with some initial results.**

*Index Terms*—**ad hoc network, computer network operation, electronic warfare, test bed**

## I. INTRODUCTION

INFORMATION EXCHANGE by means of computer networking is common today. For some users, especially in the military sector, the networks must be wireless to allow high user mobility. In situations where little or no usable infrastructure is present, an ad hoc radio network is a way to provide a functioning communications network.

When studying CNO and EW against wireless networks, it is impractical and difficult to build an entire network in a laboratory because the dynamic conditions in the field are nearly impossible to recreate. Moreover, extended field trials are both expensive and time-consuming.

Another approach is to study wired networks and then try to translate the results into the wireless domain. However, there are two important fundamental differences between wired and wireless networks. Firstly, a wireless network has less capacity, and the capacity must be shared because nodes can interfere with each other's transmissions. Secondly, a wired network has constant (and good) connection quality and is fully connected. It is thus not possible to study how applications or computer systems are affected by the varying conditions of the radio media. The potential for studying CNO or EW opportunities against wireless equipment is also lost.

In order to capture the radio media characteristics in a laboratory, the wirelessness of the network must be simulated or emulated. By choosing to emulate the network instead of simulating it, we get the opportunity to use real nodes that can run real user applications, causing real traffic to flow over the emulated network. Furthermore, we can emulate realistic wireless connections through the use of real terrain data and verified wave propagation models.

The test bed presented makes it possible to graphically display important characteristics, such as the data rate that can be used on a link (i.e. connection) or the traffic load on different links. The flow of overhead traffic, e.g. routing or medium access control (MAC) packets, can also be monitored. Moreover, the test bed makes it possible to run wireless network protocols on the real nodes. Since the test bed is independent of the physical nodes, these types of tests can be done for any type of operating system. Another possible use is to test different applications and study how they behave in an ad hoc network with low capacity links.

The test bed may also be used to test traffic analyzers on military applications and investigate, for example, if they can identify nodes or find network bottle-necks that are especially sensitive to attack. This type of information can then be used to determine strategies for the deployment of electronic intelligence (ELINT) or communications intelligence (COMINT), jamming, CNO, or more conventional units.

When considering EW operations, it is always interesting to be able to test jamming waveforms or strategies without actually airing the signals. By using a tool such as our test bed, it is possible to maintain military and corporate secrecy, while still being able to study the effects of jamming on different types of networks. Furthermore, it is possible to study whether (and in that case how) a change of network protocols affects a network's robustness against jamming or its chances of remaining undetected if the enemy has access to equipment for signal detection or direction finding (DF).

When it comes to CNO, the test bed environment is mainly focused on finding weaknesses in network topologies, routing protocols, or other network-dependent applications. Both as an attacker and as a defender it is interesting to find the nodes most sensitive to attacks. For example, it is possible to perform denial of service attacks on certain nodes in the network. This can be accomplished through the use of some type of malicious code such as viruses, worms or trojans.

## II. RELATED WORK

The idea of simulating or emulating networks is not new, but the level of simulation or emulation differs. The focus of the simulator/emulator also differs greatly from wireless link issues to network protocols, user application effects or network planning. We will discuss a few well-known network emulators. There is, however, a long list of other available emulators and simulators, including some commercially available ones such as Simena [1] or OPNET [2].

The NIST Net Linux-based Network Emulation Tool [3]

E. Johansson is with the Swedish Defence Research Agency, Linköping, Sweden (phone: +46 13 378091; fax: +46 13 378511; e-mail: erijo@foi.se).
M. Persson is with the Swedish Defence Research Agency, Linköping, Sweden (e-mail: matpe@foi.se).

emulates an entire network in a single computer. It allows control of many network parameters such as data rate, latency and bit error rates. Some of these are set manually, and some are generated by statistical routines. The emulated networks are stationary, wired networks. No graphical overview of the network is available.

Furthermore, NIST Net requires one network card inside the emulator for each node or subnet that you want to include in your emulation. NIST Net uses a Linux kernel to control the behavior of the packets through a kernel hook in the network stack. This is done by replacing the original incoming Internet protocol (IP) packet handler with its own, and putting the old handler back when the hook is removed. This means that if NIST Net crashes, the computer will lose the original handler, and the kernel will thus lose its ability to handle IP packets. Moreover, the temporary handler limits the rate of the incoming packets on the interface. If this incoming traffic is high, large queues and a large amount of processor power is needed. This need would be reduced if NIST Net limited the outgoing traffic instead, as there is less traffic going out of the computer than coming in.

The Mobile Network Emulator (MNE) [4] is a test environment that provides flexible and dynamic topology control. It is intended for the testing of internet protocols in mobile, wireless networks. In MNE, a small control program is installed in each node. The topology of the network is modified by sending commands to the nodes to block and unblock connections, based on the distance between the nodes. This results in connections that are either good or non-existent. The blocking of the connections is done with the *iptables* utility that is normally used for firewalling and the resulting connections are displayed on a map. It is not possible to limit the bandwidth or change the latencies. It is also recommended that the network control traffic, e.g. MAC protocol packets or position information, be transmitted as multi-cast on a separate channel. In that case, such information is unaffected by events (congestion, packet loss, jamming, etc.) in the network.

The Mendosus project [5] has a test bed that emulates a local area network (LAN) and which can inject faults into the network. This procedure is similar to the results of jamming or denial of service attacks. The networks are, so far, stationary and wired, although nodes may appear or disappear.

The Network Emulation Testbed (NET) [6] is designed for testing routing protocols on large networks. It can handle a large number of virtual nodes connected in a LAN (each physical PC can handle approximately 20 virtual nodes). Furthermore, NET can enforce bandwidth limitations, delay and frame losses.

The main advantage of our approach compared with the emulators mentioned above is that it enables realistic wireless connections to be emulated (based on real geographical information) and at the same time allows real applications on real nodes to be run over the network using real network protocols. The positions and connections of the nodes are displayed on a map of the terrain area used. The map is also used for displaying results.

Basically, in this paper we combine the areas of EW, network emulation and CNO.

## III. Test bed Description

The test bed contains four main components: an ad hoc network emulator, a computer network emulator, a virtual local area network (VLAN) switch, and a number of nodes that form the actual network. An overview of the test bed system can be seen in Fig. 1.

The Windows® PC runs the ad hoc radio network emulator (AHRNE), i.e. performs the communications and EW calculations necessary to determine the link quality. This PC also shows the nodes' virtual positions in the terrain on a map, along with a graphical representation of some of the test bed results. The Linux PC runs the computer network emulator (CNE), i.e. it uses a daemon to control the physical links between the nodes. At the moment, it regulates the connectivity and (if a connection exists) the available data rate. The switch applies VLANs to separate the connections (links) to and from the ordinary computers that form the network nodes.

The remainder of this section will describe the test bed components in greater detail, starting with the individual nodes and working upwards to the ad hoc network emulator.

### A. *Network Nodes*

The computers (a k a nodes) using the emulated network are connected to the switch with normal twisted pair cables. They use standard communication protocols such as transmission control protocol (TCP), user datagram protocol (UDP), IP, and the Ethernet protocol. They should also use either static routing or dynamic routing to discover their connections. The node can use any operating system, any network protocol, and run any applications, resulting in any type of network traffic.

The equipment that forms the node is stationary, e.g. in a laboratory. However, the ad hoc network emulator allows the node to virtually be moved around in a real terrain area and to experience the same conditions as it would in that environment.

### B. *VLAN Switch*

The switch used, HP Procurve 2524 [7], can use VLANs according to the IEEE 801.1Q standard [8]. With this standard, each packet sent through the switch is tagged with a VLAN identity and only certain ports on the switch can receive packets tagged with this identity. Thus, it is possible to have several separate VLANs inside the switch. The configuration of the switch and these networks is done via the switch's web interface.

In the switch, packets are only tagged when they pass through the port that is connected to the computer network emulator, i.e. to the Linux PC that controls the maximum data rate for each link (connection). Each connected computer uses its own separate VLAN, which makes it impossible for a computer to listen to other connections. If a computer is the intended destination, it receives the packets via the controlling computer network emulator.

Duplicate MAC addresses on different VLANs are not supported by the HP switch used. We have solved this problem by letting the CNE translate MAC addresses back and forth. Another solution would be to acquire a switch that automatically solves this type of problem.

### C. *Computer Network Emulator*

All the traffic in the emulated net goes through the CNE, consisting of a Linux PC. Each incoming packet has been tagged with a VLAN identity by the switch. The Linux

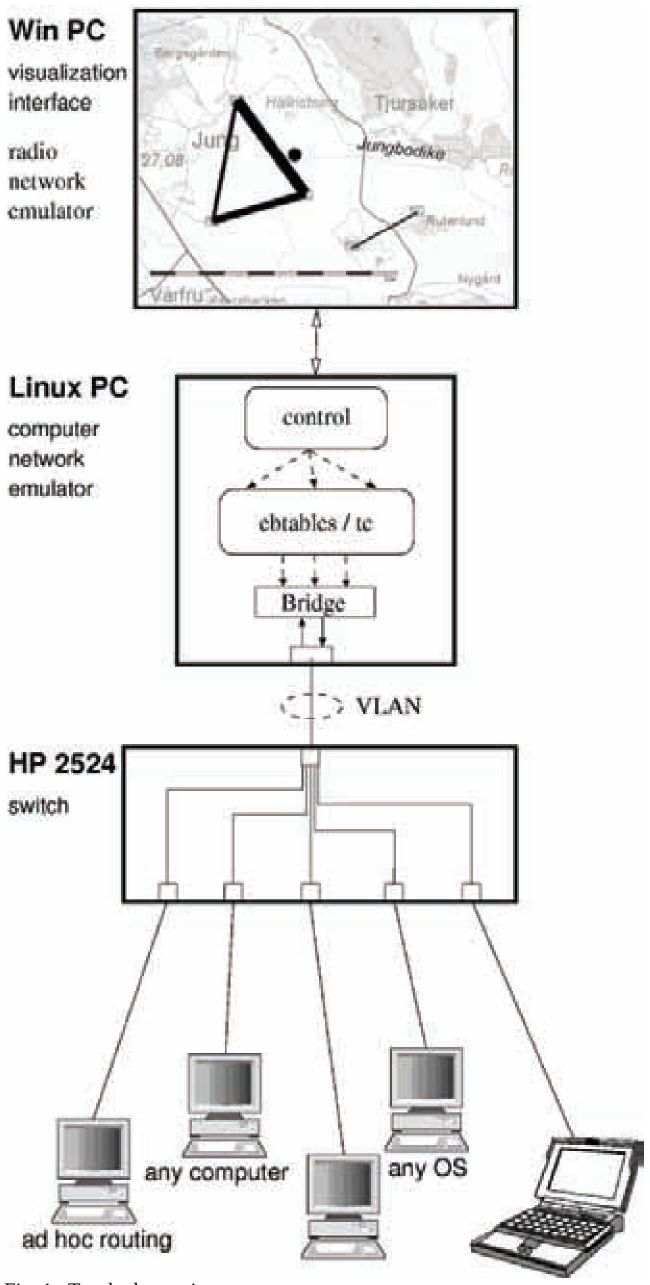


Fig. 1. Test bed overview.

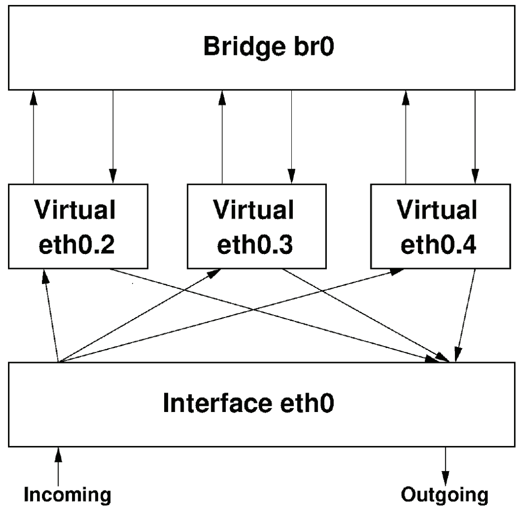


Fig.2. Enlargement of the bridge and the interfaces.

network drivers can read this identity and thus place a packet in the right virtual interface. This means that there is one Ethernet interface for each node in the network. All the interfaces are connected via the bridging functions (IEEE 802.1d standard [9]) in the Linux kernel. See Fig. 2 for an example in which the Linux PC's interface is connected to the bridge through three virtual interfaces. Essentially, the topology of the emulated net is constructed inside the virtual interfaces.

The *ebtables* [10] utility is a firewalling tool used for Ethernet frame filtering on a Linux bridge. Here we use *ebtables* to drop packets when a pair of nodes, which lacks connection (direct or indirect), still tries to exchange traffic. This is accomplished by matching a certain source MAC address to a certain outgoing virtual interface. The *ebtables* are also used for marking the packets with a certain flow identity (Flowid) for further use later in the queuing process.

In order to control the various data rates used between the different nodes, the traffic shaping utility, *tc* [11], is used. This utility is used to configure the traffic control functionality in the Linux kernel. The traffic control can shape, schedule, police and drop traffic. Using *tc* does not actually limit the data rate on a link, but rather modifies the queuing functions in the kernel. Normally, a simple first-in-first-out (FIFO) queue handles the transmission of network packets. Here, we use hierarchical token bucket [12] queuing.

In addition, *tc* defines a hierarchy of queues and for each subqueue sets the maximum data rate for the outgoing packets. Thus, it is possible to set limits for the connection between each pair of nodes in the emulated network.

An example of a packet's way from one node to another:

1. the packet is generated in node 1
2. it arrives at the switch where it is tagged with a VLAN identityit enters the incoming interface in the CNE
3. it is placed in the virtual incoming interface corresponding to node 1

4. it is marked by *ebtables* with the Flowid 1
5. it moves through the bridge
6. it is put in the destination's virtual interface
7. it goes into a queue and then into a certain subqueue depending on the Flowid
8. it is then held back (i.e. forced to wait) to avoid exceeding the maximum data rate
9. it enters the outgoing interface of the CNE
10. it arrives at the switch
11. it is sent out on the appropriate VLAN
12. it arrives to the destination node.

The glue between the CNE and the AHRNE is a control program written in Perl. This program communicates with AHRNE via a socket. The control program translates the data rates received from AHRNE to the corresponding calls to *tc* and *ebtables*. Also, the control program sets up the bridge and the virtual interfaces when the system starts.

## *D. Information Transfer*

It is the calculation results from the AHRNE that are used by *ebtables* and *tc* to determine what data rate a link may use and which nodes are connected. The AHRNE connects to a certain port on the CNE, and control information is subsequently transferred by use of a socket and TCP to the CNE.

At the moment, the control information consists of three integers for each link. These integers represent the transmitting node, the receiving node, and the maximum data rate possible on that link. Each time the calculations are updated, control information is transmitted.

In the same manner, the CNE periodically transmits information to the AHRNE regarding the measured, momentary traffic load on each link. The same syntax is used as for the transmissions from the AHRNE. Each time the AHRNE receives new information, the network(s) is/are redrawn to show the network use.

## *E. Ad Hoc Radio Network Emulator*

The AHRNE runs on a Windows PC and allows us to choose geographical locations for a number of nodes by placing them on a map. The nodes can be moved during the course of the simulation. The propagation of the wireless signal between each pair of nodes can then be calculated based on real terrain data for the area between the nodes, the used equipment, and the receiver environment. A node's environment may, apart from noise, also include effects of jamming.

Based on the results of the wave propagation calculations, a number of networking parameters can be determined. An example of such a parameter is the signal-to-noise ratio (SNR), i.e. how strong the signal of interest is in the receiver compared with the level of noise. Another parameter of importance in our tests is a link's maximum data rate. The network parameters of choice are transferred to the CNE.

### *1) Radio Node Model*

As an example of readily available wireless communication equipment, we use a wireless local area network (WLAN) card. The first card chosen to be implemented in the test bed is Orinoco PCMCIA Silver/Gold [13, 14], see Table I for brief specifications. The abbreviations used are: direct sequence spread spectrum (DSSS), wired equivalent privacy (WEP), and carrier sense multiple access with collision avoidance (CSMA/CA).

TABLE I
PARAMETERS FOR THE ORINOCO SILVER/GOLD CARD

| | |
|---|---|
| Frequency | 2400 – 2484 MHz |
| EIRP | 85 mW |
| Data rates | 11, 5.5, 2, 1 Mbps |
| Channel bandwidth | 22 MHz |
| Modulation | DSSS |
| MAC Protocol | IEEE 802.11 b, CSMA/CA |
| Security | 152/128/64-bit WEP |

In our model, the output power may be varied, although this is not in the card specifications. The reason is that the allowed efficient isotropic radiated power (EIRP) varies between different countries, e.g. in the European Union, the maximum EIRP for a WLAN is limited to 100 mW. A node can have an isotropic antenna with 0 dB gain (antenna amplification) in all directions or a directional antenna. The directional antenna is modeled based on D-link ANT24-1400 [15] and has a maximum gain of 14 dBi (lobe width ± 30$^\circ$). Other antennas can easily be modeled and added to the test bed.

The chosen WLAN card supports IEEE 802.11b [16 - 19] and 802.11g. Here we look at 802.11b. Using this protocol, four data rates are available to the user: 11 Mbps, 5.5 Mbps, 2 Mbps, and 1 Mbps. The higher the data rate is on a link, the better the signal in the receiver needs to be. A common way to determine the quality of the signal is to measure the SNR in the receiver. In an environment where jamming or intersystem interference is present, the signal-to-jamming-ratio (SJR) or signal-to-interference-ratio (SIR) is measured instead of the SNR. Here, the noise is included in the jamming or interference.

In the WLAN node model, we use the SNR (or, in the event of jamming, SJR) in the receiver to determine the maximum data rate for a transmission between each pair of nodes. The relationship between SNR (SJR) and the usable data rate is shown in Table II for the chosen WLAN card. If the maximum (or minimum) SNR is reached, a higher (lower) data rate is automatically used.

TABLE II
RELATIONSHIP DATA RATE — SNR (OR SJR)

| Data rate | SNR (or SJR) [dB] | |
|---|---|---|
| [Mbps] | Min. | Max. |
| 11 | 16 | - |
| 5.5 | 11 | 16 |
| 2 | 7 | 11 |
| 1 | 4 | 7 |
| No link | - | 4 |

In a more detailed model, other aspects such as receiver sensitivity and transmission delays should be taken into account. Furthermore, it should be noted that, at the present time, the effects of intersystem interference are not taken into account when determining whether communication can take place and at what data rate, i.e. SIR is not calculated or used. This means that we assume that no simultaneous transmissions are made on overlapping channels within interfering distance.

As mentioned in Section III A, we use the Ethernet protocol (i.e. IEEE 802.3) [20] as a MAC protocol on the computer network. There are certain differences between this protocol and IEEE 802.11b. We have modeled the varying data rates of IEEE 802.11b and their relationship to the transmission conditions by letting the SNR (or SJR) determine which data rates can be used. However, in IEEE 802.11b CSMA/CA is used, while Ethernet uses CSMA with optional collision detection (CD). The differences between using CA and CD have not been dealt with yet and may influence the result in some situations. Furthermore, since the interference is not taken into account when calculating SNR or SJR, the capacity sharing that takes place among the nodes in real life is not fully modeled. Thus in some situations the emulated network may have higher capacity than the corresponding real network.

*2) Jammer Model*

The jamming is modeled as additive white Gaussian noise (AWGN) within a chosen frequency band. The width of this band may be freely chosen within the WLAN frequency band. The output power of the jammer may also be chosen freely. It must, however, be noted that for a unit whose only power supply is batteries, an output power of approximately 1 mW – 10 W is realistic. The jammer has two antennas available, the same ones as for the WLAN node. It is, of course, possible to use a more sophisticated antenna instead.

AWGN jamming is a good choice when little is known of the system you are attempting to jam. If more information is available, there are often more intelligent jamming methods. For example, it has been shown that if pulses are used instead of continuous jamming, major savings in output power are possible (up to 10, 000 times less power consumption) [21]. This is due to the fact that IEEE 802.11 and IEEE 802.11b have not been designed for dealing with intentional interferences (i.e. jamming).

*3) Wave Propagation and Terrain Data*

To calculate the SNR or SJR for a certain transmitter-receiver combination, the path loss must first be calculated. The path loss represents the reduction of signal strength that results from obstacles encountered by the signal on its way to the destination. Only obstacles that are included in the terrain data (e.g. hills, rivers, forests) can be taken into account when calculating the path loss. It is thus important to have detailed, and up-to-date geographical information available. Here we, use terrain data from The National Land Survey of Sweden (Lantmäteriverket) that has a resolution of 50 meters and a height resolution of 1 meter.

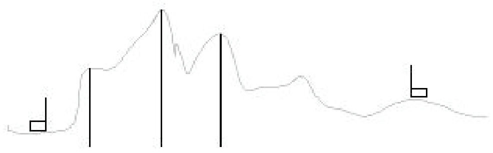


Fig. 3. Example of a terrain profile with three knife-edges.

We calculate the path loss using a model from the wave propagation library Detvag-90 [22]. Based on the frequency band at hand, we chose a diffraction model by Vogler [23]. This type of model picks a number of dominating heights (a k a knife-edges) along the path profile. Fig. 3 shows a terrain profile between two nodes and how three knife-edges may be chosen from this profile.

The diffraction of the transmitted radio waves over these knife-edges is then calculated to determine the signal strength at the receiver. At the moment, three knife-edges are used.

The knife-edge model takes the transmission frequency, the nodes' positions, the antenna heights, and polarizations into account when calculating the path loss. Furthermore, terrain data are used to determine ground heights and terrain types along the transmission path. Ground constants for conductivity or permittivity are not used in Vogler’s model, but the terrain type is needed to find the terrain heights along the path. The propagation model then uses the combined height of both ground and terrain to determine where to place the knife-edges.

*4) Noise and Loss*

The SNR (or SJR) in a receiver is not only dependent on the path loss, but also on a number of other factors such as the noise and additional losses (compared to the path loss). The noise can be divided into two components: external and internal noise. The total noise in the receiver is the sum of these components.

The internal noise is generated within the receiver and depends on factors such as the cables, the antenna, the receiver temperature, and the reception bandwidth. For example, a receiver with 22 MHz bandwidth (our WLAN card) has an internal noise of at least -130 dB.

The external noise is due to the environment in which the receiver is placed and the frequency used. The noise models used are based on the recommendations of the International Telecommunication Union (ITU) [24, 25]. In the frequency band used, our models for external noise typically increase the noise level by approximately -40 - +10 dB, depending on the environment the receiver is placed in.

Additional losses can occur for a number of reasons. Polarization loss occurs if the transmitter and receiver antennas have different polarization or if the signal, due for example to reflections, has changed its polarization along the way. Losses due to the cables used are also common. These losses, as well as a general system loss, are included in our calculations.

### F. Graphical Result Presentation

A link from node A to node B and the corresponding link from B to A may experience different conditions, e.g. the noise level or the amount of jamming present at node A and node B can differ. The maximum data rate that is possible on the respective links may thus differ. It is difficult to show results for both these links in an easily comprehensive manner, especially when there are many nodes and possibly several networks in a small geographical area.

Hence, at the moment, results are limited to showing links in one direction. We choose to show the link with the heaviest traffic load (measured as a percentage of the maximum data rate possible). If the traffic load is equal, the link with the lowest maximum possible data rate is displayed.

In our graphical representation of the results, we use the thickness of a line (i.e. link) to portray the maximum possible data rate and the color to show the link's traffic load. The link colors are basically the colors of the rainbow, where dark blue is a traffic load of 0-10 % (of the maximum data rate the link can maintain), and red represents a load of 91-100 %. The map, with its network overview, can thus yield link information at a glance.

An example of the graphical representation can be found in Fig. 4. Here, a network with five nodes is depicted and the following notation is used: *1→0: 3.0 (11)* means that *Node_1* is transmitting 3.0 Mbps to *Node_0* and that the link has a maximum data rate of 11 Mbps (— is used for links without a traffic load).

No link information is discarded, and detailed information regarding both link directions can also be presented for the user. See Fig. 5 for an example.

## IV. Experimental Results

The implementation of the test bed has recently been finished and has thus yet to be used for many of the purposes suggested in Section I. We will, however, mention two short examples of initial observations, along with a description of the limitations made.

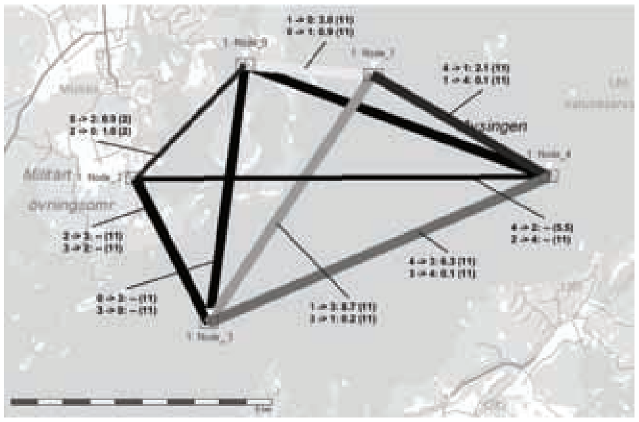
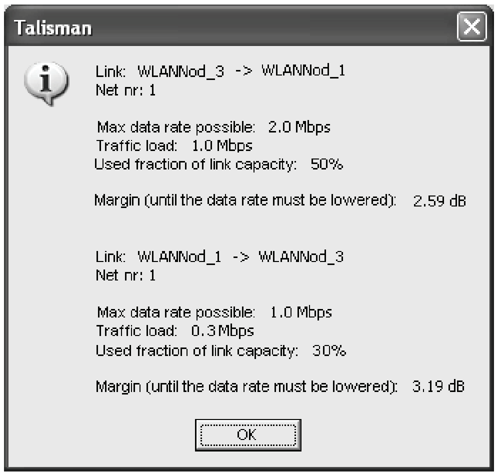


Fig. 4. A WLAN network with five nodes.


Talisman

Link: WLANNod_3 -> WLANNod_1
Net nr: 1

Max data rate possible: 2.0 Mbps
Traffic load: 1.0 Mbps
Used fraction of link capacity: 50%

Margin (until the data rate must be lowered): 2.59 dB

Link: WLANNod_1 -> WLANNod_3
Net nr: 1

Max data rate possible: 1.0 Mbps
Traffic load: 0.3 Mbps
Used fraction of link capacity: 30%

Margin (until the data rate must be lowered): 3.19 dB

OK

Fig. 5. An example of link information that can be displayed.

### A. Limitations

The level of detail used in the test bed for both modeling and calculations can be varied, as well as what parameters are exchanged between the ad hoc network and the computer network emulators. The results chosen for graphical presentation can also be altered.

As this paper focuses on the methodology of building the test bed, uncomplicated node and jammer models are used. The effects of unintentional electromagnetic interference, for example, are excluded. The results of the calculations are so far only used to control the maximum data rate for each link. Furthermore, the graphical result presentation is limited to the maximum data rate and the used capacity of each link.

The limits of the test bed regarding maximum number of nodes or maximum traffic load have not yet been explored. The maximum number of VLANs, i.e. the maximum number of nodes, in the switch is 29. The maximum amount of traffic that the CNE can handle has yet to been determined. However, as it is easy to find Ethernet network cards with higher speeds than the available WLAN cards, this does not limit the test bed as long as Ethernet is used on the nodes. There is also a limit to the number of nodes (999) that the ad hoc network emulator can handle. The graphical interface, however, will start to experience difficulties at an earlier stage due to the time needed for redrawing.

In the tests presented here, five nodes are used. At the moment, each node is a PC (with a Linux SLAX distribution [26]) and runs the optimized link state routing (OLSR) protocol [27] as implemented by [28]. This protocol can be exchanged for any other routing protocol. The user/application traffic used so far consists of data packets (of different sizes) and is generated and sent by a script via TCP connections. OLSR control traffic is broadcasted in the network, i.e. sent on the same channel as the user traffic to all nodes within range.

The test bed emulates the data rates and some of the behavior of the wireless MAC protocol IEEE 802.11b. The

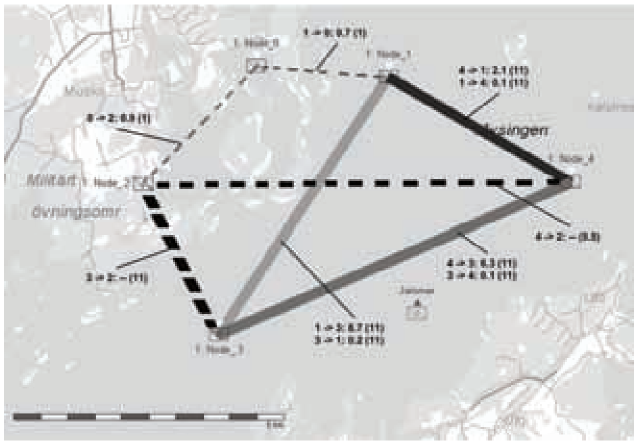


Fig. 6.  The results of adding a jammer to the scenario of Fig. 4.

actual nodes in the network, however, run an adapted version of the Ethernet protocol, see Section III A and E for more information.

### B. *A Basic Network Example*

As soon as a number of nodes (computers) are connected, we can see traffic starting to flow in the network. A small amount of traffic can be seen propagating through the network regardless of whether any users or applications have anything to transmit. This traffic is the network overhead, generated by the network protocols.

### C. *An Example on the Effects of Jamming*

If a jammer is added to a scenario, what will happen to the network links is not always intuitive. We added a jammer to the scenario in Fig. 4 and the result is displayed in Fig. 6. The dashed lines in the figure represent one-way links.

We can see here that links relatively far from the jammer are those that are the most affected by the jamming. The reason for this is that these links had SNRs close to the minimum needed for a 2 Mbps transfer even before the jamming. When the jamming started, they had no margins left to deal with this additional interference and were forced to change the maximum data rate to 1 Mbps. The high capacity links close to the jammer experience a dramatic drop in SNR, but because they had large margins, they managed to maintain a rate of 11 Mbps.

## V. Future Work

The wave propagation and noise models used are verified. In the process of building the test bed, additional modeling has been done. Certain assumptions and simplifications, e.g. concerning the MAC layer, have also been made. It would be of great interest to perform a series of tests to determine whether these factors, i.e. models, assumptions and simplifications, have any impact on the test bed results. To determine how big the impact is, these experiments must compare the test bed with real field trials for different scenarios. The test results will then show where improved modeling will be of the greatest use.

Regarding the ad hoc radio network emulation and the radio channel model, we already know that we would like to make certain improvements. Delays should be added, as well as bit error rates and bit error patterns. Furthermore, the sharing of network capacity necessary to avoid interferences in a wireless network has not been fully modeled. To achieve a more realistic behavior, either a more advanced model of a wireless MAC protocol should be made or a real wireless MAC protocol should be used.

By using a more advanced switch, we can improve the handling of duplicate MAC addresses in different VLANs. A more advanced switch can probably also handle a larger number of VLANs simultaneously.

A more important improvement would be to make a special kernel module for network queue handling in Linux. This module would replace the FIFO queue (see Section III C) with a more advanced queue. A design goal would be to get better rate balancing between the outgoing interfaces. It should also introduce bit errors in certain patterns, drop random packets, delay packets and move packets around according to specified instructions.

At the moment, it is possible to manually move the nodes in the mapped terrain. To improve the realism of the emulation, it would be of interest to have more mobile nodes. This can either be done through some mobility model or by associating each node with a path or at least a speed and a direction.

From an EW point of view, the implementation of detection/direction-finding models would also be of interest. This would enable studies of how the routing and MAC protocols used, or the type of traffic, affects the discovery and direction-finding of nodes in the network. By analyzing the transmissions or the traffic, it is also possible to determine which type of unit a node is and to use this information to determine what measures (e.g. jamming, monitoring, or CNO) should be taken.

When adding more functionality to the test bed, the graphical representation of the results must also be augmented. Delays, bit error rates, the spreading of malicious code, and coverage area within which direction-finding is possible are all examples of results that you might want to present graphically.

## VI. Conclusions

We have made a versatile test bed where real applications can exchange real data over a computer network and where the connections are subjected to wireless phenomenons such as path loss and jamming. The test bed can be used not only for the assessment of CNO, although this was the main objective, but also when studying network protocols or EW against different types of networks. Furthermore, the test bed gives opportunities to further look at ways of combining EW and CNO to achieve the best disruptive effect. Our initial results show that, although the present models are simple, interesting results can be achieved and improve the understanding between the involved disciplines, i.e. ad hoc networking, EW, and CNO. These multidisciplinary studies are the great advantage of the approach used.

## REFERENCES

[1] Simena – Network Emulator, http://www.simena.net/

[2] Operational Network Evaluation Tool (OPNET), http://www.opnet.com

[3] M. Carson, and D. Santay, "Tools: NIST Net: a Linux-based network emulation tool," *ACM SIGCOMM Computer Communication Review*, Volume 33, Issue 3 , July 2003, pp 111-126.

[4] J. P. Macker, W. Chao, and J. W. Weston, "A low-cost, IP-based mobile network emulator (MNE)," in *Proc. 1st IEEE Military Communications Conference (MILCOM)*, Boston, USA, Oct 2003, pp. 481-486.

[5] X. Li, R. Martin, K. Nagaraja, T. D. Nguyen, and B. Zhang, "Mendosus: A SAN-Based Fault-Injection Test-Bed for the Construction of Highly Available Network Services," in *Proc. 1st Workshop on Novel Uses of System Area Networks (SAN-1)*, Cambridge, MA, USA, Feb 2002.

[6] S. Maier, D. Herrscher, and K. Rothermel, "On Node Virtualization for Scalable Network Emulation," in *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS '05)*, Philadelphia, PA, USA, July 2004, pp. 917–928.

[7] Manufacturers specification, HP Procurve 2524 Switch, http://www.hp.com/rnd/products/switches/switch2524-2512/overview.htm

[8] IEEE 802.1Q, IEEE Standard for Virtual Bridged Local Area Networks, http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf

[9] IEEE 802.1D, IEEE Standard for Local and Metropolitan Area Networks — Media Access Control (MAC) Bridges, http://standards.ieee.org/getieee802/download/802.1D-2004.pdf

[10] B. De Schuymer, "Ethernet Bridge Frame Table Administration (ebtables)," http://ebtables.sourceforge.net/documentation.html

[11] A. N. Kuznetsov, "Show / Manipulate Traffic Control Settings, tc(8)", Linux man page, http://www.die.net/doc/linux/man/man8/tc.8.html

[12] M. Devera, "HTB - Hierarchical Token Bucket," http://luxik.cdi.cz/~devik/qos/htb/

[13] Manufacturers specification, WLAN card, ORiNOCO 11b/g PC Card, http://www.tribecaexpress.com/Proxim_Orinoco_pc_cards.htm

[14] M. Coinchon, "Radio theory and link planning for Wireless LAN (WLAN)," http://www.swisswireless.org/wlan_calc_en.html

[15] Manufacturers specification, WLAN antenna, D-Link ANT24-1400, ftp://ftp10.dlink.com/pdfs/products/ANT24-1400/ANT24-1400_ds.pdf

[16] IEEE Std 802.11, IEEE Standards for Local and Metropolitan Area Network — Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[17] IEEE 802.11b, Supplement to 802.11-1999,Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band, http://standards.ieee.org/getieee802/download/802.11b-1999.pdf

[18] IEEE 802.11b, Corrigendum1 to IEEE 802.11b, http://standards.ieee.org/getieee802/download/802.11b-1999_Cor1-2001.pdf

[19] VOCAL Technologies, Datasheet — 80211b CCK Wireless LAN up to 11 Mbps, http://www.vocal.com/data_sheets/ieee80211b.html

[20] IEEE 802.3, IEEE 802.3 LAN/MAN CSMA/CD Access Method, http://standards.ieee.org/getieee802/download/802.3-2002.pdf

[21] G. Noubir, and G. Lin, "Low-Power DoS Attacks in Data Wireless LANs and Countermeasures," in *Proc. MobiHoc 2003*, Annapolis, USA, http://www.sigmobile.org/mobihoc/2003/posters/p223-noubir.pdf

[22] B. Asp, G. Eriksson, and P. Holm, "Detvag-90 – Final Report," FOA (FOI), Linköping, 1997, FOA--R-97-00566-504—SE

[23] L. E. Vogler, "An Attenuation Function for Multiple Knife-Edge Diffraction," Radio Science, vol.17, no.6, pp. 1541-1546, 1982

[24] International Telecommunication Union Recommendation, "HF Propagation Prediction Method", ITU-R P.533-5

[25] International Telecommunication Union Recommendation, "Radio Noise", ITU-R. P.372-7

[26] Linux SLAX distribution, http://slax.linux-live.org/

[27] T. Clausen (Ed.), and P. Jacquet (Ed.), "Optimized Link State Routing (OLSR)," IETF Request for Comments, October 2003, www.ietf.org/rfc/rfc3626.txt?number=3626

[28] OLSR algorithm implementation available at http://www.olsr.org/

# E Oscar – ACSF 2006

Martin Karresand och Nahid Shahmehri. Oscar – File Type and Camera Identification Using the Structure of Binary Data Fragments. *Proceedings of the 1st Conference on Advances in Computer Security and Forensics, ACSF*, ss 11–20, 2006. The School of Computing and Mathematical Sciences, John Moores University.

# Oscar – File Type and Camera Identification Using the Structure of Binary Data Fragments

Martin Karresand[1,2], Nahid Shahmehri[1]

[1] Dept. of Computer and Information Science
Linköpings universitet,
Linköping, Sweden
[2] Dept. of Systems Development and IT-security
Swedish Defence Research Agency,
Linköping, Sweden
{g-makar,nahsh}@ida.liu.se

**Abstract** Mapping out the contents of fragmented storage media is hard if the file system has been corrupted, especially as the current forensic tools rely on meta information to do their job. If it were possible to find all fragments belonging to a certain file type, it might also be possible to recover a lost file. The Oscar method identifies the file type of data fragments based on their structure. This paper presents an improvement of the Oscar method. The new version is built on using 2-grams to create a model of different file types. The method is evaluated for JPEG, Windows executables, and zip files, reaching a 100% detection rate with 0.12% false positives for JPEG. We also use the method to identify the camera make used to capture a JPEG picture from a fragment of the picture.

## 1 Introduction

Being able to survey the contents of a hard disk is vital in, for example, a forensic examination or during recovery of lost data. Unfortunately fragmentation together with a corrupt file system complicates the mapping out of the hard disk. Depending on the level of fragmentation the task can be anything from hard to almost impossible. To further complicate the situation, there are, to the best of our knowledge, no tools [1,2,3,4,5,6,7] available that can identify the file type of a data fragment without relying on meta data in some form. Due to this lack of adequate tools a forensic examiner can miss fragments of JPEG pictures containing child pornography on a hard disk or other storage media.

If all fragments of a certain file type can be found, the fragments can be used to recreate the original files. A method [8] to do the latter exists, but this work aims to provide a solution to the first part.

To be able to determine the file type of a data fragment without using meta data we have to look at the structure and contents of the fragment. For text-based data this is trivial, but for binary data we cannot rely on finding readable strings to use. The structure of the data itself is, however, unique for different types of files, which can be seen in Fig. 1 showing histograms of the byte frequency distribution of a typical .exe, .zip, JPEG, and .ps file.

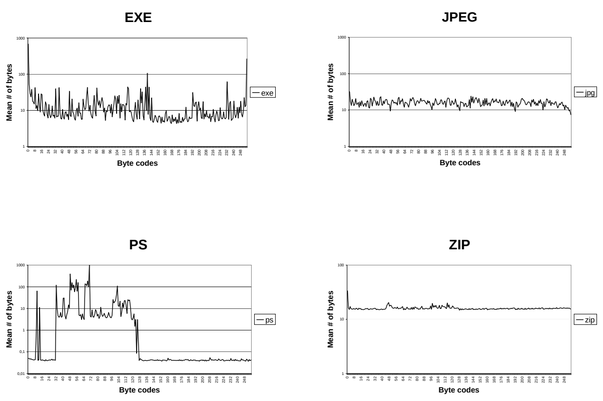


**Figure 1.** Histograms calculated over a number of 4 kB blocks of four different file types, .exe, JPEG, .ps, and .zip. Only the data part of the files was used when creating the JPEG histogram. A logarithmic scale is used for the Y-axis.

As the size of storage media is rapidly increasing, and probably will for the foreseeable future, the execution time of the tools used for forensic examinations is of importance. There are no real-time execution requirements connected to the forensic process, but a criminal suspect cannot be held in custody indefinitely, hence the forensic examiner still has time limits to adhere to.

This work is a further development of the Oscar method [9,10]. The goal is to find a method to correctly identify the file type of arbitrary data fragments that is fast enough to be used as the main tool for forensic examinations of current and future storage media. The original Oscar method uses 1-grams, i.e. single bytes, to create a histogram of the byte frequency distribution (BFD) of a data block. The mean and standard deviation of each byte frequency is used to form a model, called centroid, of the byte frequency distribution. The similarity of an unknown sample and the centroid is measured by calculating the difference between each byte value frequency of the sample and the centroid, using a sum of squares weighted by the standard deviation of each byte frequency.

The Oscar method has been extended by including the principle of rate of change (RoC) [10], which measures the absolute value of the difference between consecutive bytes. This approach allows the smoothness of the byte stream of a data block to be taken into consideration, decreasing the number of false positives from data blocks that are of a different file type from the centroid, but which have a similar byte frequency histogram.

The version of the Oscar method presented in this paper uses byte pairs, 2-grams, for the model. We have implemented the method as a JPEG detector, and also tested the 2-gram extension for Windows executables and .zip files.

During the tests of the 2-gram method we got results indicating the method is able to tell some digital camera makes apart. We therefore extended our experiments to also include a test of that ability. The fact that cameras of different makes code picture data in slightly different ways, which shows in the JPEG stream, made us want to test the possibility to recognise camera makes. Related work in camera recognition has been done for complete pictures [11,12,13,14], but not for picture fragments.

Due to the above-mentioned reasons this paper addresses the following questions:

– What detection rate can be achieved when using 2-grams for file type identification of data fragments?
– Is it possible to use 2-grams to identify the camera make used to capture a picture from a fragment of the picture file?

## 2 Method

The new idea presented in this paper is based on the use of 2-grams instead of 1-grams as in the original Oscar method. There are both advantages and disadvantages to using 2-grams. The disadvantage of using 2-grams is the exponential increase of the centroid's size. This mainly affects the execution speed of the algorithm, but also the memory footprint. In theory the effect is a 256-fold increase in process time and memory footprint compared to the original 1-gram methods. The increase in memory footprint can be ignored due to the amount of RAM used in modern computers; the requirement of the current implementation is a few hundred kB of RAM. By optimising the algorithm the processing time can be decreased, but the decrease is not bounded.

The main advantage is the automatic inclusion of the order of the bytes into the method, signalling that an unknown block does not conform to the specification of a specific file type and thus should not be categorised as such. A typical example is the appearance of certain JPEG header byte pairs disallowed in the data part. Another example is special byte pairs never occurring in files created by a specific camera or software.

### 2.1 File type identification

The centroid modelling a file type is created by counting the number of unique 2-grams in a large number of 1 MB blocks of data of a specific file type. The data in each 1 MB block is used to form a frequency distribution of all 65536 different possible 2-grams and the mean and standard deviation of each 2-gram is calculated and stored in two $256 \star 256$ matrices. The values are then scaled down to correspond to a block size of 4 kB.

The 4 kB size is used because it is equal to the common size of memory pages and disc clusters in current home computers. The reason for using 1 MB data blocks is to get a solid foundation for the mean and standard deviation calculations. When using a block size less than the possible number of unique 2-grams the calculations inevitably become unstable.

When the data type of a 4 kB sample data block is to be identified, the similarity between the sample and a centroid is measured by using a quadratic distance metric. Equation 1 describes the metric, where matrix $S$ depicts the sample 2-gram frequency distribution and $C$ the mean value matrix of the centroid. The standard deviation of 2-gram $ij$ is represented by $\sigma_{ij}$. There is also a smoothing factor $\alpha = 0.000001$, which is used when $\sigma_{ij} = 0$ to avoid division by zero.

$$\mathrm{d}\left(S, C\right) = \sum_{i=0, j=0} \left(s_{ij} - c_{ij}\right)^2 / \left(\sigma_{ij} + \alpha\right). \tag{1}$$

The quadratic term is used to favour smaller deviations over larger. As can be seen in Eq. 1 the differences are weighted by the standard deviation of each 2-gram. This is done to lessen the influence of 2-grams that vary significantly and in that way only use the key features of the 2-gram distribution.

The distance value of a sample is compared to a empirically predetermined threshold. Samples giving distance values below the threshold are categorised as being of the same file type as the centroid.

The reason for not using a more advanced distance metric is execution speed. The current maximum size of a hard disk is 750 GB and hence a forensic examination can involve a tremendous amount of data to scan.

### 2.2 Camera recognition

To establish the probable camera make of an unknown picture fragment it is compared to a set of centroids created from pictures of different camera makes. Preferably several hundreds of pictures from each camera make should be used. The set of pictures should be subdivided into groups and and the pictures from one group should not be compared to a centroid created from data including that group.

The comparison featuring the Cartesian product of all picture groups and centroids is made using different thresholds and the number of true positives is recorded. These results are then used to build a matrix showing the threshold at which a certain centroid starts detecting pictures of a certain camera make and model.

When the probable camera make of an unknown sample is to be established a new matrix is created and matched to the centroid/picture matrices. The picture fragment is then categorised as captured by a camera of the same make as the best matching matrix represents.

## 3 Evaluation

The 2-gram version of Oscar was tested, both with regard to its ability to correctly categorise the file type of data fragments, and to identify the camera make of JPEG fragments. The different cameras are shown in Table 1.

**Table 1.** The camera make and models used for the experiments

| | |
|---|---|
| Canon DIGITAL IXUS 400 | Canon DIGITAL IXUS v |
| Canon PowerShot A70 | Casio EX-Z40K |
| Fuji FinePix2400Zoom | Fuji FinePix E550 |
| Kodak CLAS Digital Film Scanner/HR200 | Konica KD-310Z |
| Konica Minolta DiMAGE G400 | Konica Minolta DiMAGE X60 |
| Nikon D50 | Nikon E3500 |
| Sony DSC-P8 | |

The pictures are all standard family photographs, both indoor and outdoor sceneries with varying lighting conditions and settings. Some of the pictures are rotated using the camera's built-in editing facility, otherwise the pictures are unedited.

### 3.1 File type identification

The file type identification experiment was performed using a 72 MB test file consisting of 57 files of 51 different types concatenated into one. The files were taken from an office computer running Windows XP, fully patched. The individual files were padded with zeros to adjust their lengths to a multiple of 4 kB.

The padding was added to make the concatenated file resemble a part of a hard disk with 4 kB clusters. The method does not use a sliding window, but instead scans one data block at a time and thus the fact that the files in the test file were continuous, i.e. not fragmented, did not affect the experiment.

The experiment was performed by scanning the test file using a JPEG centroid, a zip file centroid, and a Windows executable centroid. The true positives and false positives were then counted and plotted against each other.

The JPEG centroid was created from 8 MB chunks of the picture files from the 12 cameras and a scanner used in the camera recognition test. The header part of the JPEG files were stripped off before creation of the centroids. The zip file centroid was made from 104 MB of different zip-based files, created using WinZip, or the Linux gzip and zip utilities. We used the complete zip files, i.e. kept the footer part of them, because there are no disallowed byte pairs in the data part or footer of zip files. The Windows executable file centroid was created from a 147 MB large concatenation of different files. Our requirement for inclusion was that all files were reported as either "MS-DOS executable (EXE), OS/2 or MS Windows", or "MS-DOS executable (EXE)" by the Linux *file-4.12* utility. All files begin with the magic number 0x4D5A, "MZ". Also in this case we kept the files as-is, since the header is only one of several different sections making up an executable file.

The template file used as ground truth for the experiment labelled the JPEG header as non-JPEG. The executable and zip files were labelled as true positives in their entirety. The distribution of 4 kB fragments in the test file was 1150 JPEG, 4397 Windows executables, 1426 zip, and 11510 fragments of other types.

### 3.2 Camera recognition

The camera recognition experiment was performed using an 80/20 model, i.e. we created a centroid from 80 % of the picture data for a specific camera and used the remaining 20 % for testing. Each centroid was tested against all different cameras giving us the Cartesian product of the picture/centroid combinations. Consequently, there were five sub-experiments, which were used to calculate a mean value for the detection rate.

Since the amount of data available differed between the cameras, we set the size of each sub-part of the centroid to the size of the smallest part, which was 8 MB, giving us centroids of 32 MB. That size corresponds to approximately 30 pictures.

## 4 Result and Discussion

The results show a very good detection rate for JPEG data fragments, with a low rate of false positives, but poor results for the zip and Windows executable types. Regarding the camera recognition experiment, the Fuji cameras stand out clearly, as well as the scanner. The other cameras give signature matrices that are more similar, which are harder to tell apart.

### 4.1 File type identification

The results of the file type recognition experiment can be seen in Fig. 2.

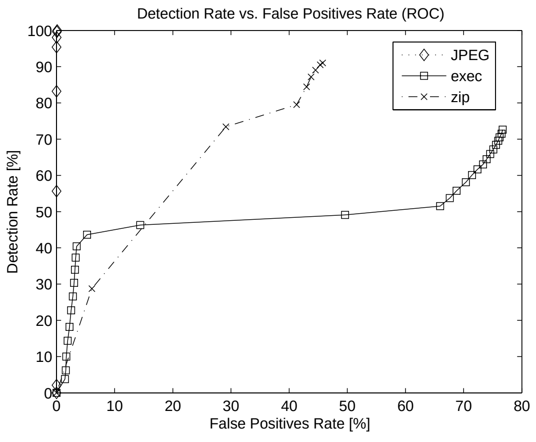


**Figure 2.** The detection rate vs. the false positive rate using JPEG, Windows executable, and zip centroids

The method reaches 100% detection rate for JPEG file fragments with a false positive rate of 0.1% and there are no false positives until the detection rate reaches 98.1%. The distance threshold can be adjusted to change the behaviour of the method, so that it finds every fragment of a certain type, plus a few fragments of other types, or so that it does not find every fragment, but the ones found are all correct, depending on what is desirable.

Since the false positive rate of the zip files and the Windows executable files are almost or more than 50% these cannot be used as they are. We do, however, think that the results can be improved by fitting the selection of data for centroid creation to the standards connected to the file types.

### 4.2 Camera recognition

Fragments of pictures taken by the Fuji camera are clearly distinguishable from fragments of the rest of the cameras, as are the scanner fragments. It is also possible to separate the the Konica Minolta DiMAGE X60 fragments and the Casio fragments from other camera makes, but not with as high a degree of confidence as with the Fuji and scanner fragments. The Konica Minolta (except for DiMAGE X60), Konica, Nikon, Canon, and Sony cameras are all practically indistinguishable.

Figure 3 shows the signature matrices coded as bitmaps for the different camera makes. To save space, only two matrices (Canon and Nikon) are shown for the cameras with almost similar matrices. The centroids, i.e. the individual bars, are placed in the same order from left to right as the camera types shown at the top of the figure. The colour saturation of the bars represents the number of true positives: the darker the bar, the higher the level.

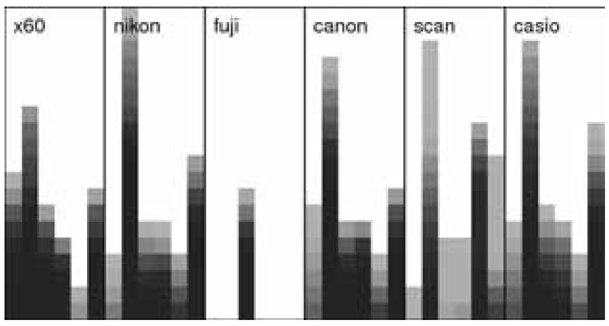


**Figure 3.** A comparison chart for pictures mapped by different centroids. The centroids, i.e. bars, are placed in the same order as the pictures, from left to right.

As can be seen in Fig. 3 the Nikon camera centroid has a better detection rate than the other centroids. The Konica Minolta DiMAGE X60 centroid starts detecting data at about the same level for all pictures, but does only reach a 100% detection rate for its own pictures. The scanned pictures are only detected to 100% by the Nikon centroid and the scanner centroid. The Fuji pictures contain a special marker sequence that the other pictures do not, consequently only the Fuji centroid can detect them. The Nikon and Canon matrices are used to represent the other cameras, which give almost similar results.

### 4.3 Discussion

The detection and false positive rates of the Windows executables centroid are poor to say the least, in fact, they are even worse than guessing. There are, however, a large number of different file formats fitting our requirement that the files should start with the “MZ” byte pair. The structure also varies within the files themselves. Typically there can be sections of readable ASCII characters, well structured binary tables, and sections of machine code, arbitrarily ordered.

The last part of a zip file contains meta data required to decompress the file. This part could be excluded from the creation of the centroid in the same way as the header parts of the JPEG files were stripped off before creation of the centroid. We chose to keep the footer parts of the zip files because these parts did not differ that much from the data parts of the files.

Due to the limited set of cameras used for the experiments we cannot really say anything more than that it is possible to distinguish Fuji camera picture fragments and the scanner fragments from picture fragments of the other cameras included in the experiment. The results point to an interesting area needing further research.

## 5 Related Work

The works presented in this section are related to the Oscar method either by the use of an n-gram algorithm for operation on fragmented data in different ways, or by their ability to identify the camera used to capture a picture without relying on make and model information in the file header.

An n-gram based method that is related to the Oscar method is the intrusion detection system PAYL [15,16] from the research group lead by professor Stolfo at Columbia University, USA. The same group has also used n-grams for file type recognition using fileprints [17,18]. The main differences between these methods and the Oscar method are the previous methods' dependency on file header information, as well as the fact that Stolfo and his coauthors do not consider byte order, which the later versions of the Oscar method does.

Also McDaniel and Heydari [19] use the concept of n-grams in a file type identification method. They use different statistical measures applied to single bytes in files to make finger prints of files. McDaniel and Heydari's main source of data is the header and footer parts of files. They try to consider the ordering of the bytes in files through the use of what they call *byte frequency cross-correlation*. One example of bytes with high byte frequency cross-correlation is the html tag markers “<” and “>”. The McDaniel and Heydari method differs from the Oscar method through the use of single bytes, as well as the their method's dependence on file header and footer data.

Yet another n-gram based method is presented in a paper by Shanmugasundaram and Memon [8]. They use a sliding window algorithm for fragmented file reassembly using n-gram frequency statistics. They let the window pass over the edges of the fragments and then combine those fragments having the highest probability of being consecutive based on the frequency of the n-grams found in the window. Their use of n-grams frequency statistics is similar to the Oscar method, but they apply the method in another way and require all fragments of a file to be known in advance. Therefore the Oscar method is applicable to the steps before Shanmugasundaram and Memon's method and is only partly related to their method. The two methods can, however, be joined together and used to first identify and then reconnect fragments of files.

Camera recognition applied to the pixel level of pictures is presented by Lukáš, Fridrich, and Goljan [12,13,14]. They base their method on the small deviations in uniformity in the CCD of a camera. The deviations are detectable even after compression and other picture manipulation exercises. In the papers they show that the deviations are unique to a specific camera. Another camera recognition method applied to the pixel level of pictures is presented by Kharrazi, Sencar and Memon [11]. They identify 34 features that can be used to identify which camera make and model was used to capture a picture. These two methods differ from the Oscar method and its camera recognition ability in that both methods are applied to unfragmented, complete pictures and the Oscar method works on fragmented data.

## 6 Conclusion and Future Work

We have developed a 2-gram version of the Oscar method, giving a 100% detection rate of JPEG fragments, coupled with a false detection rate of 0.1%. We also used the method to identify the camera make used to capture a picture. Our preliminary results show that Fuji cameras as well as scanners differ significantly from other camera makes and thus can be identified from a picture fragment.

We will continue exploring the abilities of the 2-gram Oscar method. There is still work to be done in developing centroids covering more file types. We also need to further investigate the reason for the poor results of the executable file centroid. One reason can be that our definition of executable files is too general, since the current definition covers a large number of sub-types. We also need to experiment with executables compiled for platforms other than Windows XP on IA32.

Regarding the digital camera recognition we will continue collecting pictures from different camera makes and also deepen the research on the reasons for the differences. Other approaches to the problem will also be investigated.

## References

1. CONVAR Deutschland: Pc inspector. (`http://www.pcinspector.de/file_recovery/uk/welcome.htm`) accessed 2005-10-31.

2. Carrier, B.: The Sleuth Kit. (`http://www.sleuthkit.org/sleuthkit/index.php`) accessed 2005-10-25.
3. Farmer, D., Venema, W.: The Coroner's Toolkit (TCT). (`http://www.porcupine.org/forensics/tct.html`) accessed 2005-10-25.
4. Guidance Software: Encase forensic. (`http://www.guidancesoftware.com/products/ef_index.asp`) accessed 2005-10-31.
5. QueTek Consulting Corporation: File scavenger. (`http://www.quetek.com/prod02.htm`) accessed 2005-10-31.
6. iolo technologies: Search and recover. (`http://www.iolo.com/sr/3/`) accessed 2005-10-31.
7. ilook-forensics.org: ILook Investigator Forensic Software. `http://www.ilook-forensics.org/` (2005) accessed 2006-04-28.
8. Shanmugasundaram, K., Memon, N.: Automatic reassembly of document fragments via context based statistical models. In: Proceedings of the 19th Annual Computer Security Applications Conference. (2003) `http://www.acsac.org/2003/papers/97.pdf`, accessed at 2006-04-30.
9. Karresand, M., Shahmehri, N.: Oscar – file type identification of binary data in disk clusters and ram pages. In: Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006). LNCS (2006) To appear
10. Karresand, M., Shahmehri, N.: File type identification of data fragments by their binary structure. In: Proceedings from the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2006. (2006) To appear
11. Kharrazi, M., Sencar, H., Memon, N.: Blind source camera identification. In: Proceedings of ICIP '04. 2004 International Conference on Image Processing. Volume 1. (2004) 709–712
12. Lukáš, J., Fridrich, J., Goljan, M.: Determining digital image origin using sensor imperfections. In: Proceedings of SPIE Electronic Imaging, Image and Video Communication and Processing. (2005) 249–260
13. Lukáš, J., Fridrich, J., Goljan, M.: Digital bullet scratches for images. In: Proceedings of ICIP 2005. (2005)
14. Lukáš, J., Fridrich, J., Goljan, M.: Digital camera identification from sensor pattern noise. IEEE Transactions on Information Forensics and Security **1**(1) (2006) To appear.
15. Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In E. Jonsson el al., ed.: Recent Advances in Intrusion Detection 2004. Volume 3224 of LNCS., Springer-Verlag (2004) 203–222
16. Wang, K., Cretu, G., Stolfo, S.: Anomalous payload-based worm detection and signature generation. In Valdes, A., Zamboni, D., eds.: 8th International Symposium on Recent Advances in Intrusion Detection, RAID 2005. Volume 3858 of LNCS., Springer Verlag (2006) 227–246
17. Stolfo, S., Wang, K., Li, W.J.: Fileprint analysis for malware detection. Technical report, Computer Science Department, Columbia University, New York, NY, USA (2005) Review draft.
18. Li, W.J., Wang, K., Stolfo, S., Herzog, B.: Fileprints: Identifying file types by n-gram analysis. In: Proceedings from the sixth IEEE Sytems, Man and Cybernetics Information Assurance Workshop. (2005) 64–71
19. McDaniel, M., Heydari, M.: 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) 332.1

# F P2P-IDS – PDMST 2006

Claudiu Duma, Martin Karresand, Nahid Shahmehri och Germano Caronni. A Trust-Aware, P2P-Based Overlay for Intrusion Detection. *Proceedings of the 3rd IEEE International Workshop on P2P Data Management, Security and Trust (PDMST 2006)*, ss 692–697, 2006. IEEE Press.

# A Trust-Aware, P2P-Based Overlay for Intrusion Detection

Claudiu Duma[1], Martin Karresand[2], Nahid Shahmehri[1], and Germano Caronni[3]

[1]*Department of Computer and Information Science*
*Linköpings universitet, Sweden*
[2]*Swedish Defense Research Agency, Sweden*
[3]*Sun Microsystems Laboratories, USA*

## Abstract

*Collaborative intrusion detection systems (IDSs) have a great potential for addressing the challenges posed by the increasing aggressiveness of current Internet attacks. However, one of the major concerns with the proposed collaborative IDSs is their vulnerability to the insider threat. Malicious intruders, infiltrating such a system, could poison the collaborative detectors with false alarms, disrupting the intrusion detection functionality and placing at risk the whole system. In this paper, we propose a P2P-based overlay for intrusion detection (Overlay IDS) that addresses the insider threat by means of a trust-aware engine for correlating alerts and an adaptive scheme for managing trust. We have implemented our system using JXTA framework and we have evaluated its effectiveness for preventing the spread of a real Internet worm over an emulated network. The evaluation results show that our Overlay IDS significantly increases the overall survival rate of the network.*

## 1 Introduction

Current Internet attacks are automated and highly distributed, being capable of affecting very dispersed and large zones of the Internet in a very short time. For instance, the Code Red worm infected more than 350000 vulnerable systems in less than a day. Similarly, in August 2003 the Sobig.f worm produced more than 1 million copies of itself within the first 24 hours. At its peak, the worm accounted for more than one in every 17 e-mail messages.

The traditional intrusion detection systems (IDS) are limited and inferior in comparison to the attackers' capabilities. Typically, traditional IDSs work in isolation, only seeing relatively small subsections of the Internet, and thus they are unable of deriving significant trends in the whole network. This is especially true for new and emerging attacks, where being able to observe a large amount of deviant behavior would increase the detection and protection capabilities.

The collaboration of multiple intrusion detection mechanisms has the potential to increase the detection and protection capabilities of IDSs in the face of escalating aggressiveness of Internet attacks. However, the proposed collaborative architectures, e.g. [12, 5, 6, 7, 4, 11], have strong assumptions on the trustworthiness of the collaborative environment – all parties comprising the system are trusted to truthfully report intrusion events. A malicious insider, infiltrated in such a collaborative group, could disrupt the functionality of the collaborating systems and even put at risk the users depending on these systems [2]. For instance, a malicious peer could report false intrusion attempts with the goal of making other peers believe that an attack is imminent. In this case, the other peers will unnecessarily increase their security level, possibly even blocking services that are critical for their users.

In this paper we propose a P2P overlay for intrusion detection that addresses the insider threat by means of a trust-aware correlation engine and an adaptive scheme for managing trust. The trust-aware correlation engine is capable of filtering out warnings sent by untrusted or low quality peers, while the adaptive trust management scheme uses past experiences of peers to predict their trustworthiness. To alleviate the network flooding concerns, which might appear in collaborative IDSs, we also devise adequate safeguard mechanisms which protect the peers and the network from message overload.

We have implemented our proposed Overlay IDS in an experimental prototype that is based on JXTA P2P framework [3]. We evaluated the effectiveness of our solution in preventing the spread and infection by the Slapper worm [10]. Our experiments show that the use of our Overlay IDS increases the overall survivability of the network by more than 34%, on average for one trusted peer, compared to the case when the individual IDSs work in isolation.

The remainder of this paper is organized as follows. Sec-

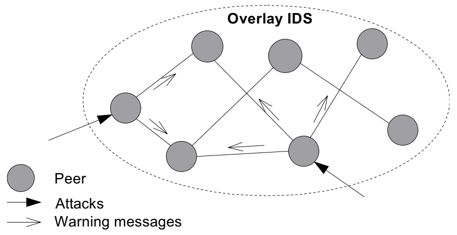


**Figure 1. Overlay IDS - network view**

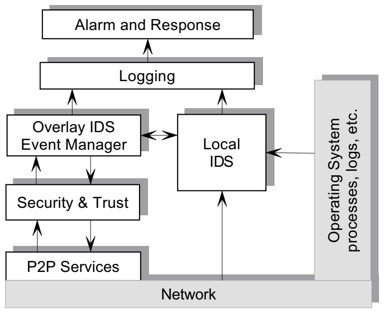


**Figure 2. Overlay IDS - peer view**

tion 2 introduces our Overlay IDS architecture, Section 3 details the trust-aware correlation mechanism, and Section 4 presents the evaluation test bed along with the evaluation results. Section 5 discusses related work, while Section 6 concludes the paper and gives pointers to our future work.

## 2 Overlay Intrusion Detection System

The Overlay IDS connects the local IDSs, which previously worked in isolation, in a peer-to-peer network. Each local IDS becomes therefore a node in the overlay P2P network. These connected nodes can communicate and cooperate with each other for the purpose of detecting and preventing Internet attacks. Figure 1 illustrates a part of an Overlay IDS network where two peers are attacked and thus they send warning messages to the other peers, which can take immediate, manual or automated, actions. If the warned peers are vulnerable to the attack, they can secure their systems before the attack reaches them, for instance by blocking the vulnerable port and installing relevant patches.

The Overlay IDS architecture, shown in Figure 2, contains the modules that have to be present in each of the peers that are part of the Overlay IDS.

The local IDS represents the intrusion detection systems, or related security systems such as anti-virus or firewalls, that are installed at a particular host. In general, the local IDS can employ various detection mechanisms, both network-based and host-based. When detecting an intrusion event, the local IDS informs (directly or through log files) the Event Manager about it.

The Event Manager is the key component in the architecture that extends the traditional IDS functionality with a P2P-based warning facility and a trust-aware correlation engine. Using the P2P Services component, the Event Manager informs other peers about the local intrusions as well as receives information from other peers about remote intrusions. The Event Manager filters the incoming and the outgoing warnings based on specifiable rules. The messages which pass the incoming filter are used, together with the local alerts, by the correlation engine.

The incoming and outgoing filters protect the local system and the Overlay IDS respectively from possible warning overload that could degrade the system's performance, flood the network, and even turn the Overlay IDS into a tool for DoS attacks.

The Security and Trust module assures the integrity and authenticity of the exchanged messages. It also implements the trust model that describes how trust relationships among interacting peers are established and maintained. It is important that the information carried by the warning messages is trustworthy and not tampered with. Otherwise the warnings will lead the system to erroneous decisions.

The P2P Services module integrates the P2P functionality by providing the set of services for network organization, management, and communication between peers. The P2P Services module can be implemented using available P2P frameworks and protocols such as JXTA or Pastry. To achieve the Overlay IDS functionality the P2P Services module must provide at least two services: sending and receiving of messages to and from the peers in the P2P network. This is usually realized through group communication primitives.

To assure interoperability of different IDSs, we use the Intrusion Detection Message Exchange Format (IDMEF) [1] for the warning messages.

## 3 Trust-Aware Alert Correlation

The life cycle process of a peer is depicted in Figure 3. Each peer correlates and analyzes locally generated alerts and warnings from other peers. The correlation is aware of peers' trustworthiness. That is, the correlation block takes input information from the acquaintance list - a list of peers which are trusted by the local peer. If the correlation engine detects a new incident, the Incident Response is activated. Whether the event was an incident or not, the local peer will always evaluate the event and accordingly adjust its trust in the acquaintance peers.

### 3.1 Trust Management

The advantages brought by the Overlay IDS rely on the fact that local IDSs can communicate and share intrusion related events. However, this also brings the question of trusting the events that the peers report. For instance, a malicious peer could report false intrusion attempts with the goal of making other peers believe that an attack is imminent. In this case, the other peers will unnecessarily increase their security level, probably even blocking their services. To

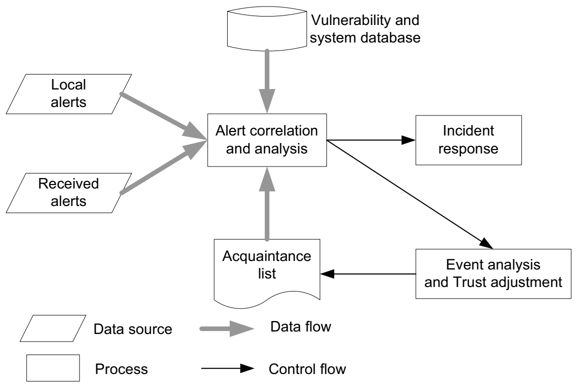


**Figure 3. The life cycle process of a peer**

cope with this problem we employ an adaptive trust mechanism for the selection of trustworthy peers.

Each peer $P_i$ maintains a list of acquaintance peers, which contains peers that $P_i$ has interacted with and knows their trustworthiness. For each peer $P_j$ in the acquaintance list, $P_i$ keeps two variables: $s_{ij}$ denoting the number of successful experiences of i with j; and $u_{ij}$ denoting the number of unsuccessful experiences of i with j. With these, the trustworthiness of peer j, as computed by peer i, is:

$$t_{ij} = w_s \frac{s_{ij} - u_{ij}}{s_{ij} + u_{ij}} \tag{1}$$

The *significance weight* $w_s$ depends on the total number of experiences that are available for trust computation. If there are only a few experiences available, then the formula can not truly reflect peer's trustworthiness. Thus, if the number of experiences $s_{ij} + u_{ij}$ is under a certain minimum $n$ then $w_s = (s_{ij} + u_{ij})/n$, and otherwise $w_s = 1$.

When assessing a certain intrusion event, $P_i$ takes in consideration the warnings that have been sent by the peers in the acquaintance list. Warnings from more trusted acquaintances will have a greater impact than warnings from less trusted ones. Moreover, warnings will be taken in consideration only if the trust of the sending peer is greater than a certain $trust\_threshold$.

The acquaintance list is dynamically built and maintained such that only peers whose trustworthiness passes the trust_threshold stay in the list. We use a probation flag to mark peers whose trustworthiness is below the trust_threshold. A peer which is flagged as being *under probation* must pass the trust_threshold in a given probation period. If the flagged peer passes the threshold in the given time, the flag is removed. Otherwise, the peer is removed from the acquaintance list, and a new randomly chosen peer will take its place. By randomly selecting a new peer, the update acquaintance algorithm (Algorithm 1) continuously searches for the best available candidates for the acquaintance list. Note that newly added peers will be flagged and will start with a probation period. If they cannot pass the trust_threshold during the probation period, they will be removed from the list and other peers will be taken for probation.

Thus, for each peer $P_j$ in the acquaintance list, $P_i$ also maintains a probation flag $pf_{ij}$, denoting whether $P_j$ is flagged or not, and a probation time $pt_{ij}$, denoting the time elapsed since $P_j$ was flagged for probation. The probation time is stated in terms of number of decision events at the local peer $P_i$.

After each intrusion decision event, peer $P_i$ will update the number of successful and unsuccessful experiences as well as the probation time. $P_i$ has an experience with $P_j$ whenever $P_i$ has to make an intrusion decision and $P_j$ has sent or was expected to send information that would help $P_i$ in the decision. Four cases are distinguished, as follows:

1. if there was an attack and $P_j$ has sent a warning about it:
   $s_{ij}$ = $s_{ij} + 1$, $u_{ij}$ = $u_{ij}$, and $pt_{ij}$ = $pt_{ij} + 1$;
2. if there was an attack and $P_j$ did not warned about it:
   $s_{ij}$ = $s_{ij}$, $u_{ij}$ = $u_{ij} + 1$, and $pt_{ij}$ = $pt_{ij} + 1$;
3. if there was no attack but $P_j$ has sent a warning:
   $s_{ij}$ = $s_{ij}$ and $u_{ij}$ = $u_{ij} + 1$, and $pt_{ij}$ = $pt_{ij} + 1$;
4. if there was no attack and $P_j$ has sent no warning:
   $s_{ij}$, $u_{ij}$, and $pt_{ij}$ remain unchanged.

Trust $t_{ij}$ increases for case 1, remains unchanged for case 4, and decreases for cases 2 and 3.

For the whole trust mechanism to work, it must be possible for peers to assess whether a certain intrusion event was indeed an attack or not. This is represented by the Event Evaluation block in the life cycle process of a peer, Figure 3. The evaluation could be done at the moment of the incident generation, if enough information exists, or after the incident has passed. For post mortem evaluation of an incident, peers could take as input both local information, such as clear signs of local intrusions, or globally available information, such as reports of a new worm breakout from specialized emergency organizations.

In addition to the dynamically built acquaintance list, a peer could also have a manually managed list of trusted peers. This list contains peers for which the trust has been built off-line, e.g. by direct human contact. Such a list is particularly beneficial when a new peer joins the Overlay IDS and has no acquaintances.

## 3.2 Alert Correlation

The local peer will correlate the received alerts and its own alerts (if any). When correlating received alerts, the local peer takes in consideration the trustworthiness of the sender peers. In particular, the confidence level of the correlated alert is computed by combining the received confidence with the trust as it follows:

$$C_i = w_{dir} * c_i + w_{ind} * \frac{1}{N} \sum_{j=1}^{N} c_j * t_{ij} \tag{2}$$

**Algorithm 1** Update acquaintance list

```
for all $P_j \in$ acquaintance_list do
  if $t_{ij} <$ trust_threshold $\wedge$ $pf_{ij}$ == false then
    $pf_{ij}$ = true; //flag the peer
    $pt_{ij}$ = 0;
  end if
  if $pf_{ij}$ == true then
    if $pt_{ij} >$ probation_time $\wedge$ $t_{ij} <$ trust_threshold then
      Remove acquaintance $P_j$;
      Randomly select another peer as the j'th acquaintance;
      $pt_{ij}$ = 0; $s_{ij}$ = 0; $u_{ij}$ = 0;
    end if
    if $t_{ij} >=$ trust_threshold then
      $pf_{ij}$ = false; // remove the flag
    end if
  end if
end for
```

$C_i$ is the confidence in the correlated alert, as correlated by peer $P_i$, and $c_j$ is the confidences of the alert received from peer $P_j$. The direct and indirect weights, $w_{dir}$ and $w_{ind}$, are used by peer $P_i$ when combining the direct knowledge (locally generated alerts) and the indirect knowledge (received alerts) respectively. The number of peers from the acquaintance list, not flagged and which have sent alerts used in the correlation, is denoted by $N$.

To avoid dependence on one or a very small number of peers, we require that N be greater than a minimum threshold number of trusted peers denoted by $N_{min}$. If N is lower than this limit, the weight $w_{ind}$ is decreased by $N/N_{min}$. Thus, a high $N_{min}$ threshold assures that the decisions taken by a peer can not be controlled by a small set of colluding malicious peers. However, a high $N_{min}$ might also affect the capacity of detecting true positive incidents when only a small number of (truthful) warnings are available for correlation. The tradeoff brought by $N_{min}$ is further evaluated in the next section.

If the confidence of the correlated alert passes a certain threshold, the peer will activate the Incident Response module that takes passive or active (and possibly automated) actions. Although a critical point in the intrusion protection chain, the response action is beyond our current work.

## 4 Implementation and Evaluation

We have implemented an evaluation prototype of our proposed Overlay IDS architecture based on JXTA P2P framework [3]. JXTA has support for basic P2P services, including secure group communication primitives. JXTA is based on Java, which assures platform independence for our prototype.

The Event Manager is implemented as a JXTA application, while for the local IDS module we used SNORT [9]. Note that plug-ins exist that enable SNORT to generate IDMEF compliant alerts.

The goal of the experiments is to evaluate the effectiveness of our Overlay IDS in defending against a real Internet worm attack when only a part of the entire population of peers is capable of detecting and protecting from the attack. All the other peers are vulnerable to the worm attack until they receive warning messages.

We have compared the number of infected peers when the Overlay IDS was disabled and enabled respectively. Without the Overlay IDS all the vulnerable peers were eventually infected, whereas with the Overlay IDS in place a significant fraction of the initial vulnerable peers could actually be saved by the warnings sent by the peers capable of detecting the threat.

### 4.1 Evaluation Test Bed

We have emulated our test bed network using VMWare 4.0 (`www.vmware.com`), such that all the attacks conducted during the experiments were contained within a virtual network.

We conducted the experiments for a virtual network with 36 clients, grouped on 6 distinct sub-networks. All clients were running RedHat Linux 7.3 and Apache 1.3.23 web server with OpenSSL 0.9.6b encryption enabled. Such a configuration is vulnerable to the Slapper worm [10], which was used in the experiments. Each client had the Java Runtime Environment installed to enable the execution of JXTA. Also, the clients capable of detecting the worm had SNORT 2.2.0 with libpcap 0.8.3 [9] installed.

### 4.2 Experiments

We defined the survival rate as the number of nodes which resisted the worm attack divided by the number of all nodes in the network. We compared the survival rate for the case when the clients were part of the Overlay IDS and when they were not. We have also considered the cases where correlation of alerts from $N_{min}$ equals 1, 2, and 3 peers were needed to raise an alarm ($N_{min}$ was defined in Section 3.2).

In our experiments we have considered two sets of clients – one set which was capable of detecting and thus protecting from the worm and one set which was not capable of detecting the worm. With the Overlay IDS enabled, the protected clients sent warning messages whenever they detected an attack. We assumed that vulnerable clients, which received warnings before being hit by the worm, could immediately protect themselves and resist the impending attack. The instantaneous protection could be possible by closing the attacked port or, at extreme, disconnect from the network until the vulnerability is fixed.

We experimented with 1, 3, 6, and 9 protected peers, while all other peers were vulnerable to the worm attack. For each case we have repeated the experiment five times. Figure 4 plots the average survival rate for different config-

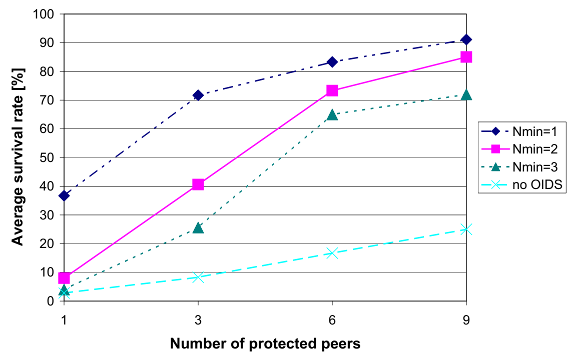


**Figure 4. Survival rate**

urations of $N_{min}$. The results indicate a significant increase of the average survival rate when the Overlay IDS was used compared to when it was not used (no OIDS). For instance, for 3 protected peers the average survival rate with Overlay IDS was 71.7% for $N_{min} = 1$, 40.6% for $N_{min} = 2$, and 25.6% for $N_{min} = 3$, whereas for the case without Overlay IDS the average survival rate barely reached 8.3%.

Figure 4 shows that the average survival rate increases asymptotically with the number of protected clients. That is, the more clients protected the higher the probability is that the worm will hit first clients from the protected set which will then warn the other clients. Also having more clients sending messages increases the chance that a vulnerable client will receive the warning in time.

Figure 4 also shows that, on one hand, the survival rate decreases with the increase of the number of trusted peers needed for correlation ($N_{min}$). On the other hand, the resiliency of the network increases with $N_{min}$ as the impact of a malicious or faulty peer is diminished. This also means that the false alarm rate decreases accordingly. Thus, by configuring $N_{min}$, we can achieve either a faster detection system with a higher survival rate but prone to false alarms, or a more robust system with a lower level of false alarms but which (due to lack of trust) could also miss some of the real alarms.

The current prototype implementation was not tuned to efficiently use the network bandwidth. Messages were sent whenever attacks were detected, without any intelligent filtering. Although this is not economical in respect to communication, it gives the best functioning conditions for the detection and protection mechanisms. In our future experiments we plan to use efficient communication models, and investigate the possible tradeoffs between the effectiveness of the detection mechanisms and the communication efficiency gained through message filtering policies.

## 5 Related Work

Fault tolerance, scalability, and self organization are properties generally accounted for P2P systems. These properties make the P2P technologies attractive for developing a large spectrum of applications beyond the controversial file sharing. Very recently researchers have also signaled the benefits of P2P technologies for intrusion detection systems. Therefore, we focus our attention on the few recently proposed intrusion detection systems which, as ours, employ mechanisms pertaining to P2P.

Vlachos and colleagues introduce NetBiotic [11], a system which monitors the attack activity at each host and uses P2P mechanisms to build a global view of the attacks in the whole network. Though the NetBiotic architecture is similar to ours, the approach to sending warning messages differs significantly. NetBiotic monitors the change in attack frequency and propagates only that information, whereas our system sends alarms to other peers directly, achieving a faster reaction time. Moreover, our Overlay IDS transfers the function of detecting alerts from the sensor peers to the peers that could correlate information received from very sparse sources.

Ramachandra and Hart present an intrusion detection system based on mobile agents and the concept of neighborhood watch [8]. In their system, each node has a set of neighbors, which are examined for suspicious or malicious behavior through the means of mobile agents. The agents are sent to inspect the neighbors' systems and report back the results. If intrusions or other anomalies are suspected, the inspected node may be declared compromised and excluded from the network. Although the neighborhood watch is appealing, the approach also raises serious security concerns. A malicious host can tamper with the agent such that the agent will report only good information, thus overcoming the intrusion detection and putting at risk all the hosts in its virtual neighborhood. Sending agents to inspect the neighbors' systems is also a rather intrusive mechanism with serious consequences on privacy.

Another P2P-based approach to intrusion detection is proposed by Janakiraman and colleagues [4]. The authors introduce Indra, a P2P solution to network intrusion detection and prevention. The system is based on a set of specialized and proprietary implemented daemons, which provide security services such as detection of intrusions and access control to resources. Instead, our architecture reuses existing IDS tools and services. Also, our approach aims at detecting and protecting from both internal and external attacks with regard to the Overlay IDS, while their approach considers only the attacks that originate from inside the P2P network.

Krügel and colleagues propose a decentralized system to collect and correlate events from different sources scattered over a protected network [5]. The authors have designed a specification language, which models intrusions as blocks of patterns, and a P2P algorithm, which detects the occurrence of such attack patterns. The system consists of distributed collaborating sensors deployed at various locations throughout the protected network. However, the system is not purely P2P as it has a central control unit to administer the system, posing therefore a single point of failure problem. By contrast, our Overlay IDS is purely decentralized.

Additionally, the solution in [5] uses specifically designed sensors for signaling intrusion events, while our proposal reuses and integrates the already deployed security mechanisms and tools.

We note two major differences of our work as compared to other attempts of employing P2P technologies for intrusion detection. First, the previous work has only partly addressed the issue of trust. For instance, Janakiraman and colleagues [4] recognize the major role of trust in a P2P environment but do not propose any solution. In our work, we identify trust as a major component in our architecture. Second, the related work has limited or no evaluation, thus leaving open questions regarding the effectiveness and viability of the proposed solutions. In contrast, we have designed a secure and extensible evaluation test bed which allows us to validate, through experiments, our proof of concept.

Alert correlation has been proven valuable in the cases where the sources of alerts are heterogeneous intrusion detection devices. Our alert correlation engine is similar to M2D2 [6], in that it uses trust and confidence, and to M-correlator [7], especially for the alerts' relevance. However, we also take in consideration the potential malicious behavior of peers in the Overlay IDS and thus, augment the previous correlation techniques with a trust management mechanism which is capable of detecting and avoiding misbehaving peers. Our trust mechanism uses past experiences to predict peers trustworthiness and, using the update algorithm, it continually searches for the optimal set of trusted peers.

## 6 Conclusions and Future Work

Although the collaborative IDSs have a great potential for addressing the challenges posed by current Internet attacks, the proposed systems are vulnerable to malicious insiders. In this paper, we have addressed the insider threat issue by proposing a trust-aware, P2P-based Overlay IDS. At the core of our system are a trust-aware alert correlation engine and an adaptive scheme for trust management. The main advantages of our solution are:

- The Overlay IDS is totally decentralized.
- The trust-aware correlation engine makes the Overlay IDS resistant to insider threats.
- Our approach does not require the development of specialized IDS mechanisms, but it is capable of making use of the existing and established IDS mechanisms and tools.
- The Overlay IDS approach is flexible and can be applied to open environments, with fully autonomous peers, as well as to closed network settings.

We have implemented the Overlay IDS using JXTA, and we have evaluated its effectiveness for preventing the spread of an Internet worm within an emulated network. As shown by the evaluation results, the use of our Overlay IDS increases the overall survival rate of the network with more than 34% on average, for one trusted peer.

Efficiency and scalability are some of the critical issues that we would like to address in our future work. We plan to adopt efficient communication models and experiment with different message filtering and peer grouping strategies. We are currently extending our evaluation test bed and plan to conduct experiments with different types of attacks, for different operating systems.

## References

[1] H. Debar, D. Curry, and B. Feinstein. The Intrusion Detection Message Exchange Format. IETF Internet draft, January 2005. `http://ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt`.

[2] C. Duma, N. Shahmehri, and E. Turcan. Resilient Trust for Peer-to-Peer Based Critical Information Infrastructures. In *Proceedings of 2nd International Conference on Critical Infrastructures (CRIS 2004)*, October 2004.

[3] L. Gong. Project JXTA: A Technology Overview. White paper, 2002. `http://www.jxta.org/project/www/docs/jxtaview_01nov02.pdf`.

[4] R. Janakiraman, M. Waldvogel, and Q. Zhang. Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention. In *Proceedings of 12th IEEE Workshop on Enterprise Security (WETICE)*, pages 226–231, June 2003.

[5] C. Krügel, T. Toth, and C. Kerer. Decentralized Event Correlation for Intrusion Detection. In *Proceedings of the 4th International Conference Seoul on Information Security and Cryptology*, pages 114–131, 2002.

[6] B. Moring, L. Mé, H. Debar, and M. Ducassé. M2D2: A Formal Data Model for IDS Alert Correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection*, pages 115–137, 2002.

[7] P. A. Porras, M. W. Fong, and A. Valdes. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In A. Wespi, G. Vigna, and L. Deri, editors, *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'2002)*, volume 2516 of *LNCS*, pages 95–114. Springer, 2002.

[8] G. Ramachandran and D. Hart. A P2P intrusion detection system based on mobile agents. In *Proceedings of the 42nd annual Southeast regional conference*, pages 185–190, New York, NY, USA, 2004.

[9] SNORT. The open source network intrusion detection system, `http://www.SNORT.org`.

[10] P. Szor and F. Perriot. Linux.Slapper.Worm. `http://securityresponse.symantec.com/avcenter/venc/data/linux.slapper.worm.html`, 2003.

[11] V. Vlachos, S. Androutsellis-Theotokis, and D. Spinellis. Security applications of peer-to-peer networks. *Computer Networks*, 45(2):195–205, 2004.

[12] G. B. White, E. A. Fisch, and U. W. Pooch. Cooperating security managers: A peer-based intrusion detection system. *IEEE Network*, 10(1):20–23, January/February 1996.

# G Oscar – WDFIA 2006

Martin Karresand och Nahid Shahmehri. Oscar – Using Byte Pairs to Find File Type and Camera Make of Data Fragments. *Proceedings of the 2nd European Conference on Computer Network Defence, in conjunction with the First Workshop on Digital Forensics and Incident Analysis (EC2ND 2006)*, ss 85–94, 2007. Springer Verlag.

# Oscar – Using Byte Pairs to Find File Type and Camera Make of Data Fragments

Martin Karresand[1,2], Nahid Shahmehri[1]

[1] Dept. of Computer and Information Science
Linköpings universitet,
Linköping, Sweden
[2] Dept. of Systems Development and IT-security
Swedish Defence Research Agency,
Linköping, Sweden
{g-makar,nahsh}@ida.liu.se

**Abstract** Mapping out the contents of fragmented storage media is hard if the file system has been corrupted, especially as the current forensic tools rely on meta information to do their job. If it was possible to find all fragments belonging to a certain file type, it would also be possible to recover a lost file. Such a tool could for example be used in the hunt for child pornography. The Oscar method identifies the file type of data fragments based solely on statistics calculated from their structure. The method does not need any meta data to work. We have previously used the byte frequency distribution and the rate of change between consecutive bytes as basis for the statistics, as well as calculating the 2-gram frequency distribution to create a model of different file types. This paper present a variant of the 2-gram method, in that it uses a dynamic smoothing factor. In this way we take the amount of data used to create the centroid into consideration. A previous experiment on file type identification is extended with .mp3 files reaching a detection rate of 76% with a false positives rate of 0.4%. We also use the method to identify the camera make used to capture a .jpg picture from a fragment of the picture. The result shows that we can clearly separate a picture fragment coming from a Fuji or Olympus cameras from a fragment of a picture of the other camera makes used in our test.

**Keywords:** Camera recognition, computer forensics, data recovery, file type identification, 2-gram frequency distribution.

## 1 Introduction

Being able to survey the contents of a hard disk is vital in, for example, a forensic examination or during recovery of lost data. Unfortunately fragmentation together with a corrupt file system complicates the mapping out of the hard disk. Depending on the level of fragmentation the task can be anything from hard to almost impossible. To further complicate the situation, there are, to the best of our knowledge, no tools [1,2,3,4,5,6,7] available that can identify the file type of a data fragment without relying on meta data in some form. Due to this lack of adequate tools a forensic examiner can miss fragments of .jpg pictures containing child pornography on a hard disk or other storage media.

If all fragments of a certain file type can be found, the fragments can be used to recreate the original files. A method [8] to do the latter exists, but this work aims to provide a solution to the first part.

To be able to determine the file type of a data fragment without using meta data we have to look at the structure and contents of the fragment. For text-based data this is trivial, but for binary data we cannot rely on finding readable strings to use. The structure of the data itself is, however, unique for different types of files. This is true even for compressed files, which can be seen in Fig. 1 showing histograms of the byte frequency distribution of typical .jpg and .zip files.

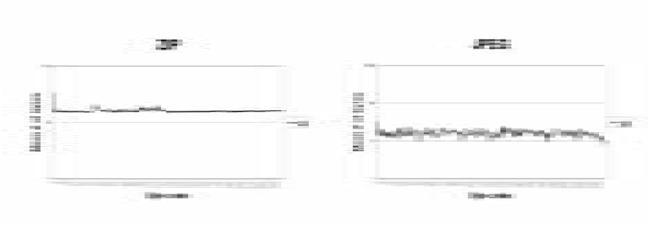


**Figure 1.** Histograms calculated over a number of 4 kB blocks of .jpg and .zip files. The headers of the .jpg files were stripped of before calculating the statistics. A logarithmic scale is used for the Y-axis.

The Oscar method [9,10] has been proposed as a method to identify the file type of arbitrary data fragments in a way that is fast enough to be used as the main tool for forensic examinations of current and future storage media. The first version of the method uses 1-grams, i.e. single bytes, to create a histogram of the byte frequency distribution (BFD) of a data block, and to measure the rate of change (RoC) of consecutive bytes. The Oscar method has been updated [11] to use byte pairs, 2-grams, instead of single bytes. The first test results indicated that the method made it possible to tell some digital camera makes apart using 4 kB picture data fragments.

This paper deepens the camera recognition tests, first presenting the necessary background. The implementation of the 2-gram method has been slightly updated and we perform the tests using larger amounts of data for each camera make.

## 2 Method

The Oscar method is built measuring the distance between an unknown sample and a model (called centroid) of a specific file type. The centroid contains the mean frequency and the corresponding standard deviation values of the bytes or byte pairs of the file type.

### 2.1 2-gram Oscar

The centroid modelling a file type is created by counting the number of unique 2-grams in a large number of 1 MB blocks of data of a specific file type. The data in each 1 MB block is used to form a frequency distribution of all 65536 different possible 2-grams, and the mean and standard deviation of each 2-gram is calculated and stored in two $256 \star 256$ matrices. The values are then scaled down to correspond to a block size of 4 kB.

The 4 kB size is used because it is equal to the common size of memory pages and disc clusters in current home computers. The reason for using 1 MB data blocks is to get a solid foundation for the mean and standard deviation calculations. When using a block size less than the possible number of unique 2-grams the calculations inevitably become unstable.

When the similarity between a sample fragment and a centroid is measured the same type of quadratic distance metric as for the 1-gram Oscar method is used. Equation 1 describes the metric, where matrix $S$ depicts the sample 2-gram frequency distribution and $C$ the mean value matrix of the centroid. The standard deviation of 2-gram $ij$ is represented by $\sigma_{ij}$. There is also a smoothing factor $\alpha = 0.000001$, which is used when $\sigma_{ij} = 0$ to avoid division by zero.

$$\mathrm{d}\left(S, C\right) = \sum_{i=0, j=0} \left(s_{ij} - c_{ij}\right)^2 / \left(\sigma_{ij} + \alpha\right). \tag{1}$$

For the camera recognition experiments the smoothing factor $\alpha$ is calculated as

$$\alpha = \frac{1}{2^n} \qquad n = [1, 2, \ldots, 60], \tag{2}$$

where $n$ is the size of the data in MB. For sizes larger than 60 MB $n = 60$.

The quadratic term is used to favour smaller deviations over larger. As can be seen in Eq. 1 the differences are weighted by the standard deviation of each 2-gram. This is done to lessen the influence of 2-grams that vary significantly and in that way only use the key features of the 2-gram distribution.

The distance value of a sample is compared to a empirically predetermined threshold. Samples giving distance values below the threshold are categorised as being of the same file type as the centroid.

The reason for not using a more advanced distance metric is execution speed. The current maximum size of a hard disk is 750 GB and hence a forensic examination can involve a tremendous amount of data to scan.

### 2.2 Advantages and Disadvantages

There are both advantages and disadvantages to using 2-grams. The disadvantages of using 2-grams are the exponential increase of the centroid's size and the lower precision in measuring distance, due to the scaling down of the values of the centroid.

The size of the centroid mainly affects the execution speed of the algorithm, but also the memory footprint. In theory the effect is a 256-fold increase in processing time and memory footprint compared to the original 1-gram methods. The increase in memory footprint can be ignored due to the amount of RAM used in modern computers; the requirement of the current implementation is a few hundred kB of RAM. By optimising the algorithm the processing time can be decreased, but the decrease is not bounded.

The problem of scaling down the values cannot be disregarded. If we assume a uniform distribution of byte pairs in a file and look at a 4 kB fragment of that file, the probability of a specific byte pair is $\frac{4096}{65536} = \frac{1}{16}$. Each time a byte pair is found in the 4 kB fragment, it equals finding 16 such byte pairs in a 64 kB fragment. This renders the distance measurement inaccurate and increases the risk of detection errors. By using fragments of 1 MB (16 times larger than 64 kB) when creating the centroid we lessen the impact of the down-scaling.

The main advantage of the 2-gram method is the automatic inclusion of the order of the bytes into the method, signalling that an unknown block does not conform to the specification of a specific file type and thus should not be categorised as such. A typical example is the appearance of certain .jpg header byte pairs disallowed in the data part. Another example is special byte pairs never occurring in files created by a specific camera or software.

## 3 Evaluation

The experiments presented in [11] have been extended with one more camera make for the camera recognition and a new file type for the file type identification. The experiments are described in the following subsections.

### 3.1 File type identification

The files for the experiment were taken from an office computer running Windows XP, fully patched. The individual files were padded with zeros to adjust their lengths to a multiple of 4 kB.

The .jpg file centroid, zip file centroid, and Windows executable centroid used in earlier experiments were complemented by an .mp3 file centroid. The detection rate of each centroid was then measured together with the corresponding false positives rate.

The .jpg centroid was created from 104 MB of picture data, the headers were stripped of the original pictures before we created the centroid. For the .mp3 centroid we used 859 MB of data. The zip file centroid was made from 104 MB of different zip-based files, created using WinZip and the Linux gzip and zip utilities. The complete zip files were used. The Windows executable file centroid was created from a 147 MB large concatenation of different files, all being categorized as either “MS-DOS executable (EXE), OS/2 or MS Windows”, or “MS-DOS executable (EXE)” by the Linux *file-4.12* utility. All such files begin with the magic number 0x4D5A, “MZ”.

The experiment was performed using a 72 MB test file consisting of 57 files of 51 different types concatenated into one. The templates used as ground truth for the experiment labelled the .jpg headers as non-.jpg. The executable, mp3 and zip files were labelled as true positives in their entirety. The distribution of 4 kB fragments in the test file was 1150 .jpg, 4397 Windows executables, 1426 .zip, 1235 .mp3, and 10275 fragments of other types.

### 3.2 Camera recognition

The camera recognition experiment was performed using an 80/20 model, i.e. we created a centroid from 80 % of the picture data for a specific camera and used the remaining 20 % for testing. Consequently, there were five sub-experiments, which were used to calculate a mean value for the detection rate. Each centroid was tested against all different cameras giving us the Cartesian product of the picture and centroid combinations.

The amount of data used to create the centroids and to test them were larger than in the previous camera recognition experiment, where the centroids were created from 8 MB of data. This time we used all data available, ranging from 14 MB to 1.5 GB, with a mean of 335 MB.

The different cameras used for the test are shown in Table 1.

**Table 1.** The camera make and models used for the experiments

| | |
|---|---|
| Canon DIGITAL IXUS 400 | Canon DIGITAL IXUS v |
| Canon PowerShot A70 | Casio EX-Z40K |
| Fuji FinePix2400Zoom | Fuji FinePix E550 |
| Kodak CLAS Digital Film Scanner/HR200 | Konica KD-310Z |
| Konica Minolta DiMAGE G400 | Konica Minolta DiMAGE X60 |
| Nikon D50 | Nikon E3500 |
| Olympus D600 | Sony DSC-P8 |

The pictures are all standard family photographs, both indoor and outdoor, with varying lighting conditions and settings. Some of the pictures are rotated using the camera’s built-in editing facility, otherwise the pictures are unedited.

## 4 Result and Discussion

The .jpg and .mp3 type files both had detection rates above 75%, with less than 0.5% false positives. Regarding the camera recognition experiment, the Fuji and Olympus cameras stand out clearly. All other cameras are indistinguishable, which were not the case in the previous camera recognition experiment.

### 4.1 File type identification

The results of the file type recognition experiment can be seen in Fig. 2.

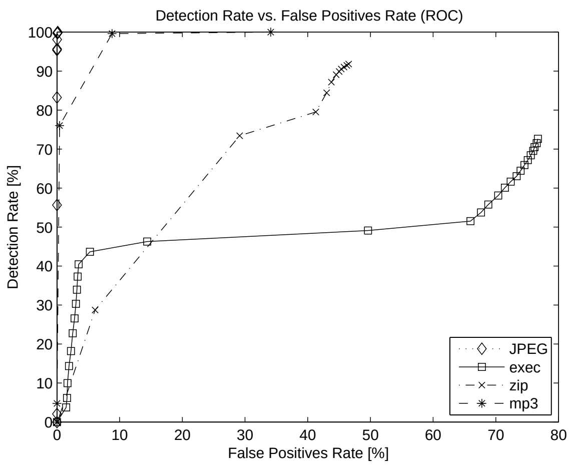


**Figure 2.** The detection rate vs. the false positive rate using .jpg, Windows executable, mp3 and zip centroids

The 2-gram method reaches 100% detection rate for .jpg file fragments with a false positive rate of 0.1% and there are no false positives until the detection rate reaches 98.1%. The .mp3 curve is only slightly lower than the .jpg, giving a detection rate of 76% with a false positives rate of 0.4%

Since the false positive rate of the zip files and the Windows executable files are almost or higher than 50% these cannot be used as they are. We do, however, think that the results can be improved by fitting the selection of data for centroid creation to the standards connected to the file types.

### 4.2 Camera recognition

Fragments of pictures taken by the Fuji and Olympus cameras are clearly distinguishable from picture fragments of the other cameras. The graphs of the two groups of cameras are almost identical within the groups. Hence the hypothesis presented in the previous paper [11] on the Oscar method, saying that it would be possible to tell cameras apart by looking at the structure of their pictures, does not hold.

Figure 3 shows the signature matrices coded as bitmaps for the different camera makes. The colour saturation of the bars represents the number of true positives: the darker the bar, the higher the level. The centroids, i.e. bars, correspond to, from left to right (see also Table 1): Casio, Konica G400, Nikon

E3500, Sony, Canon IXUS 400, Fuji 2400, Nikon D50, Canon A70, Olympus, Konica 310, Konica X60, Fuji E550, the scanner, and Canon IXUS v.

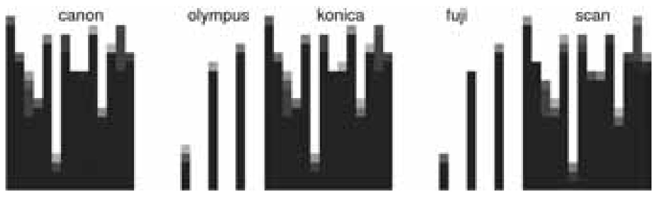


**Figure 3.** A comparison chart for pictures mapped by different centroids. Since the results can be grouped into two groups we only show five graphs.

As can be seen in Fig. 3 the Casio camera centroid has a better detection rate than the other centroids. The scanner centroid gives the highest detection rate for its own pictures, which might be an indication of a possibility to detect scanned picture fragments. The Fuji and Olympus pictures contain a special marker sequence that the other pictures do not, consequently only these centroids can detect them. The Canon A70 and Konica 310 graphs in the figure are used to represent the other cameras' graphs, which all look more or less the same.

### 4.3 Discussion

The detection rate and false positives rate of the Windows executables centroid are poor to say the least, in fact, they are even worse than guessing. There are, however, a large number of different file formats fitting our requirement that the files should start with the "MZ" byte pair. The structure also varies within the files themselves. Typically there can be sections of readable ASCII characters, well structured binary tables, and sections of machine code, arbitrarily ordered.

Due to the limited set of cameras used for the experiments we cannot really say anything more than that it is possible to distinguish Fuji and Olympus picture fragments from picture fragments of the other cameras included in the experiment. The reason for Fuji and Olympus being recognizable are their use of restart markers in the data, which make camera recognition trivial. Since also some scanner software add restart markers to their files, the group will grow larger and consequently the camera recognition ability will diminish.

The large amount of data in this experiment, in combination with the dynamic $\alpha$ value (see Eq. 2), are the main reasons for the uniform result. What can be discussed is whether the stabilising feature of the $\alpha$ is preferable or not. If we compare this experiment to the previous camera recognition experiment, we still see some minuscule differences in the detection rates. Are these differences related to the make of a camera and if so, can they be robustly enhanced and detected? To what extent do the calculation of $\alpha$ suppress these differences?

## 5 Related Work

The works presented in this section are related to the 2-gram method either by the use of an n-gram algorithm for operation on fragmented data in different ways, or by their ability to identify the camera used to capture a picture without relying on make and model information in the file header.

The 1-gram Oscar method [9,10] uses two algorithms, byte frequency distribution and rate of change, to identify the file type of a data fragment on byte level. For BFD the mean and standard deviation of each byte frequency is used to form a model, called centroid, of a specific file type. The similarity of an unknown sample and the centroid is measured by calculating the difference between each byte value frequency of the sample and the centroid, using a sum of squares weighted by the standard deviation of each byte frequency.

The rate of change (RoC) algorithm measures the absolute value of the difference between consecutive bytes. This approach allows the smoothness of the byte stream of a data block to be taken into consideration, decreasing the number of false positives from data blocks that are of a different file type from the centroid, but which have a similar byte frequency histogram.

The 2-gram Oscar method [11] differs from the 1-gram method by the fact that the latter does not automatically and fully take special byte combinations (required or disallowed) into consideration. Nor does it consider if the rate of change between bytes is positive or negative.

An n-gram based method that is related to the 2-gram method is the intrusion detection system PAYL [12,13] from the research group lead by professor Stolfo at Columbia University, USA. The same group has also used n-grams for file type recognition using fileprints [14,15]. The main differences between these methods and the 2-gram method are the previous methods' dependency on file header information, as well as the fact that Stolfo and his coauthors do not consider byte order, which the 2-gram method does.

Also McDaniel and Heydari [16] use the concept of n-grams in a file type identification method. They use different statistical measures applied to single bytes in files to make finger prints of files. McDaniel and Heydari's main source of data is the header and footer parts of files. They try to consider the ordering of the bytes in files through the use of what they call *byte frequency cross-correlation*. One example of bytes with high byte frequency cross-correlation is the html tag markers "<" and ">". The McDaniel and Heydari method differs from the 2-gram method through the use of single bytes, as well as the their method's dependence on file header and footer data.

Yet another n-gram based method is presented in a paper by Shanmugasundaram and Memon [8]. They use a sliding window algorithm for fragmented file reassembly using n-gram frequency statistics. They let the window pass over the edges of the fragments and then combine those fragments having the highest probability of being consecutive based on the frequency of the n-grams found in the window. Their use of n-grams frequency statistics is similar to the 2-gram method, but they apply the method in another way and require all fragments of a file to be known in advance. Therefore the 2-gram method is applicable to the steps before Shanmugasundaram and Memon's method and is only partly related to their method. The two methods can, however, be joined together and used to first identify and then reconnect fragments of files.

Camera recognition applied to the pixel level of pictures is presented by Lukáš, Fridrich, and Goljan [17,18,19]. They base their method on the small deviations in uniformity in the CCD of a camera. The deviations are detectable even after compression and other picture manipulation exercises. In the papers they show that the deviations are unique to a specific camera. Another camera recognition method applied to the pixel level of pictures is presented by Kharrazi, Sencar and Memon [20]. They identify 34 features that can be used to identify which camera make and model was used to capture a picture. These two methods differ from the 2-gram method and its camera recognition ability in that both methods are applied to unfragmented, complete pictures and the 2-gram method works on fragmented data.

## 6 Conclusion and Future Work

In this paper we used an variant of the 2-gram Oscar method to identify the camera make used to capture a picture. Our preliminary results show that Fuji cameras as well as Olympus differ significantly from other camera makes and thus can be identified from a picture fragment. This is however directly related to the use of restart markers in the data, making the recognition trivial. We also tested the 2-gram method on .mp3 data fragments giving a detection rate of 76% with 0.4% false positives.

There is still work to be done in developing centroids covering more file types. We also need to further investigate the reason for the poor results of the executable file centroid. One reason can be that our definition of executable files is too general, since the current definition covers a large number of sub-types. We also need to experiment with executables compiled for platforms other than Windows XP on IA32.

As a natural continuation of the work with the Oscar method we will look into the problem of recreating parts or complete original files from the fragments found.

## References

1. CONVAR Deutschland: Pc inspector. `http://www.pcinspector.de/file_recovery/uk/welcome.htm` accessed 2005-10-31.
2. Carrier, B.: The Sleuth Kit. `http://www.sleuthkit.org/sleuthkit/index.php` accessed 2005-10-25.
3. Farmer, D., Venema, W.: The Coroner's Toolkit (TCT). `http://www.porcupine.org/forensics/tct.html` accessed 2005-10-25.
4. Guidance Software: Encase forensic. `http://www.guidancesoftware.com/products/ef_index.asp` accessed 2005-10-31.
5. QueTek Consulting Corporation: File scavenger. `http://www.quetek.com/prod02.htm` accessed 2005-10-31.

6. iolo technologies: Search and recover. `http://www.iolo.com/sr/3/` accessed 2005-10-31.
7. ilook-forensics.org: ILook Investigator Forensic Software. `http://www.ilook-forensics.org/` (2005) accessed 2006-04-28.
8. Shanmugasundaram, K., Memon, N.: Automatic reassembly of document fragments via context based statistical models. In: Proceedings of the 19th Annual Computer Security Applications Conference. (2003) `http://www.acsac.org/2003/papers/97.pdf`, accessed at 2006-04-30.
9. Karresand, M., Shahmehri, N.: Oscar – file type identification of binary data in disk clusters and ram pages. In: Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006). Lecture Notes in Computer Science (2006) 413–424
10. Karresand, M., Shahmehri, N.: File type identification of data fragments by their binary structure. In: Proceedings from the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2006, Piscataway, NJ, USA, IEEE (2006) 140–147
11. Karresand, M., Shahmehri, N.: Oscar – file type and camera identification using the structure of binary data fragments. In Haggerty, J., Merabti, M., eds.: Proceedings of the 1st Conference on Advances in Computer Security and Forensics, ACSF, Liverpool, UK, The School of Computing and Mathematical Sciences, John Moores University (July 2006) 11–20
12. Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In E. Jonsson el al., ed.: Recent Advances in Intrusion Detection (RAID) 2004. Volume 3224 of Lecture Notes in Computer Science., Springer-Verlag (July 2004) 203–222
13. Wang, K., Cretu, G., Stolfo, S.: Anomalous payload-based worm detection and signature generation. In Valdes, A., Zamboni, D., eds.: 8th International Symposium on Recent Advances in Intrusion Detection, RAID 2005. Volume 3858 of Lecture Notes in Computer Science., Springer-Verlag (2006) 227–246
14. Stolfo, S., Wang, K., Li, W.J.: Fileprint analysis for malware detection. Technical report, Computer Science Department, Columbia University, New York, NY, USA (2005) Review draft.
15. Li, W.J., Wang, K., Stolfo, S., Herzog, B.: Fileprints: Identifying file types by n-gram analysis. In: Proceedings from the sixth IEEE Sytems, Man and Cybernetics Information Assurance Workshop. (June 2005) 64–71
16. McDaniel, M., Heydari, M.: 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) 332.1
17. Lukáš, J., Fridrich, J., Goljan, M.: Determining digital image origin using sensor imperfections. In: Proceedings of SPIE Electronic Imaging, Image and Video Communication and Processing. (2005) 249–260
18. Lukáš, J., Fridrich, J., Goljan, M.: Digital bullet scratches for images. In: Proceedings of ICIP 2005. (2005)
19. Lukáš, J., Fridrich, J., Goljan, M.: Digital camera identification from sensor pattern noise. IEEE Transactions on Information Forensics and Security **1**(Summer 2) (2006) 205–214
20. Kharrazi, M., Sencar, H., Memon, N.: Blind source camera identification. In: Proceedings of ICIP '04. 2004 International Conference on Image Processing. Volume 1. (2004) 709–712

FOI
Totalförsvarets forskningsinstitut
Ledningssystem
Box 1165
581 11 LINKÖPING

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

www.foi.se