KUNGLIGA TEKNISKA HÖGSKOLAN EXAMENSARBETE

IT-forensiska undersökningar av datorsystem i drift

– Live computer forensics

Av Olle Bergdahl 2006

Uppsatsen motsvarar 20 poäng.
This thesis corresponds to 20 weeks of full-time work.

Sammanfattning

Inom området IT-forensiska undersökningar sker ett paradigmskifte. Från att tidigare i huvudsak omfattat datainsamling och analys av beslagtagna hårddiskar riktas nu uppmärksamhet mot datainsamling i driftsatta datorsystem, i synnerhet data lagrat i RAM. Data i RAM kan vara relevant för en brottsutredning.

I denna studie kartläggs kriterier för val av programvara och processer som kan användas vid datainsamling av data i RAM i driftsatta datorsystem. Programvaran DD kan användas för att genomföra denna datainsamling. Att använda programvara för datainsamling är förenat med vissa problem. En begränsning som identifierats är att administratörsrättigheter krävs i operativsystemet för att kunna använda DD. Ett annat problem är att programvara för datainsamling kommer att orsaka förändring av data i måldatorns hårddisk. Innan programvara tas i bruk måste dokumentation genomföras över vilka filer programvaran kommer att påverka. Programvara har beroenden till måldatorns operativsystem och kärna. Detta beroende kommer alltid att finnas då programvarulösningar används för datainsamling. En risk vid datainsamling i driftsatta datorsystem är att måldatorns kärna kan innehålla rootkits. Detta kan medföra att kärnan levererar data som är otillförlitlig, data som har låg datakvalitet.

Ett dilemma är att programvara som används för datainsamling själv kräver minne i måldatorn. Oallokerat minne kommer att skrivas över. Programvaran bör därför vara konstruerad så att minimalt med minne krävs för att kunna köra processen datainsamling. Vidare är granskningsmöjlighet av programvara ett kriterium som bör prioriteras. Genom granskning kan man verifiera att programvaran verkligen utför de funktioner och levererar de resultat som programvaran utlovar.

Lagringskapaciteten i hårddiskar är stor och ökar ständigt. Att genomsöka hela hårddisken efter relevant data under en överskådlig tid är ogenomförbart. Data i RAM kan emellertid ge vägledning var i hårddisken man med traditionella metoder ska leta. En brottsutredning kan på så sätt genomföras snabbare och effektivare. Insamlad data kan genom informationsanalys ge bevis i form av vilka processer som körs i datorn, chatmeddelande och kryptonycklar. Bevis som man med traditionella metoder oftast inte kan finna. I vissa brottsutredningar kan datainsamlig i driftsatta datorsystem vara den enda tillämpbara metoden. Detta gäller särskilt i fall där tredje man riskerar att drabbas, exempelvis för datorsystem som är affärskritiska och vid husrannsakan på arbetsplatser.

Abstract

The landscape of digital investigations is shifting. Historically the crime investigation process has been restricted to data acquisition in hard drives of sized computers. Data in hard drives are non-volatile data (persistent data). Live forensics is the name of recent processes. Live forensics is data acquisitions of volatile memory in running (live) computer systems. The data acquired in RAM are volatile data. Normally volatile data are lost when power is removed. Volatile data can be relevant for crime investigations.

This research identifies criteria for software and processes that can be used to acquire volatile data in RAM in running computer systems. A software called DD can be used to perform the data acquisition. The process of acquiring data has some difficulties. The software acquisition tool has some limitations. For example DD needs administrator privileges in the operating system to be able to run. A dilemma concerning the data acquisition process is that data stored in the hard drive will be altered. Documentation that identifies which files will be altered is needed. Software has dependencies to the target operating system and the kernel. A risk with data acquisition is that the kernel in the target computer can contain rootkits. The untrusted kernel can produce unreliable data, data with low data quality. Unfortunately dependency to the kernel can never be removed using software solutions.

By using software acquisition tools at least one process needs to be run in the target computer system. This process will overwrite unallocated data in RAM. Therefore software should be constructed to demand as little memory as possible in the target system. That is an important criterion. The capability of being able to review the source code of the software is another criterion. By performing the review the software can be verified to ensure that the result produced by the software satisfies all requirements.

The storage capacity of hard drives is large and continues to grow. To analyze all data stored in a hard drive within a reasonable time is unfeasible. By analyzing the data acquired in RAM investigators can get guidance where to search for relevant data in hard drives. A digital investigation can thereby be performed in less time and be more efficient. Information analysis of acquired data can result in evidence such as running processes, chat messages and cryptographic keys. Evidence that is unlikely to be found using traditional methods. In some cases live forensics can be the only valid method to use. Especially when a third party risks economic loss, for example searches in business critical computer systems and workplace searches.

The thesis is written in Swedish.
This thesis corresponds to 20 weeks of full-time work.

Förord

Jag vill rikta stort tack till Peter Ericsson och alla inblandade personer på Rikskriminalpolisens IT-brottssektion som visat intresse och engagemang i mitt exjobb. Jag vill även tacka Håkan Rosvall på internationella åklagarkammaren för att ha givit mig värdefulla insikter.

Jag vill också tacka min handledare Louise Yngström som presenterat många infallsvinklar och bidragit med ett stort stöd och engagemang.

Innehållsförteckning

Figurförteckning

1. Inledning

Denna uppsats är del av examensarbete inom civilingenjörsutbildning i informationsteknik vid Kungliga Tekniska högskolan (KTH). Examensarbetet genomförs på uppdrag av Rikskriminalpolisens IT-brottssektion.

Detta kapitel beskriver bakgrund och problemformulering. Mål, syfte, avgränsningar samt metodsammanfattning presenteras även. Förväntade resultat och uppsatsens disposition beskrivs slutligen.

1.1. Bakgrund

Inom området för kriminaltekniska undersökningar riktas för närvarande uppmärksamhet mot programvara och processer som möjliggör datainsamling i driftsatta datorsystem (s.k. live computer forensics). Utmaningen kan beskrivas som insamling och analys av data från datorer som är i drift. Digitala bevis i form av kryptonycklar, processer och nätverkskommunikation kan vara exempel på möjliga resultat. Undersökningar av datorsystem i drift sker under andra förutsättningar än traditionell datainsamling av datorsystem. Programvara och processer inom live computer forensics är under utveckling och en studie inom området är därför angelägen.

Det finns ett flertal anledningar till att live forensics som undersökningsmetod har aktualiserats. Stor lagringskapacitet i hårddiskar har gjort traditionella undersökningsmetoder långsamma. Det är tidskrävande att göra en fullständig analys av en hårddisk på 300 GB. Tiden är en faktor som har betydelse. Live forensics är även intressant vid datainsamling i system som är affärskritiska/verksamhetskritiska. För vissa typer av brott är det inte proportionerligt att stänga av en server som tillhör ett företag, för att genomföra datainsamling. Att stänga ner ett system hos ett företag kan få ekonomiska förluster som konsekvens. Tredje man kan även drabbas av ekonomisk skada. I fall där brottets karaktär inte är tillräckligt allvarig för att stänga ner systemen kan live forensics vara en metod som passar bättre. Det är inte enbart dessa fakta som talar för live forensics som metod. Det främsta argumentet borde vara möjligheterna att genom live forensics finna bevis som inte går att upptäcka med traditionella metoder. En dator som är igång kan ge värdefulla bevis. Exempel på digitala bevis kan vara: processer som körs, öppna nätverksportar, lagrade kryptonycklar, data i ”urklipp”, e-postmeddelanden, chatmeddelanden, data från register, cache och RAM. Dessa data går vanligen förlorad om man stänger av datorn och genomför en traditionell undersökning. Live forensics är således ett naturligt led i utvecklingen av undersökningsmetoder för att möta kommande behov från rättsvårdande myndigheter och försvarsmakten (Panda B. et al. 2006).

Ambitionen med examensarbetet är att visa på möjligheterna med undersökningsmetoderna men även att redogöra för begränsningarna. Perspektivet i rapporten kommer att återspegla svenska kriminaltekniska undersökningar. Svensk nationell lag och rättspraxis kommer att vara utgångspunkt.

Behovet av examensarbetet är påtagligt då området är under snabb utveckling och relativt outforskat. Live forensics kan ge utredaren data som är relevant för en brottsutredning.

1.2. Problemformulering

Hur kan programvara användas för datainsamling av RAM i driftsatta datorsystem vid kriminaltekniska undersökningar?

1.3. Mål

Examensarbetets mål är att ta fram kriterier för val av programvara och processer som kan användas vid datainsamling av RAM i driftsatta datorsystem vid kriminaltekniska undersökningar.

1.4. Syfte

Syftet är att skapa en utvärderingsmodell som ska vara vägledande för val av programvara och processer för datainsamling av RAM i driftsatta datorsystem vid kriminaltekniska undersökningar.

1.5. Avgränsning

Perspektivet kommer att återspegla svenska kriminaltekniska undersökningar. Svensk nationell lag och rättspraxis kommer att vara utgångspunkt.

Windows XP SP2 är det operativsystem som ingår i undersökningen. Den främsta orsaken till valet av Windows XP som operativsystem är att det i dagsläget är det vanligaste förekommande operativsystemet. Programvara för datainsamling av RAM ska vara anpassade för att fungera i Windows XP. Undersökningen kommer att omfatta endast öppen programvara.

1.6. Målgrupp

Målgruppen är främst personal vid rättsvårdande myndigheter som är verksamma inom IT-forensik. Personal vid försvarsmakten samt universitet och högskolor inom området för IT- och datasäkerhet ingår även i målgruppen.

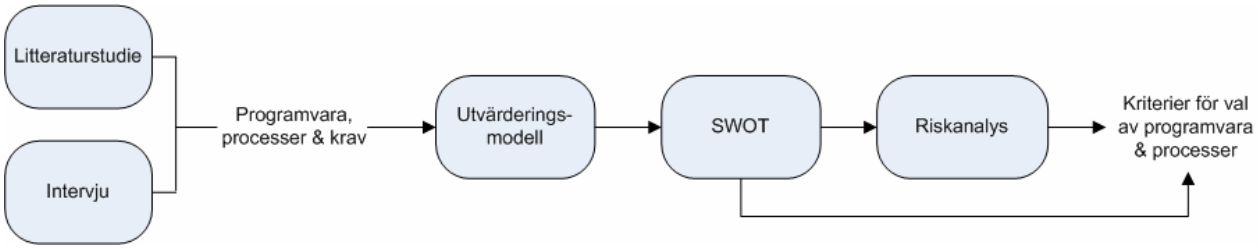
Läsaren förutsätts ha goda kunskaper inom datavetenskap i allmänhet och inom området för datasäkerhet i synnerhet.

1.7. Metodsammanfattning

Problemområdet live computer forensics är relativt nytt och outforskat, därför kommer ett explorativt och holistiskt angreppssätt att användas. Figur 1 visar en översiktlig modell över metoder och arbetsflöde.

Examensarbetet inleds med en litteraturstudie och faktainsamling inom området live computer forensics. Genom information från litteraturstudien identifieras programvara och processer som kan användas för att genomföra datainsamling i driftsatta datorsystem. Intervju med åklagare kommer även att genomföras. Information från intervjun kommer att används för att kartlägga de

juridiska aspekterna. Information från litteratur och intervju kommer att användas för att formulera de krav och risker som kan förknippas med programvara och processer. Den programvara som ingår i undersökningen kommer att väljas baserat på information från litteraturstudie och intervju. En utvärderingsmodell skapas, även den baserad på information från litteraturstudie och intervju. Programvara prövas mot kriterierna i utvärderingsmodellen och resultatet analyseras och sammanställs i en SWOT-analys. I riskanalysen undersöks resultatet från utvärderingen. Riskanalysen är en sammanställning över de faktorer som kan påverka datainsamling i driftsatta datorsystem. Resultatet av riskanalysen och utvärderingen formar gemensamt kriterier för val av programvara och processer. Framtida utvecklingsförslag inom området presenteras slutligen mot bakgrund av kriterierna.



Figur 1. Modell över metoder och arbetsflöde

Ansats

Undersökningen kommer att ha en induktiv ansats. Det innebär att data behandlas med teorier, begrepp och modeller. Arbetet går från det speciella till det generella (Rienecker & Stray Jørgensen, 2004). Alternativet vore att arbeta deduktivt. Det innebär att man drar slutsatser om enskilda företeelser, utifrån allmänna principer och befintliga teorier. Rienecker & Stray Jørgensen (2004) beskriver att deduktion utgår från en teori som man vill pröva, bekräfta eller avvisa på vissa data. En induktiv ansats har valts för att arbetssättet är mest lämpat för denna uppsats. Slutsatser kommer att dras utifrån information från litteraturstudie och intervju. Data kommer att behandlas med teorier, begrepp och modeller.

Beträffande utformningen av angreppssätt vid datainsamling och analys så kommer ett explorativt angreppssätt att väljas. Det är rimligt att använda sig av detta angreppssätt då området som ska kartläggas saknar systematisk forskning (Østbye H. et al. 2004).

Forskningsmetoden som används är kvalitativ. Dess syfte är främst att tolka och förstå insamlad data. Forskarens uppfattning eller tolkning av informationen är central för en kvalitativ forskningsmetod (Holme I.M. & Solvang B.K. 1997).

1.8. Förväntade resultat

Den programvara och de processer som identifieras innehåller troligen en rad begränsningar.
Exempel på en begränsning kan vara att program kräver administratörsrättigheter för att kunna användas. Denna begränsning blir ett problem vid exempelvis en husrannsakan om endast användarkonto (konto med begränsade rättigheter) är tillgängligt. Det kan innebära att programvaran i detta fall inte kan användas för datainsamling av RAM. Ytterligare en begränsning kan vara att datorn måste starta om för att programmet ska fungera. Det är en

begränsning i sammanhanget då data i RAM normalt går förlorad vid en omstart. De data som man önskar samla in går med andra ord förlorad.

Hur operativsystemet förändras och påverkas kommer troligen att variera vid användandet av olika programvaror. Även riskerna förknippade med användandet förväntas variera beroende på val av programvara och process.

Tillförlitigheten av resultaten kommer till stor del bero på tillförlitligheten hos källorna (litteratur och intervju). Litteraturen utgörs främst av artiklar publicerade i de vetenskapliga tidskrifterna Digital Investigation och Journal of Digital Forensic Practice. Artikelförfattarna är ofta väletablerade och erfarna inom området IT-forensik. Det finns även ett granskningsförfarande innan artiklarna blir publicerade. Generellt är det tillförlitigt att utnyttja vetenskapliga tidskrifter som källor (Nyberg R. 2000).

Tillförlitigheten av resultaten kommer även att bero på möjlighet till granskning av program som ingår i undersökningen. För öppen programvara är vanligen källkoden tillgänglig vilket ger granskningsmöjlighet. Programvaran är dock beroende av ett operativsystem för att fungera. En minskning av resultatets tillförlitighet kan bero på begränsade möjligheter till granskning av operativsystemet; Windows XP är inte ett öppet operativsystem. Bristfällig eller avsaknad av dokumentation av programvara och operativsystem kan bidra till minskning av tillförlitighet för resultatet.

1.9. Disposition

- Kapitel ett beskriver bakgrund och problemformulering. Mål, syfte, avgränsningar och metodsammanfattning ingår även i kapitel ett.
- Kapitel två omfattar den teoretiska bakgrunden, terminologi och en sammanställning av litteraturstudien. Minneshantering och risker kommer att beskrivas i kapitlet.
- Kapitel tre omfattar en utvärderingsmodell samt krav.
- Kapitel fyra innehåller funktionsbeskrivning av programvara och processer.
- Kapitel fem innehåller SWOT-analys där programvara utvärderas. I kapitel fem ingår även en riskanalys.
- Kapitel sex omfattar slutsatser, diskussion och framtida forskningsförslag. Slutligen finner läsaren en fullständig referenslista. Intervju, och programmanual återfinns i bilaga.

2. Teoretisk bakgrund

Detta kapitel avser att ge läsaren den teoretiska bakgrund som krävs för att förstå möjligheter men även problem vid IT-forensiska undersökningar av driftsatta datorsystem. Definitioner presenteras inledningsvis. En sammanfattning av tidigare arbeten och forskning inom området presenteras. Teori kring minneshantering och identifierade risker i samband med datainsamling av driftsatta datorsystem ingår i kapitlet.

2.1. Definitioner

I dagsläget har samfund och yrkesverksamma inom datasäkerhetsområdet inte lyckats enas kring de termer som berör verksamheten computer forensics. Flertalet termer saknar alltjämt tolkning och har ännu ingen allmän spridning. Denna oenighet och avsaknaden av exakta och vedertagna definitioner utgör ett problem. Detta avsnitt är en sammanställning över den terminologi som kommer att användas i denna uppsats och även utgöra grunden för fortsatta resonemang. En sammanställning av de viktigaste definitionerna återfinns i en ordlista i slutet av uppsatsen.

Terminologin i denna uppsats är främst baserad på SIS HB 550 (2003): *Terminologi för informationssäkerhet*, samt Internet Security Glossary: RFC 2828.

2.1.1. Digitala bevis

Digitala bevis är en svårdefinierad term. Ett förslag till definition av digitala bevis är: ”Information of probative value stored or transmitted in digital form” (SWGDE 2000). Definitionen är vid i sin betydelse. Den anger dock att informationen ska ha ett juridiskt testamenterat värde (probative value). Inom svensk rätt tillämpas fri bevisprövning vilket gör att man inte bör tolka betydelsen, juridiskt testamenterat värde, alltför strikt. I rättens fria bevisprövning ingår även en värdering av relevansen för varje bevis.

Casey E. (2004a) ger en annan definition på digitalt bevis: ”Encompasses any and all digital data that can establish that a crime has been committed or can provide a link between a crime and its victim or a crime and its perpetrator.” Fritt översatt till svenska: digitalt bevis omfattar någon eller all digital data som kan fastställa att brott har begåtts eller kan ge en koppling mellan ett brott och dess offer eller ett brott och dess gärningsman. Denna svenska översättning av Caseys definition kommer att användas i uppsatsen.

Termen bevis kommer i fortsättningen att användas synonymt med digitala bevis.

2.1.2. Data och information

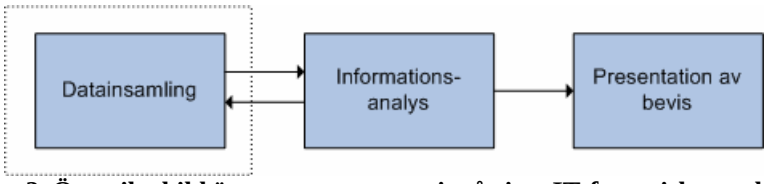
Termerna data och information kräver en närmare förklaring. Dessa termer förväxlas ofta i vardagligt tal och det kan leda till missförstånd.

Definitionerna från SIS HB 550 (2003) kommer att användas för att definiera termerna data och information.

Data definieras som representation av fakta, begrepp eller instruktioner i form lämpad för överföring, tolkning, eller bearbetning av människor eller av automatiska hjälpmedel.

Information definieras som innebörd av data. Det finns ett krav att data måste tolkas för att information ska erhållas.

Data och information i den kriminaltekniska utredningen
Betrakta Figur 2. Denna figur är en översiktsbild över de processer som ingår i en IT-forensisk utredning. Figuren är medvetet gjord som en förenkling, detaljer ingår inte i figuren.



Figur 2. Översiktsbild över processer som ingår i en IT-forensisk utredning

Datainsamling är den första processen i Figur 2. I processen datainsamling så samlas data in från exempelvis ett datorsystem. Programvara för att undanspara data i RAM kan användas som ett exempel. RAM innehåller data. I andra steget, informationsanalys, så tolkas data till information. Data ges en innebörd genom tolkningen som sker i informationsanalyssteget. Slutligen presenteras utvalda delar av informationen som bevis. Denna presentation kan exempelvis vara delar av ett förundersökningsprotokoll. Värt att notera är att processer i en utredning inte behöver ske sekventiellt (Casey 2004a). Det är med andra ord möjligt att från informationsanalysprocessen återvända till datainsamlingen för att få mer data, som sedan kan analyseras (se de dubbelriktade pilarna i Figur 2). Det icke-sekventiella arbetssättet är vanligare i praktiken.

Termen datainsamling kommer att användas istället för termen bevissäkring. Insamling av data implicerar inte att det behöver vara insamling av bevis. Bevis är exempel på information som tolkas från data (data med koppling till brott, gärningsman eller offer, jämför digitala bevis).

2.1.3. Data- och informationskvalitet

Datakvalitet har följande definition i SIS HB 550: egenskap hos data som behandlas enligt specifikation och inte ändrats eller påverkats av fel i databehandlingen.

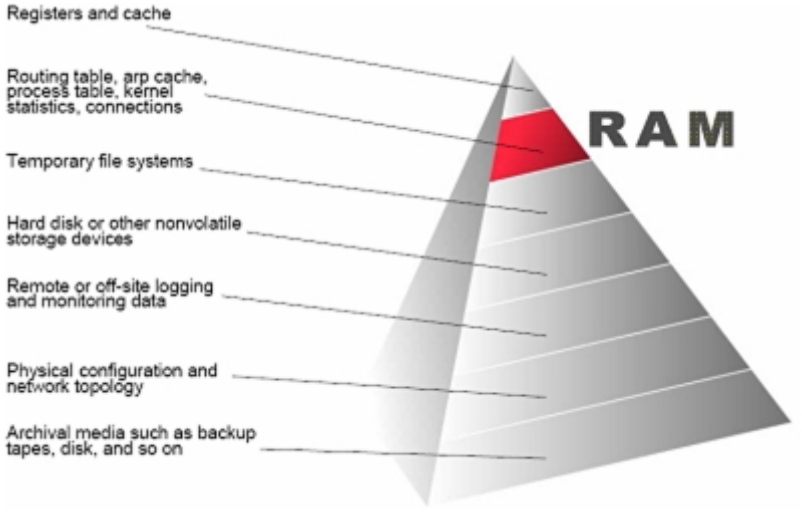
”Datakvalitet är relaterad till representation av fakta och informationskvalitet är relaterad till tolkningen av innebörden av data. Datakvalitet berör definitioner, detaljeringsgrad samt i vilken utsträckning data är korrekta. Data är kopplad till uppgiftens validitet” (SIS HB 550).

Informationskvalitet relaterar till digitala bevis. Informationskvalitet ser nämligen till meningsinnehåll för data och är beroende av användarens tolkning av data samt vad användaren ska använda data till. Användare i detta sammanhang är en IT-forensiker och data används som input till informationsanalysen för att hitta bevis som är av värde för en brottsutredning. Informationskvalitet berör uppgiftens relevans. Relevansen beskriver om data är ändamålsenliga dvs passar eller är användbara till det problem man vill lösa (SIS HB 550).

2.2. Vad är Live Forensics?

Den eviga frågan i Live Forensics sammanhang tycks vara huruvida man ska ”rycka sladden” eller inte. Anledningen att stänga av datorn är att data bevaras i ett oförändrat tillstånd, eller snarare att integriteten av data i datorns hårddisk bevaras. Att bevisen förblir oförändrande är en viktig egenskap när de presenteras i rätten. Ett bevis tillskrivs högre bevisvärde om det hanterats på ett sätt som bevisat är oförändrat. Det finns därför all anledning ur kriminalteknisk synvinkel att hitta processer och verktyg som säkerställer bevisens integritet vid datainsamling och informationsanalys.

Att ”rycka sladden” för att skydda data lagrat i hårddisk blir på bekostnad av att relevant data i exempelvis datorns RAM går förlorat. Om strömförsörjningen till en dator bryts så försvinner vanligen de data som kallas *volatile data* (flyktig eller instabil data).



Figur 3. Nivåer av flyktig data (volatile data) (Carnegie Mellon University 2006)

Live Forensics är processer och verktyg som möjliggör datainsamling från driftsatta datorsystem; datorsystem som är igång. Huvudfokus är datainsamling av flyktig data i datorns RAM (se det rödmarkerade området i Figur 3), men kan även omfatta datainsamling från systemregister och cacheminne. Figur 3 beskriver flyktighetsnivån hos data. Det förespråkas att datainsamling börjar med de data som har högst sannolikhet att förändras. Datainsamling bör därför initialt fokusera på den kategori av data som finns beskriven i pyramidens topp, för att sedan vandra ner i pyramiden under datainsamlingens gång.

2.3. Tidigare arbeten

Följande avsnitt är en kartläggning av aktuell forskning och resultat inom området Live Forensics. Det som redovisas är resultat som antas ha relevans för kriminaltekniska verksamheter som genomför datainsamling av driftsatta datorsystem. De flesta av de vetenskapliga artiklar som

refereras i denna uppsats är publicerade 2006. Litteraturstudien som helhet är fokuserad på de senaste rönen.

Andreas Schuster (2006) har skrivit en artikel med följande titel: *Searching for processes and threads in Microsoft Windows memory dumps*. Fokus i denna artikel är främst beskrivning av hur processer och trådar kan identifieras, efter det att datainsamling av RAM är gjord. Artikeln belyser problem med existerande verktyg som analyserar mönster. Verktygen har svårigheter att finna de processer och trådar som avslutats. Den sökteknik som presenteras av Schyster visar att processer och trådar under vissa omständigheter kan upptäckas trots att processerna vid datainsamlingen var avslutade. Möjlighet att finna processer och trådar trots att systemet har omstartats beskrivs även.

För denna uppsats är inledningen i Schusters artikel särskilt intressant. Författaren nämner kortfattat två olika sätt att genomföra datainsamling av RAM, nämligen genom programvaran DD (Garner, 2004) och en metod som kallas ”BugCheck trap” eller CrashOnCtrlScroll. Benämningen CrashOnCtrlScroll kommer fortsättningsvis att användas. Anledningen är att den benämningen tycks mer vanligt förekommande och missförstånd och förvirring kan därför undvikas. Garners programvara DD kommer att benämnas DD i fortsättningen. I Windows möjliggör programvaran DD kopiering av \Device\PhysicalMemory till en fil. \Device\PhysicalMemory är ett så kallat section object även kallat file-mapping object. Detta objekt kan användas av vissa applikationer för att få åtkomst till RAM (Microsoft TechNet 2006). Genom att koppla detta section object till den virtuella adressrymden så kan DD undanspara RAM till en målfil. Metoden är både populär och rekommenderad. Den främsta fördelen med DD är att programmet skapar en målfil som har ett enkelt format. *”The file offset equals the absolute adress in physical memory”* nämner Schuster. Av säkerhetsskäl så är åtkomsten till \Device\PhysicalMemory blockerad för vanliga användarkonton i Windows XP. Programvaran DD kräver med andra ord administratörsrättigheter för att fungera.

Den andra metoden för datainsamling av RAM i driftsatta datorsystem som beskrivs av Schuster kallas CrashOnCtrlScroll. Metoden använder ett dokumenterat förfarande där man ändrar värdet i en registernyckel[1]. När man sedan trycker en tangentsekvens (Ctrl + två gånger på scrollLock) orsakas en önskad systemkrasch. Resultatet av en CrashOnCtrlScroll är att en minnesdump skapas som en fil på datorns hårddisk. Data i RAM kommer alltså orsakat av systemkraschen undansparas i en fil i datorns hårddisk som sedan kan kopieras för vidare informationsanalys. Schuster skriver att den främsta nackdelen med detta förfarande är det komplexa formatet på filen som undansparas. CrashOnCtrlScroll verkar dessutom utelämna vissa delar av data i RAM vilket inte är önskvärt. En ytterligare nackdel är att filen sparas på datorns hårddisk, vilket medför risken att annan värdefull bevisning på hårddisken går förlorad. Att förändra data i måldatorns hårddisk strider mot vedertagna IT-forensiska undersökningsprinciper. Till fördelarna med att använda CrashOnCtrlScroll är att filens komplexa format gör den läsbar i Microsofts debugger. Informationsanalys av komplexa datastrukturer i kärnan kan lättare genomföras. Farmer och Venema (2004) nämner en annan viktig fördel. Efter det att registernyckeln är ändrad kan vem som helst som har tillgång till tangentbordet aktivera CrashOnCtrlScroll oavsett om någon användare är påloggad i systemet eller ej. Administratörsrättigheter krävs alltså inte för att

[1] Metoden finns mer utförligt beskriven i Microsoft Knowledge Base artikel 254649.

aktivera den önskade systemkraschen. Administratörsrättigheter krävs dock för att ändra värdet i registernyckeln.

Mike Dickson (2006) har skrivet en artikel med titeln: *An examination into MSN Messenger 7.5 contact identification*. Denna artikel presenterar metoder för att bevisa att kontakt har etablerats mellan en misstänkt gärningsman och ett offer, via chatprogrammet MSN. Artikeln fokuserar i första hand på förutsättningar enligt brittisk lag men innehåller även delar som är relevanta ur ett svenskt perspektiv. Artikeln bygger på ett scenario kring brottstypen grooming[2], men man kan även tänka sig andra typer av brott där undersökningsmetoderna blir användbara. Artikeln ger en handfast metod för att avgöra om en misstänkt gärningsman varit i kontakt med ett offer via chatprogrammet MSN. Det är inte bara information om att parterna har varit i kontakt med varandra som kan bevisas, utan minst lika viktigt att det inte är någon tredje part som har tagit kontroll över den misstänkta gärningsmannens konto.

Dickson presenterar en intressant observation: *”Conversations are not logged anywhere in MSN other than within the designated chat log file, if such use is elected. However, entire conversations can appear in order, in Unicode, in RAM”*. Detta resultat borde vara relevant för kriminaltekniska undersökningar. Resultatet visar på en i raden av möjligheter att genom datainsamling av RAM finna bevis som är relevanta för en brottsutredning.

Beträffande programvara för datainsamling av RAM nämner Dickson kortfattat: *”There are hardly any tools available for Windows that will successflully forensically image RAM, and of course there are issues with the imaging program writing into RAM itself, something that can hardly be helped.*” Genom att välja programvara som påverkar RAM i så liten utsträckning som möjligt kan effekten av dessa problem minimeras, nämner Dickson. Vidare föreslår Dickson att programvaran DD kan användas för datainsamling av RAM.

Dickson beskriver i artikeln hur han använder programvaran DD för datainsamling av RAM: *”The command line used to image the RAM was dd if=\\.\PhysicalMemory of=memory.dump conv=noerror bs=8192”* Processen får följande resultat: *”This latter process reveald several items of interest, not least of which is the presence of all the incoming messages embedded in RAM…”*.

Ytterligare en artikel som bör uppmärksammas är skriven av Brian D. Carrier och Joe Grand (2003) med titeln: *A hardware-based memory acquisition procedure for digital investigations*. Artikeln beskriver en metod för datainsamling av RAM som bygger på en hårdvarulösning. Hårdvaran som utvecklats kallas Tribble. Den består av ett PCI-kort som installeras i datorn innan en incident inträffar. När kortet aktiveras genom en strömbrytare så stannar datorns CPU. Kortet använder DMA (Direct Memory Access) för att kopiera data i RAM till ett externt lagringsmedia, exempelvis en Firewire hårddisk eller minneskort. Då kopiering av RAM avslutats så startar datorns CPU och operativsystemet fortsätter att köra. En mer ingående beskrivning av Tribble kommer inte att presenteras då hårdvarulösningar inte omfattas av denna uppsats. Det som däremot är relevant att redovisa är de argument som ges, programvara kontra

[2] I brist på svenskt begrepp används grooming som kan definieras som vuxna som söker kontakt med barn över internet i sexuellt syfte.

hårdvarulösningar, vid datainsamling av RAM. Carrier och Grand ger följande argument **mot** att använda programvara för datainsamling av RAM:

- *”Rootkits and Trojan horse attacks against applications and operating system kernels can cause the system to produce unreliable data, so it is desirable not to rely on the resources that the attacker could have modified.”*

  Här belyser författarna det dilemma som möter IT-forensiker vid datainsamling av driftsatta datorsystem. Data som lagras i RAM går förlorat om strömmen bryts. En IT-forensiker har inte möjlighet att genomföra en datainsamling av samma data i en kontrollerad miljö fri från skadlig kod. Risker finns att kärnan levererar otillförlitlig data.

- *”Existing methods for acquiring volatile memory involve untrusted software and are invasive because they typically write back to memory and may also write to the system´s hard disk.”*

  Detta påstående tycks vara vanligt förekommande och illustrerar problematiken med programvara för datainsamling av RAM på ett bra sätt. Att programvaran vid användandet själv skriver till minnet (RAM) är ett problem. Att programvaran kan påverka datorns hårddisk är ett ytterligare problem. Dessa aspekter kommer att belysas ytterligare.

- *”The user space applications that are used in the acquisition can be run from a trusted device (such as a CD-ROM), but the kernel is always needed to extract the data from memory. There is no way to avoid using the kernel with a software solution because it controls the scheduling of access to the processor and controls all the data flow to the storage locations.” … ”Using an untrusted kernel with software acquisition tools decreases the reliability of the evidence.”*

  Tillförlitigheten hos de data som insamlats kommer till stor del bero på vilken nivå av tillförlitighet man kan tillskriva operativsystemets kärna, skriver Carrier och Grand.

Carrier och Grand ger följande argument **för** användandet av deras föreslagna hårdvarubaserade lösning (Tribble), för datainsamling av RAM:

- Tribble får åtkomst till RAM utan att använda operativsystemet.
- Tribble kan genomföra datainsamling av RAM utan att själv skriva till minnet som ska undansparas.

Carrier och Grand ger följande argument **mot** användandet av deras föreslagna hårdvarubaserade lösning (Tribble) för datainsamling av RAM:

- Tribble måste installeras innan själva incidenten inträffar.
- Tribble är svår att testa och utvärdera för användarna. Data i minnet som man vill undanspara ändras kontinuerligt och det är svårt att avgöra om en exakt kopia har undansparats.
- Det är svårt för användare att avgöra om data har skrivits eller ej till det minne som man vill undanspara.

Carrier och Grand ger slutligen exempel på programvara som kan användas för datainsamling av RAM. Exemplet som nämns är ännu en gång programvaran DD.

Falk & Lindblom (2005) har skrivit en magisteruppsats med titeln: *Loggar som bevisning.* I uppsatsen så belyser författarna bland annat problematik kring att loggar som bevis sällan ifrågasätts av rätten. En genomgång av tidigare rättsfall visar att loggar som bevis tillskrivs ett högt bevisvärde. Orsakerna till att riktigheten hos loggarna sällan ifrågasätts kan vara många. Bristande kunskap kring felkällor i loggarnas skapande och hantering är en förklaring som författarna nämner. Falk & Lindbloms uppsats utgår från ett svenskt perspektiv.

En amerikansk studie, nyligen publicerad av Adams J. et.al. (2006), *Gap Analysis: Judicial Experience and Perception of Electronic Evidence*, belyser samma problematik som Falk & Lindblom presenterade, nämligen problemet med bristande IT-kunskap i rätten. I studien av Adamas J. et.al. har domare svarat på en enkätundersökning som innehöll frågor om deras kunskaper inom IT. Studien ger exempel på att digitala bevis tillskrivs högt bevisvärde. Riktigheten hos digitala bevis ifrågasätts aldrig eller mycket sällan och det tycks bero på okunskap om vilka felkällorna är. Förekomsten av digitala bevis i rätten förväntans att öka.

Förundersökning
Förundersökningen i mål 1581/05, brottsrubricering: dataintrång har studerats i samband med litteraturstudien. I förundersökningen beskrivs en genomförd husrannsakan. I protokollet för husrannsakan är det dokumenterat att en driftsatt dator påträffats. I protokollet står beskrivet vilka kommandon som användes vid undersökning av den driftsatta datorn. Vidare så står det angivet att husrannsakan videofilmades. Protokollet ger information om vilken dokumentation som förekommer i samband med husrannsakan, samt illustrerar hur IT-forensiker kan hantera driftsatta datorer som påträffas.

2.4. Allmänna principer för minneshantering

I grafikorienterade operativsystem (exempelvis Windows XP) så saknas ibland tillräckligt minne för att hantera alla aktiva processer. Processer som inte får plats i minnet måste lagras på disk och föras in dynamiskt för att köras. Två strategier finns för att lösa problematiken med minne som blir fullt, nämligen swapping och virtuellt minne (Tanenbaum & Woodhull, 1997). Detta avsnitt är en kort sammanfattning av minneshantering. Avsikten med avsnittet är att redovisa allmänna principer för minneshantering och beskriva de begrepp som används i sammanhanget. Principer för minneshantering kommer att påverka arbetet då IT-forensiker genomför datainsamling av driftsatta datorsystem.

2.4.1. Virtuellt minne

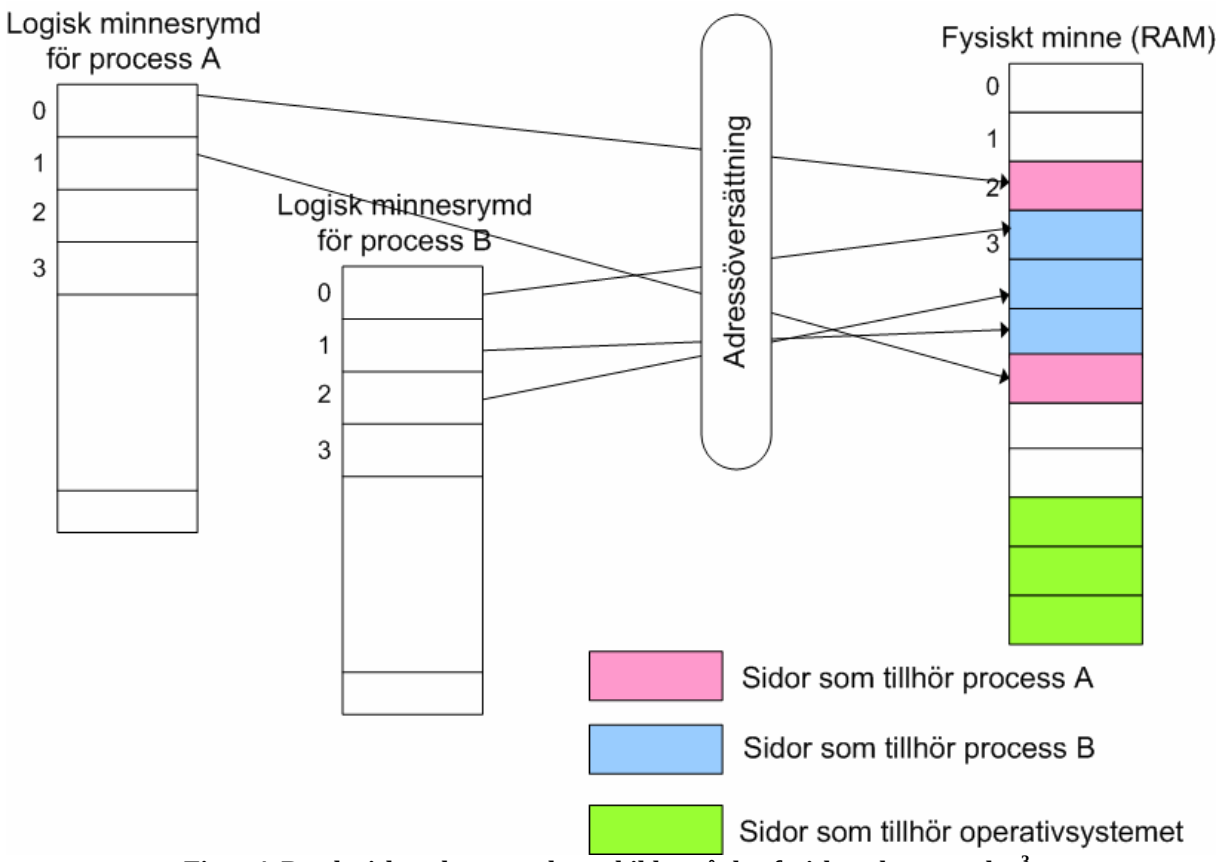
Virtuellt minne är en strategi som tillåter processer att köras trots att de bara delvis finns tillgängliga i minnet. Grundidén med virtuellt minne är att den sammanlagda storleken på ett program, data och stack kan överstiga den mängd primärminne (RAM) som finns tillgängligt i datorn. Operativsystemet behåller de delar av programmet som används för stunden i primärminnet och resten på disk. Som exempel kan ett 32 Mbyte program köras på en dator med 16 Mbyte kapacitet på primärminnet. Detta genom att noggrant välja vilka 16 Mbyte som ska ingå i minnet för varje instans och vilka delar av programmet som ska flyttas mellan disk och minne efter behov.

Virtuellt minne fungerar så att varje process kommer att tilldelas en logisk adressrymd av operativsystemet. Dessa adresser kallas logiska adresser eller virtuella adresser. Adresserna översätts till verkliga adresser i fysiska minnet när processen körs, Figur 4 illustrerar denna adressöversättning. Man skiljer den virtuella adressrymden från adressrymden i primärminnet och man brukar därför kalla primärminnet för det fysiska minnet (physical memory) eller RAM (random access memory). Det fysiska minnet är en eller flera minneskapslar som sitter monterade i datorn. Adresser i primärminnet som operativsystemet och maskinvaran använder kallas för fysiska adresser (Brorsson 1999).

Demand paging är namnet på en enkel metod för att implementera virtuellt minne (Tanenbaum & Woodhull 1997). Varje process har en logisk adressrymd som skapas som en fil i datorns hårddisk. Adressrymden är uppdelad i block som normalt är 4Kbyte. Blocken kallas för sidor (pages). Även primärminnet delas upp i block av samma storlek, dessa kallas för sidplatser (page frames). När en process ska köras så flyttar operativsystemet de sidor som hör till processen till lediga sidplatser i det fysiska minnet (se Figur 4). Det fysiska minnet i Figur 4 innehåller följande data och instruktioner: process A, process B samt delar operativsystemet (Brorsson 1999). Process A och process B arbetar med virtuella adresser, och alla minnesreferenser som görs måste översättas för att kunna nå rätt adress i det fysiska minnet. Operativsystemet däremot har stora delar som arbetar med det fysiska minnet direkt och en adressöversättning behövs inte för dessa delar. Resultatet av att varje process får en logisk adressrymd är att det totala antalet sidor som kan adresseras kan bli fler än antalet sidplatser i det fysiska minnet. För ett 32-bits system kan den totala storleken på virtuellt minne vara 2^{32} vilket motsvarar ungefär 4 Gbyte.

Brorsson (1999) skriver att det i dagsläget är så pass billigt med minne och att de har en hög packningstäthet att skyddsmekanismen som virtuellt minne ger är den främsta fördelen.

Principen för själva adressöversättningen kommer inte att förklaras närmare. Anledningen är att den inte bedöms nödvändig för att läsaren ska förstå det fortsatta resonemanget i uppsatsen. Läsare som däremot önskar fördjupa sig i minneshantering uppmanas att studera böckerna av Brorsson och Tanenbaum & Woodhull (se referenslistan).



Figur 4. Den logiska adressrymden avbildas på den fysiska adressrymden[3].

2.4.2. Swapping

Swapping är den enklaste strategin att ta till när ett minne blir fullt. Den går ut på att föra in varje process i sin helhet, köra den en stund, och sedan föra den tillbaka till swap space som är en fil i datorns hårddisk.

I takt med att priset på minne sjunker är det numera vanligt med stor minneskapacitet, även i datorer för privatpersoner. Datorer har ofta mer kapacitet i RAM än vad som behövs för generella applikationer. Behovet av swapping har på senaste tid minskat.

2.4.3. Lokalitetsprinciper

Referenslokalitet är ett begrepp som beskriver att en processor inte kommer att referera till alla minnesadresser med samma sannolikhet. Vissa minnesadresser kommer att refereras mycket oftare än andra. Små delmängder av den totala minnesrymden kommer att användas under långa tidsperioder.

[3] Bilden är hämtad från Brorsson M (1999). *Datorsystem Program- och maskinvara*. S 302. Bilden är omarbetad. Bilden har bland annat färglagts för att öka tydligheten.

Termen referenslokalitet kan delas upp i två kategorier: tidslokalitet (temporal locality) och rumslokalitet (spatial locality) Brorsson (1999).

Tidslokalitet betyder att de minnesceller som nyligen blivit refererade med stor sannolikhet kommer bli refererade igen. Brorsson (1999) nämner några förklaringar till detta fenomen. Tidslokalitet kan bero på programstrukturer och programslingor, repetitivt anropade subrutiner och att programmen återanvänder variabler.

Rumslokalitet betyder att minnesceller med adresser i närhet till adresser som nyligen har refererats med stor sannolikhet kommer att bli refererade även de. Brorsson (1999) beskriver att detta beror på att programinstruktioner är sekventiella samt att variabler som är associerade med varandra ofta ligger nära varandra i minnet.

Lokalitetsprinciper kan användas för optimering av datorsystem (Brorsson 1999).

2.4.4. Hur minneshantering påverkar datainsamling

Man bör som IT-forensiker vara medveten om hur minneshantering fungerar. Sättet som operativsystemet allokerar minnesresurser och data på är viktigt.

Denna undersökning fokuserar på data i RAM. Man bör vara medveten om att data i RAM inte ger en fullständig bild över samtliga applikationer och dess data. Som förklarats ovan använder operativsystem olika tekniker för att hantera minnet som en resurs när processer körs. Då man genomför en datainsamling av RAM får man en ögonblicksbild av de data som finns i RAM just vid datainsamlingstillfället. Det kan mycket väl vara så att RAM är fullt och processer finns i antingen virtuella minnet eller swap. Kanske hittar man endast delar av en process i RAM. Det är viktigt att känna till begränsningar som blir till följd av att enbart betrakta data i RAM. RAM innehåller förvisso data om processer och applikationer, men ger inte alltid en fullständig bild av vad som händer i datorn.

Det faktum att minne blir billigare och swapping sker i allt mindre utsträckning får konsekvenser i kriminaltekniska sammanhang. Nackdelen ur ett kriminaltekniskt perspektiv är att data i swap är oförändrade jämfört med data i RAM. Detta betyder att sannolikheten att påträffa användbar data i swap blir lägre. Fördelen blir att data som en gång har placerats i swap kan finnas kvar där under en längre tid. (Farmer & Venema 2005).

Kunskap om hur lokalitetsprinciper fungerar kan hjälpa IT-forensiker att påskynda arbetet med informationsanalys av stora mängder insamlad data. Om man väl påträffat relevant data så finns anledning att fortsätta undersökningen i adresser som är i närheten av dessa data, detta på grund av rumslokaliteten.

2.5. Risker med Live Forensics

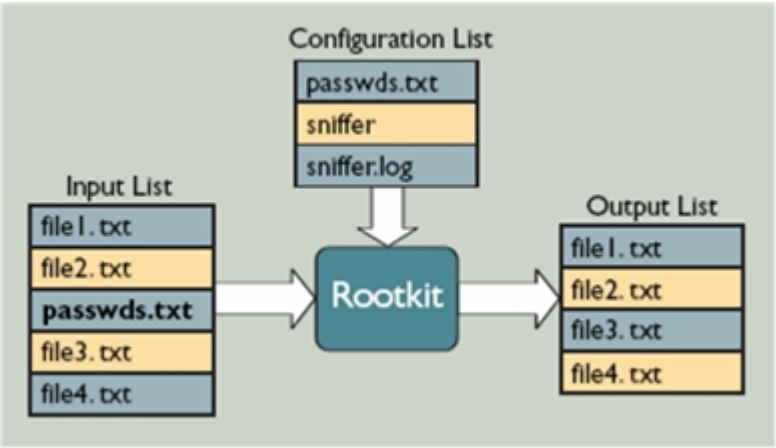
”Den enda skillnaden mellan levande och död analys är resultatens tillförlitlighet” (Carrier B. 2006). Vad är det då för typ av risker som påverkar tillförlitligheten hos data? Detta avsnitt

kommer att behandla de vanligast förekommande riskerna som en utredare bör ha kunskap om vid undersökningar av system i drift. Området risker, och hur dessa påverkar tillförlitigheten hos data, är ett komplicerat kapitel och svårt att sammanfatta. Målsättningen är inte att ge en uttömmande lista över riskerna utan snarare att belysa problematiken och ge exempel på processer och metodik för att minska skada av uppkomna risker.

2.5.1. Rootkits

Rootkits klassas som den vanligaste källan till vilseledande data vid analys av levande datorsystem (Carrier B. 2006). Förenklat kan rootkits beskrivas som ett program för datorintrång. De fungerar som en trojansk häst och modifierar det befintliga operativsystemet så att en hacker kan få åtkomst till en dator genom en bakdörr. Ett rootkit har förmågan att för användaren av måldatorn dölja förekomsten av själva attacken. Detta möjliggörs genom ett filter som placeras i dataflödet hos en måldator. Figur 5 beskriver situationen där ett rootkit används som ett filter. Filen ”passwds.txt” göms för användaren. Detta fenomen utgör ett problem vid undersökningar av datorsystem i drift. Problem uppstår då en utredare exempelvis vill lista innehållet av filer i en katalog. Utredaren använder en applikation som genom operativsystemet efterfrågar en lista. Listan passerar en rad mjukvarukomponenter innan den slutligen visas för användaren, och vid varje länk i denna kedja kan ett filter vara installerat som gömmer namn på den fil som kan innehålla relevant data. Filen med relevant data existerar i datorn men utredaren ser inte filen på grund av att ett rootkit har filtrerat bort namnet.

Rootkits kan utgöra ytterligare problem i samband med en brottsutredning. Det är tänkbart att en gärningsman medvetet installerar rootkits i förhoppningen att dessa kan fungera som alibi. Datakvaliteten kan påverkas om rootkits påträffas i måldatorn. Gärningsmannen kan hävda att insamlande data är otillförlitliga och har låg datakvalitet eftersom rootkits funnits i måldatorn. Detta kan utgöra ett problem vid en brottsutredning.

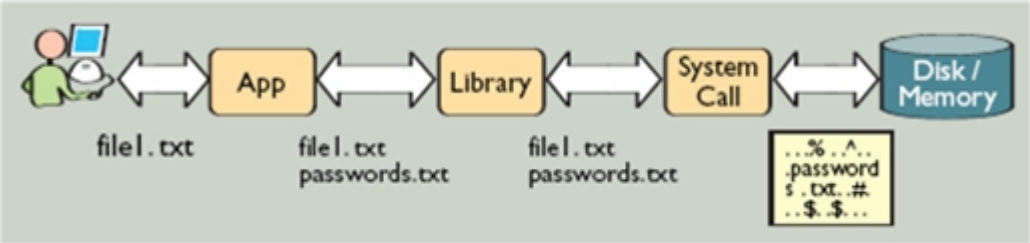


Figur 5[4]. Rootkits filtrerar allehanda data från olika källor och döljer data (potentiella bevis) för användaren (utredaren).

[4] (Carrier B. 2006)

Nedan följer en mer detaljerad beskrivning av olika tekniker som rootkits använder. Beskrivningen är i första hand tänkt att ge läsaren grundläggande information om hur rootkits fungerar.

Det finns ett flertal olika platser där en hacker kan installera rootkits. Figur 6 beskriver ett typiskt dataflöde i en dator. Användaren interagerar med en applikation som använder dynamiska bibliotek som är installerade i datorn. Applikationer eller bibliotek använder ett gränssnitt för att kommunicera med kärnan (kernel). Kärnan har direktåtkomst till minne, hårddisk och nätverkskommunikation. Varje länk: applikation, bibliotek (library) och systemanrop kan således omslutas av ett rootkit som fungerar som ett filter.



Figur 6[5]. Ett exempel av ett typiskt dataflöde i en dator. Bilden illustrerar här ett rootkit installerat i applikationsnivån, ty filen ”passwords.txt” filtreras i just applikationslagret.

Rootkits kan enligt ovanstående resonemang delas in i tre kategorier: application-level rootkits, library rootkits och kernel rootkits. En kort beskrivning följer för var och en av dessa.

Application-level rootkits – Detta är den mest primitiva varianten av rootkits. Ett systemprogram (en exekverbar systemfil) ersätts med en trojansk motsvarighet som för användaren kommer att dölja exempelvis fil- och processnamn, öppna nätverksportar och systemkonfigurationsvärden.

Application-level rootkits kan vara relativt enkla att avslöja. Flera applikationer kan oftast användas för att nå samma data vilket innebär att *alla* dessa applikationer måste ersättas med en trojansk motsvarighet. I ett Unix-system kan applikationerna: `ls`, `find` och `du` användas för att visa filnamn. Samtliga av dessa applikationer måste då ersättas för att kunna dölja filnamn i ett Unix-system om man använder application-level rootkits.

Library rootkits – En applikation kan vara dynamiskt kompilerad. Detta innebär att applikationen (programmet) utnyttjar ett gemensamt filbibliotek i datorn. Anledning till att man ofta vill kompilera en applikation dynamiskt är att själva programfilen blir mindre i storlek då den inte själv behöver innehålla alla nödvändiga filer. Ur lagringsmässig synvinkel är denna metod eftersträvansvärd ty det leder till effektivare utnyttjande av datorns hårddisk. Med dagens kapacitet på hårddiskarna borde emellertid denna vinst vara av marginell betydelse. Det finns dock andra fördelar med dynamisk kompilering. Det är enklare att uppdatera en fil i ett bibliotek istället för att behöva omkompilera alla program som innehåller den filen när en uppdatering genomförs. Till nackdelarna med dynamisk kompilering hör att biblioteken kan vara en måltavla för rootkits.

[5] (Carrier B. 2006)

Library rootkits angriper datorns bibliotek. Ett bibliotek kan omslutas av en trojansk motsvarighet som innehåller samma funktion som originalbiblioteket. När en applikation (vanligen dynamiskt kompilerad) anropar biblioteket (den trojanska motsvarigheten) anropar den i sin tur originalbiblioteket som utför systemanropet. Data som returneras av originalbiblioteket går genom den trojanska motsvarigheten där data filtreras innan data når användaren. Med library rootkits kan en hacker filtrera på ett ställe, nämligen biblioteket, och resultatet av filtreringen kommer att beröra alla applikationer som har beroenden (dynamiska beroenden) till biblioteket.

Kernel rootkits – Denna typ av rootkit angriper operativsystemets kärna (kernel). Egenskaperna påminner om det sätt som library rootkits fungerar på. Kärnan omsluts av en trojansk motsvarighet. Ett rootkit utför systemanropet och filtrerar sedan data som returneras.

Det finns i huvudsak två metoder att installera kernel rootkits. En metod är att utnyttja laddningsbara kärnmoduler eller drivrutiner för enheter som många operativsystem stödjer. Fördelen är att kärnan varken behöver omkompileras eller laddas om för att få effekt av åtgärden. Andra alternativet är att modifiera den version av kärnan som finns lagrad på hårddisken. Ett rootkit installeras då nästa gång systemet startas. Oavsett vilken av metoderna man väljer så är resultatet detsamma: relevant data kan existera i systemet men dessa data döljs för användaren (utredaren) genom filtrering.

2.5.2. Konflikt med minnestyper

Arne Vidström (2006) nämner olika metoder för datainsamling av RAM och konstaterar följande: *”In most practical cases we are left with no other option than doing a dump from the PhysicalMemory section object using DD from the Forensic Acquisition Utilities or similar.”* Programvara för datainsamling av RAM som beskrivs av Vidström är programvaran DD (Garner 2004).

Vidström beskriver ett problem som är relaterat till minneshantering och användandet av programvaran DD. Kortfattat kan nämnas att problemet kan uppstå om mer än en minnestyp används i en sidtabell som refererar till samma fysiska sida. Då DD körs så mappar man delar av minne som man själv inte ”äger” till den egna adressrymden. Den riktiga ”ägaren” av minnet kan i vilket ögonblick som helst ändra attribut på den fysiska sidan eller flytta sidan till hårddisk. Windows XP har en kontrollmekanism för att upptäcka inkompatibla attribut för sidor. I Windows XP kan detta medföra att man inte får åtkomst till vissa delar av fysiska minnet. De problem som kan uppstå är svårt att exakt förutspå eftersom de är beroende av vilken processordesign man betraktar, skriver Vidström. De effekter som beskrivits ovan finns även kortfattat återgivna i en readme-fil som ingår i programmet DD.

2.6. Data i RAM är inte så flyktig som vi tror

Data i de oallokerade delarna i RAM kan vara oförändrade under relativt lång tid. Uppgifter gör gällande att sidor i RAM kan förbli oförändrade i 13 timmar (Farmer & Venema, 2004), andra uppgifter beskriver att strukturer i driftsatta system kan vara oförändrade över 14 dagar(!)

(Schuster 2006). Hur lång tid data kan överleva i RAM beror på många faktorer exempelvis, hur stor minneskapaciteten är samt hur mycket minnesresurser som krävs för de program som körs.

Farmer & Venema (2004) presenterar en upptäckt, nämligen att data i RAM i vissa datorsystem kan finnas kvar trots att datorn har omstartas. De flesta datorsystem nollställer RAM vid omstart och detta tycks vara operativsystemsoberoende. Generellt verkar Intelprocessorer ha en instruktion i BIOS som nollställer minnet. Apple G4-datorer nollställer normalt inte minnet vid omstart. Farmer & Venema skriver att de flesta datorsystem kan konfigureras i BIOS för att nollställa RAM.

Tim Vidas (2006) skriver att strömmen kan brytas i 30 sekunder utan att riskera någon förändring av data i RAM. Sanningshalten i Tim Vidas påstående är oviss då inga test är genomförda som kan bekräfta uppgiften.

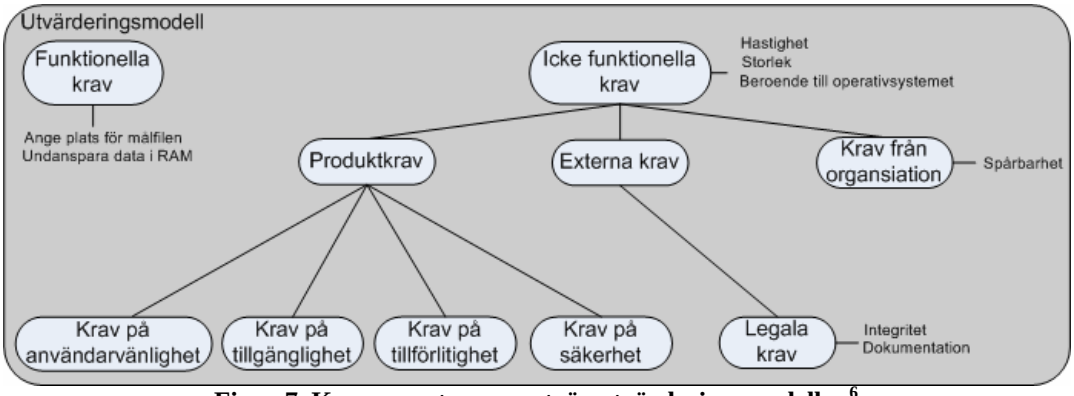
3. Utvärdering

I detta kapitel presenteras en utvärderingsmodell. De krav på programvara som har identifierats ingår även. I kapitlet beskrivs den kontext och de förutsättningar som programvara för datainsamling är tänkt att fungera i.

3.1. Utvärderingsmodell

Sommerville (2004) beskriver en testmetod som går under benämningen kravbaserade test. Metoden följer principen att varje krav på programvaran ska vara testbar. Kraven ska formuleras på ett sätt som möjliggör skapandet av test så att en utomstående observatör kan granska att kraven är uppfyllda. Kravbaserade test betraktas som validering snarare än felsökning. Målet med testmetoden är att visa att programvaran har implementerats så att kraven uppnåtts.

Nedanstående modell är skapad för att kunna analysera och utvärdera programvara för datainsamling av RAM i driftsatta datorsystem. Utvärderingsmodellen baseras på de krav som ställs på programvaran. Kraven är skapade med information från litteraturstudien och intervjuer. Figur 7 illustrerar de komponenter som utgör utvärderingsmodellen. Figur 7 kommer ursprungligen från Sommerville (2004) *Software Engineering.* Figuren är omarbetad till en utvärderingsmodell genom att införa grupperingar av krav som är relevanta för den programvaran som ska utvärderas. Kraven kan delas in i följande två huvudkategorier: funktionella krav och icke-funktionella krav.



Figur 7. Komponenter som utgör utvärderingsmodellen[6]

3.2. Krav

Funktionella krav beskriver vad en programvara ska göra, vilka tjänster en programvara ska erbjuda. Exempelvis att programvara ska kunna genomföra datainsamling av RAM. Funktionella krav beskriver vidare hur en programvara ska fungera givet en viss input.

Icke funktionella krav anger egenskaper hos programvara som relaterar till programvaran som helhet. Exempelvis de icke funktionella egenskaperna: tillgänglighet, tillförlitighet och säkerhet.

[6] Bilden kommer ursprungligen från Sommerville (2004) *Software Engineering* men är omarbetad. Specifika grupperingar av krav som är relevanta för den programvara som ska utvärderas har införts.

Krav på programvara beskrivs inom industrin på en mängd olika sätt. I somliga fall är kraven skrivna på en hög abstraktionsnivå och i andra fall utgörs kraven av formella beskrivningar av tjänster och funktioner hos programvaran. Krav används som verktyg för kommunikation med inblandade parter i utvecklingsprojekt och vid upphandling av programvara och system. Rollerna är olika vid exempelvis upphandling och implementation. Läsaren och användandet av kraven varierar och det är därför naturligt att kraven formuleras på olika sätt. Krav kan sorteras i två kategorier: användarkrav (user requirements) och systemkrav (system requirements) (Sommerville 2004).

Användarkrav beskrivs med en hög abstraktionsnivå. Dessa krav är oftast skrivna med vanligt språk och tillhörande diagram. Kraven anger främst vilka tjänster som förväntas ingå i programvaran och de begränsningar som påverkar tjänsterna.

Systemkrav beskriver ett systems funktion och tjänster på en detaljerad nivå. Kraven ska vara precisa och exakt ange vad som ska implementeras. Grafisk notation, matematiska specifikationer och designspråk används för att beskriva systemkrav.

3.2.1. Krav på programvara för datainsamling av RAM

I detta avsnitt återfinns de krav som kan ställas på programvara som har till uppgift att utföra datainsamling av RAM i datorsystem som kör Windows XP. Syftet med sammanställningen är att formulera en idé om vilka krav som bör prioriteras vid upphandling, utvärdering och utveckling av omnämnd programvara. Kraven som beskrivs utgör även huvudsakligen den utvärderingsmodell som beskrivits i tidigare avsnitt.

3.2.2. Funktionella krav

Funktionella krav	Beskrivning
Att utföra datainsamling av data i RAM	Förutsatt att datorn kör operativsystemet Windows XP SP2, samt att datorn är driftsatt (att operativsystemet är igång), ska programvaran kunna undanspara data lagrat i måldatorns RAM till en målfil.
Att välja lagringsplats för målfilen	Målfilen ska kunna lagras på externt portabelt media (exempelvis extern hårddisk eller USB-minne). Möjlighet att lagra data i måldatorns hårddisk ska även finnas.

3.2.3. Icke funktionella krav

Icke funktionella krav	Beskrivning
Programvaran ska vid exekvering utnyttja minimalt med utrymme i måldatorns RAM	Programvaran för att undanspara data i RAM kommer vid exekvering utnyttja (allokera) RAM i den måldator som är föremål för undersökningen. Programvaran ska vara konstruerad på ett sätt som gör att den inte utnyttjar mer RAM än nödvändigt för att uppfylla kraven.
Hastighet och effektivitet	Programvara ska kunna undanspara data i RAM i tillräckligt hög hastighet för att undvika att väsentliga delar av data i RAM förändras under den tiden som processen pågår.
Beroenden till operativsystemet	Programvaran ska vid användning vara konstruerad för att i största möjliga mån undvika att påverka filer i datorns operativsystem eller andra filer i måldatorns hårddisk.

Produktkrav	Beskrivning
Användarvänlighet	Programvaran ska uppfylla de förväntningar man kan ställa beträffande användarvänlighet.
Tillgänglighet	Programvaran ska ha hög tillgänglighet. Då processen att undanspara data i RAM efterfrågas ska processen påbörjas snarast.
Tillförlitighet	Programvaran ska leverera data av hög datakvalitet
Säkerhet	Tillräckliga skydd ska finnas i hanteringen av programvaran för att motverka risker i samband med insyn, förlust eller påverkan av de data som programvaran levererar.

Legala krav	Beskrivning
Dokumentation	Programvaran ska vara dokumenterad i sådan omfattning att den i detalj beskriver vilka filer i måldatorns operativsystem som påverkas vid användandet av programvaran. Dokumentationen ska beskriva under vilka förutsättningar programvaran kan användas och vilka begränsningar som programvaran har. De risker som är förknippade med användandet av programvaran ska även finnas dokumenterade.

Integritet	Data som levereras av programvaran ska hanteras på ett sätt som gör att integriteten av data bevaras och att data förblir oförändrad.
Förvaring	Bevismaterialet (data) ska förvaras i en miljö som är lämplig för att säkerställa dess integritet.

Krav från organisation	**Beskrivning**
Spårbarhet	Mekanismer för loggning i samband med datainsamling bör finnas i programvaran.

3.3. Mycket data och liten påverkan

Måldatorn kommer i någon form påverkas av den programvara som används för att undanspara minne; detta tycks oundvikligt. Det gäller emellertid att finna löningar och metoder som minimerar förändringar i måldatorn. Programvara för att genomföra datainsamling av RAM bör uppfylla följande kriterier:

Program bör vara litet i storlek – Ett program som tar mycket minne i anspråk riskerar att använda minnesutrymme som kan innehålla relevant data. Generellt brukar gälla att kommandobaserade program utan grafiskt användargränssnitt är relativt små i storlek, vilket är önskvärt.

Program bör vara effektivt beträffande hastigheten – Data i RAM är flyktiga vilket medför att hastigheten hos program som används för datainsamling blir viktig. Önskvärt är program som tidsmässigt är effektiva så att väsentliga delar av data i minnet förblir oförändrade under tiden då processen datainsamling pågår.

Program bör utformas så att fel undviks – Programvara kan vid fel instruktioner och misstag från användaren orsaka skada. Med god utformning av användarvänligheten kan program skapas på ett sätt som reducerar risken för fel vid användandet. Ett grafiskt användargränssitt kan exempelvis begränsa antal valmöjligheter att endast omfatta tillåtna inmatningsalternativ. Det grafiska gränssnittet är för användaren ett intuitivt och bekant sätt att interagera med programvara. Fördelarna med grafiken blir på bekostnad av att programmet växer i storlek och därmed riskerar att förändra relevant data i minnet.

3.4. Kontext

Detta avsnitt beskriver den kontext som programvara och processer är tänkta att fungera i. Kontext är en term som valts framför exempelvis termen miljö. Kontext är beskrivning av ett sammanhang (en omgivning) och ett tillstånd. Kontext är en abstrakt modell. För att begreppet ska vara användbart bör kontexten på förhand vara väldefinierad. Det ska gå att avgöra i tid och rum när kontexten är i fokus. I SIS HB 550 står: ”För att förstå data är dess ursprungliga kontext väsentligt.”

Grundscenariot är en husrannsakan. Varje husrannsakan är givetvis unik och variationen kan vara stor. Trots detta finns generella mönster och förutsättningar som är återkommande. Det är relevant att identifiera och beskriva dessa förutsättningar. Beskrivningar kan tjäna som stöd då programvara utvärderas eller utvecklas. En diskussion om risker i samband med användandet av programvara och processer kan föras mot bakgrund av dessa beskrivningar.

Beskrivningen av kontext utgår från följande komponenter: Teknik, roller, processer, organisation, information och system. Det finns anledning att kortfattat beskriva dessa komponenter.

Teknik
Vid en husrannsakan kan en dator påträffas. De tekniska förutsättningar hos den dator som är föremål för undersökning är exempelvis:

- Att datorn som undersöks har minst en tillgänglig USB-plats
- Att datorn har CD-ROM-läsare/DVD-läsare
- Att datorn använder operativsystem Windows XP
- Att datorn har tillgång till elektricitet

Roller
Roller avser inte specifika individer i kontexten. Att det är Erik eller Pelle som är IT-forensiker vid en husrannsakan är inte intressant i detta sammanhang. Det som är av intresse är att det finns en viss roll och vilka egenskaper som utmärker den rollen. Följande roller har identifierats:

- IT-forensiker som genomför datainsamling av RAM i måldatorn
- Åklagare som fattar beslut som husrannsakan
- En eller flera brottsmisstänkta personer

Processer
Processer beskriver en verksamhet. Det kan exempelvis vara villkor som ska vara uppfyllda för att en process ska påbörjas. Processer är en uppsättning instruktioner.

- Beslut om husrannsakan är taget
- Hur datainsamling av RAM ska genomföras
- Hur husrannsakan ska dokumenteras

Organisation
Det finns en eller flera organisationer som ingår i kontexten. Organisationer kan beskriva mål för en verksamhet.

- Polismyndighet
- Åklagarmyndighet

Information
En kontext innehåller information. Informationen i sammanhanget kan exempelvis vara:

- Kännedom om brottsmisstanke
- Kännedom om geografisk plats för husrannsakan
- Tolkning av data lagrat i RAM

System
System sammanfattar de tjänster som ska tillhandahållas. Den tjänst som är framträdande i kontexten är:

- Tjänsten att genomföra datainsamling av RAM i driftsatta datorsystem

4. Programvara och processer för att undanspara data i RAM

Detta avsnitt är en sammanställning och beskrivning av programvara och processer för datainsamling av RAM i Windows XP. Programvara som valts är DD. En alternativ metod, CrashOnCtrlScroll, som nämnts i avsnitt 2.3, kommer även kortfattat att beskrivas.

Programvaran DD har valts av flera anledningar. Den främsta anledningen är att programvaran har uppmärksammats och beskrivits i de flesta av de vetenskapliga artiklar som behandlar området live forensics. Vidare har DD i litteraturen beskrivits som den enda i nuläget rekommenderade programvaran för att genomföra datainsamling av RAM i Windows XP. DD är öppen programvara. Denna undersökning har som avgränsning att endast omfatta öppen programvara. Ytterligare skäl att DD har valts är att programvaran är baserad på en ursprunglig version som har använts under lång tid. Slutligen så har DD valts för att inget annat tillgängligt alternativ, varken genom information från intervju och litteraturstudie, har framkommit.

Metoden att undanspara data i RAM genom CrashOnCtrlScroll som nämnts i avsnitt 2.3 förtjänar viss uppmärksamhet. CrashOnCtrlScroll är förvisso ingen programvara, än mindre en öppen programvara. Trots detta finns fördelar som gör metoden relevant. Användare behöver inte vara påloggade i systemet för att data i RAM ska kunna undansparas.

4.1. Funktionsbeskrivning DD

I programsviten Forensic Acquisition Utilities (FAU) som är skapat av George M. Garner Jr. ingår programmet DD. Funktionsbeskrivningen utgår från version 1.2.0.2131 (beta2-RC5) av FAU. Programmet DD är en implementation som inspirerats av det ursprungliga GNU DD. Gleason BJ & Fahey D. (2006) ger en historisk skildring av uppkomsten och namngivningen av programmet DD. Förkortningen DD betyder ”Data Definition”. DD kan användas i IT-forensiska sammanhang för att ta spegelkopior av lagringsmedia. Det som gör DD relevant för live forensics är att programmet kan genomföra datainsamling av RAM i driftsatta datorer.

I Microsoft Windows XP finns en objektenhet (object device) \Device\PhysicalMemory som ger åtkomst till RAM. DD kan undanspara data från denna objektenhet.

4.1.1. Hur DD kan användas

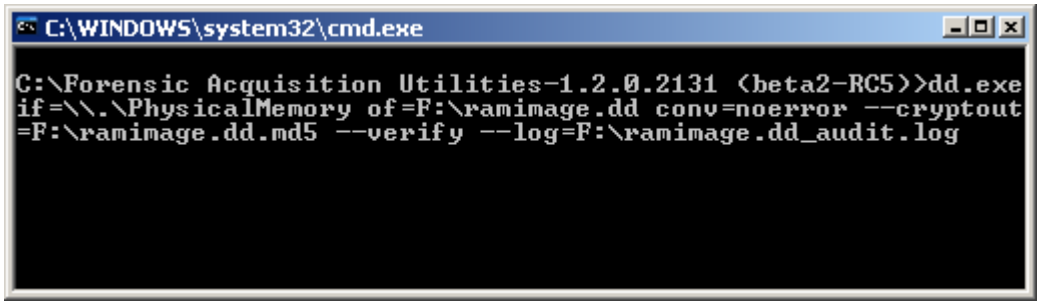
DD kan användas på två sätt: textbaserat genom en kommandotolk eller genom ett grafiskt användargränssnitt (GUI). Båda alternativen kommer att presenteras. I bilagan återfinns även ett utdrag från hjälpsidorna för DD som beskriver programmets olika inställningar och valmöjligheter.

DD genom kommandotolk
DD kan användas från en kommandotolk såsom cmd i Windows. Syntaxen för DD är uppbyggd enligt följande mönster: dd if=[SOURCE] of=[DESTINATION] [OPTIONS]

if= anger input som kan vara antingen en fil eller en enhet (hårddisk, diskett, band eller annan enhet såsom RAM). SOURCE (källfil) avser platsen för input.
of= anger målfil som är spegelkopia av källan. DESTINATION anger platsen för målfilen. Rekommenderat är att platsen för målfilen bör vara ett portabelt lagringsmedia utanför själva måldatorn, exempelvis ett USB-minne eller en extern hårddisk. Detta eftersom man inte vill förändra data i måldatorn.

OPTIONS ger möjligheten att ange en rad olika inställningsalternativ för DD. I bilagan finns en förteckning över dessa valmöjligheter.

För att genomföra datainsamling av RAM för Windows XP då datorn är i drift kan följande kodexempel som visas i Figur 8 anges i kommandotolken.



Figur 8. DD används textbaserat genom kommandotolken cmd

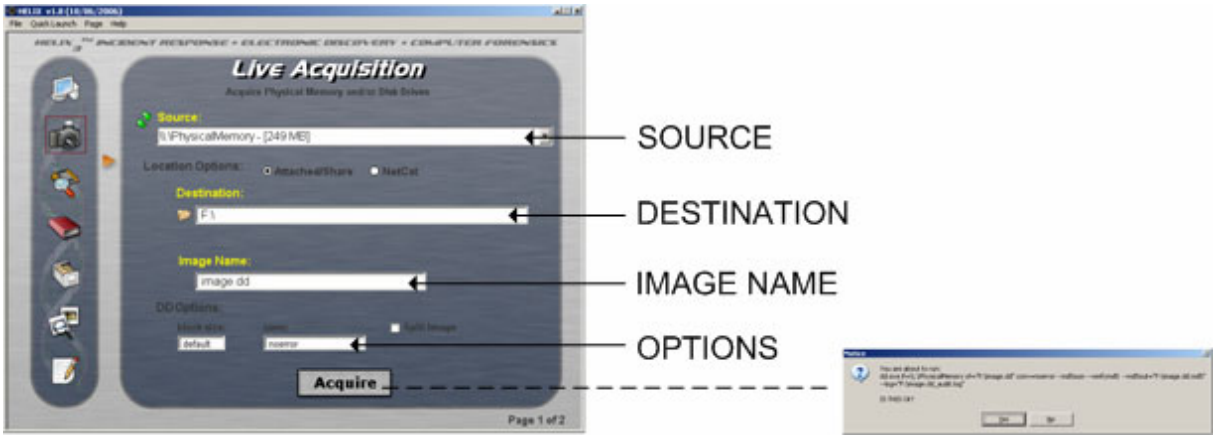
Kodexempel: dd.exe if=\\.\PhysicalMemory of=F:\ramimage.dd conv=noerror
--cryptout=F:\ramimage.dd.md5 --verify --log=F:\ramimage.dd_audit.log

I exemplet ovan är PhysicalMemory angivet som källfil, det innebär att källan som ska kopieras är RAM. Målfilen i exemplet är F:\ramimage.dd. Enheten F representerar i exemplet ett USB-minne som är anslutet till datorn. Inställningen conv=noerror anger att programmet ska fortsätta kopieringen om läsfel skulle uppstå. För att visa på några av möjligheterna med DD så är i kodexemplet ovan kommandot cryptout angivet och även kommandot verify. Detta innebär att en hashsumma beräknas av målfilen. Denna hashsumma verifieras sedan av DD. En logg i form av en textfil genereras och loggen sammanställer hela processen: att undanspara data i RAM, beräkna hashsumman och slutligen verifiera den. Tre filer blir resultatet (output) i detta exempel: själva spegelkopian av RAM (målfilen), hashsumman beräknad från spegelkopian, samt en loggfil som sammanställer processen.

DD genom ett grafiskt användargränssnitt
E-fense Inc. har sammanställt en CD-skiva som heter Helix. CD-skivan finns tillgänglig för nerladdning[7]. Skivan är en sammanställning av IT-forensisk programvara och programvara för incidenthantering. Skivan innehåller programvara för både Linux och Windows. Programmen i

[7] http://www.e-fense.com/helix/

Windowsmiljö kan nås genom ett GUI. DD har i Helix berikats med ett GUI. En miniatyrbild av detta GUI visas i Figur 9.



Figur 9. DD används grafiskt genom användargränssnittet i Helix

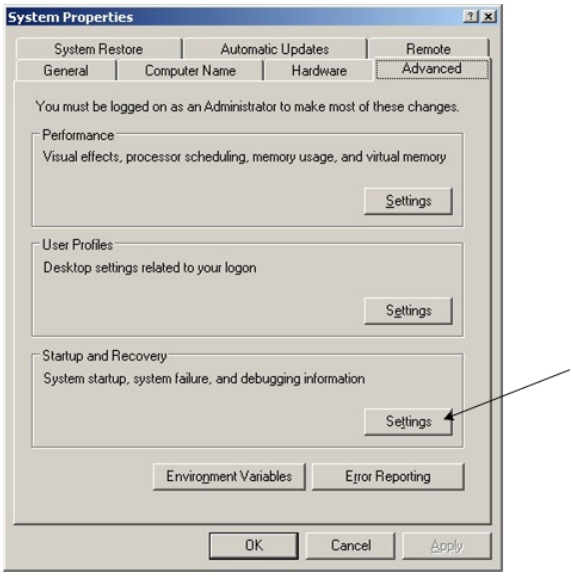
Användaren får ett grafiskt inmatningsformulär som skiljer sig något i jämförelse med det textbaserade sättet att använda DD. Inmatningsformuläret hjälper användaren att skapa den syntax som sedan används av programmet. Källan (source) anges genom att välja alternativ i en lista. De enheter som finns i datorn samt RAM ingår bland alternativen. Destination kan anges för att välja att undanspara målfilen på ett externt lagringsmedia (USB-minne), alternativt på en plats i datorns hårddisk. En inmatningsruta finns där målfilen kan namnges. Beträffande valmöjligheter (options) har dessa begränsats till att endast kunna påverka blockstorlek, konvertering av output såsom noerror eller komprimering av målfilen och möjlighet till uppdelning av målfilen (split image). När fälten är ifyllda finns en knapp ”Acquire” för att starta processen datainsamling av RAM. En dialogruta visas som innehåller den syntax som skapats genom de val som användaren gjort. Användaren kan bekräfta och därmed starta processen alternativt återgå till inmatningsvyn och utföra ändringar. Beräkning av hashsumma för målfilen, verifiering och loggning ingår i syntaxen utan att användaren explicit behöver ange det.

GUI har en rad fördelar. Det är ett intuitivt och tydligt sätt för användare att interagera med programvara. Funktionerna i programvaran har i positiv mening begränsats genom att för användaren endast visa de mest nödvändiga inställningarna. Många fel och misstag som kan uppstå i samband med användandet av programvaran kan därför undvikas. Begränsningarna hos inmatningsformuläret medför nackdelen att programvaran inte blir lika flexibel som den textbaserade varianten. Möjligheten att ange prioritet för trådar och utökade inställningar saknas. Den främsta nackdelen är att GUI är mer utrymmeskrävande i måldatorns RAM jämfört med den rent textbaserade varianten.

4.2. CrashOnCtrlScroll

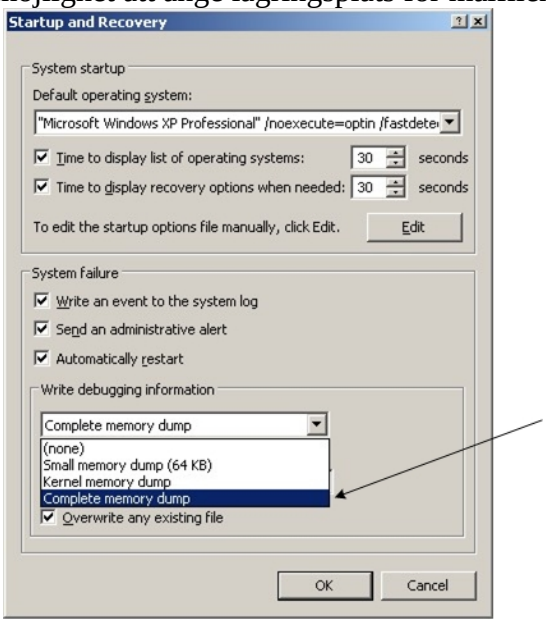
Windows XP behöver konfigureras innan CrashOnCtrlScroll kan användas. I detta avsnitt beskrivs hur en dator kan förberedas för datainsamling med CrashOnCtrlScroll. Ett resonemang kring användningsområdet ingår även.

Figur 10 visar vyn för systemegenskaper. Alternativet ”Startup and Recovery” väljs.



Figur 10. Vy för systemegenskaper och inställningar

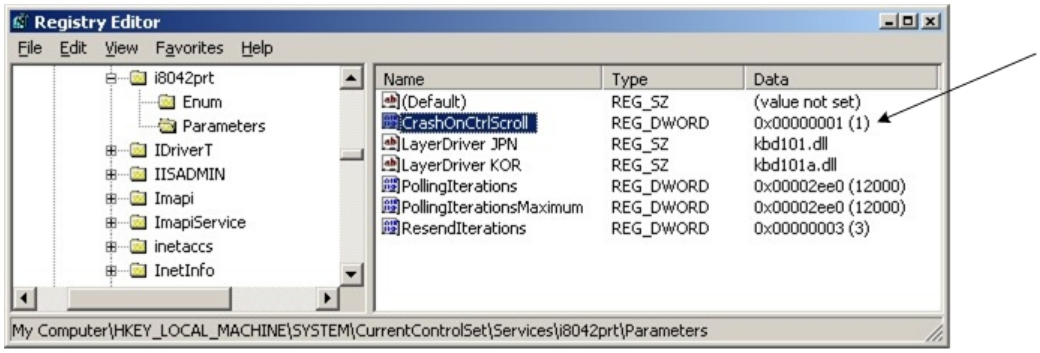
Figur 11 visar de inställningsalternativ som finns tillgängliga för den målfil som skapas. Den typ av minnesdump (målfil) som ska väljas är alternativet fullständig minnesdump (Complete memory dump). Default-alternativ i Windows XP är: liten minnesdump (small memory dump). Detta är anledningen att man aktivt måste ange alternativet fullständig minnesdump. I vyn som visas i Figur 11 finns även möjlighet att ange lagringsplats för målfilen.



Figur 11. Konfiguration av minnesdump

Figur 12 visar konfiguration av registernyckeln. Registereditorn startas och användaren flyttar till följande position:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\i8042prt\Parameters

Denna position framgår av Figur 12 och syns fönstrets nederkant. Ett nytt värde skapas och namnges CrashOnCtrlScroll. Datatypen ska vara REG_DWORD. Värdet sätts till 1. Dessa värden framgår av Figur 12. Pilen hänvisar till den rad som skapats.



Figur 12. Konfiguration av registernyckel och registervärde

När dessa steg är genomförda så måste datorn startas om för att inställningarna ska få effekt. När datorn är omstartad kan den kontrollerade systemkraschen aktiveras. Den aktiveras genom att användaren trycker följande sekvens på tangentbordet: CTRL+SCROLL LOCK+SCROLL LOCK. En blåskärm visas och datainsamling av RAM påbörjas. Resultatet av CrashOnCtrlScroll är en målfil med motsvarande storlek som måldatorns RAM (Microsoft Knowledge Base)

CrashOnCtrlScroll är en datainsamlingsmetod som troligtvis inte kommer att används vid en husrannsakan. Anledningen är att måldatorn kräver omstart för att inställningarna ska få effekt. Vidare så lagras målfilen i måldatorns hårddisk vilket strider mot vedertagna forensiska undersökningsprinciper. Ytterligare en begränsning är att Crashonctrlscroll inte fungerar med ett USB-tangentbord (Microsoft Knowledge Base).

CrashOnCtrlScroll kan däremot bli användbart för datainsamling i samband med incidenthantering på företag. En incidenthanteringsgrupp på ett företag kan genomföra denna inställning i förebyggande syfte innan datorer tas i bruk. Att på förhand göra inställningen medför att man inte behöver starta om datorn när behovet av datainsamling uppstår. Om inställningen är gjord på förhand så kan CrashOnCtrlScroll aktiveras när behov av datainsamling av RAM uppstår. Det kan finnas flera orsaker till att behovet av datainsamling uppstår. Relevant data kan användas vid utredning av IT-relaterade brott och informationsstöld för att nämna några exempel. Data i RAM insamlat enligt denna metod kan användas i samband med en internutredning hos företag. Det är även tänkbart att en incidenthanteringsgrupp på ett företag kan bistå polismyndigheten med dessa data.

5. Analys

Detta kapitel innehåller en SWOT-analys där programvaran DD utvärderas. I kapitlet ingår även en riskanalys. Kriterier för att skapa tillförlitlig programvara presteras och slutligen följer ett avsnitt om IT-forensiska undersökningsprinciper.

5.1. SWOT-analys

SWOT-analys är ett verktyg för att utvärdera Styrkor (Strengths), Svagheter (Weaknesses), Möjligheter (Opportunities) och Hot (Threats) för en verksamhet (Dealtry 1992).

Första steget i en SWOT-analys är att definiera ett mål även kallat sluttillstånd (end state).
Ett mål måste vara tydligt definierat och det måste finnas en rimlig möjlighet att uppnå målet. Analysmetoden kan fungera som ett planeringsstöd för att vägleda en verksamhet mot ett specifikt mål. Verksamheten i detta fall är kriminaltekniska undersökningar. Målet är definierat i målbeskrivningen nedan. En SWOT-analys kan användas för att beskriva hur man ska undvika hot och svagheter men även hur man tar tillvara på styrkor och möjligheter.

5.1.1. SWOT-analys tillämpat på programvaran DD

Målbeskrivning: Att hitta bevis som är relevant för en brottsutredning.

I nedanstående analys har styrkor, svagheter, möjligheter och hot identifierats. DD används i en process som styrs av målbeskrivningen.

Styrka(S):

- DD kan undanspara data i RAM i driftsatta datorsystem, DD kan även genomföra datainsamling av oallokerat utrymme i RAM
- DD kräver vid körning endast liten mängd av måldatorns RAM
- DD levererar en målfil som har ett enkelt format
- Den ursprungliga versionen av DD är ett beprövat program
- DD kan beräkna och verifiera hashsummor för målfiler
- DD har möjlighet att ställa trådprioritet så att processen datainsamling av RAM kan genomföras snabbare
- DD kan vara den enda möjliga metoden vid datainsamling (proportionalitetsprincipen[8])

Svaghet(W):

- Datainsamling av driftsatta system sker inte i en kontrollerad och säker kontext
- Tillämpningen av DD för datainsamling i driftsatta datorsystem är relativt ny och dokumentation och erfarenhet saknas
- DD kräver administratörsrättigheter för att köras

[8] Proportionalitetsprincipen – om tvång måste användas så ska det stå i proportion till det som ändamålet omfattar.

Möjligheter(O):

- Att hitta relevant data för brottsutredningen som inte går att finna med traditionella metoder
- DD är öppen programvara och möjlighet finns att granska källkoden
- Lagringskapaciteten på hårddiskar är stor och ökar i ständig takt. Att genomföra analys av all data under en överskådlig tid är ogenomförbart. Förundersökningen ska bedrivas så snabbt som möjligt (skyndsamhetsprincipen[9]) (Falk & Lindblom, 2005). Data i RAM kan ge strategisk vägledning och indikationer på var i hårddisken relevant data finns lagrad. Indikationer kring var i hårddisken IT-forensiker ska börja leta kan snabba på informationsanalysprocessen för stora datamängder.

Hot(T):

- Att kärnan levererar ett missvisande resultat på grund av rootkits
- Att DD förvanskar data i måldatorns hårddisk. Integriteten kan påverkas, DD påverkar systemfiler i måldatorns operativsystem
- Data i RAM är flyktig och relevant data kan försvinna innan den har hunnit samlas in.

SWOT-analysens andra steg är att från ovanstående identifierade uppgifter besvara en rad frågor. Genom att besvara frågorna konkretiseras idéer och resonemang kring vilka kriterier som är viktiga vid val av programvara för datainsamling av RAM i driftsatta datorsystem.

I följande frågor beskrivs analysresultatet

1. Hur kan varje Styrka Användas?

- *DD kan undanspara data i RAM i driftsatta datorsystem, DD kan även genomföra datainsamling av oallokerat utrymme i RAM*

DD kan användas vid kriminaltekniska undersökningar som programvara i en process för datainsamling av RAM i driftsatta datorsystem. Denna process hjälper IT-forensikern tillsammans med traditionella metoder att skapa en helhetsuppfattning kring data och användarmönster i en måldator. Att DD även kan genomföra datainsamling av oallokerat utrymme i RAM är en styrka. Det är sannolikt att relevant data för en undersökning finns i de oallokerade delarna av minnet. Att kunna undanspara dessa data är relevant för brottsutredningen.

- *DD kräver vid körning endast liten mängd av måldatorns RAM*

Denna styrka är eftersträvansvärd. Programvaran som används för datainsamling av RAM ska endast kräva minimalt av måldatorns minne (i idealfallet inget minne) för att fungera. Detta är ett av de kriterier som bör prioriteras. Det man vill göra är att undanspara all data i måldatorns RAM. För att kunna göra det måste programvaran själv använda (allokera) lite av utrymmet i RAM för att kunna fungera. En del av de data man önskar undanspara måste alltså direkt offras för att kunna nå resten av data. Detta är ett av områdets dilemman.

- *DD levererar en målfil som har ett enkelt format*

Datainsamling är första steget i processen att finna data som är relevant för brottsundersökningen. I andra steget analyseras data (genomsöks och tolkas) i hopp om att hitta bevis.

[9] Skyndsamhetsprincipen – regleras i RB 23:4, st 2 p.1. Innebär att förundersökningen skall bedrivas så snabbt som möjligt.

Informationsanalysfasen underlättas om de data som insamlades i steg ett har ett hanterbart format. DD levererar en målfil som har ett enkelt format. Målfilens offset motsvarar den absoluta adressen i RAM.

- *Den ursprungliga versionen av DD (GNU DD) är ett beprövat program*

Garners version av DD bygger på den ursprungliga versionen GNU DD som historiskt använts under relativt lång tid. Programmet har främst använts för att säkerhetskopiera databand. Programmet är för den tillämpningen beprövad och vedertagen. Användningsområdet att utnyttja DD för att genomföra datainsamling av RAM är förhållandevis ny. Att DD för äldre tillämpningar är beprövat kan emellertid bidra till en ökad tillförlitighet åt programmet även i den nya tillämpningen.

- *DD kan beräkna och verifiera hashsummor för målfiler*

Att beräkna hashsummor för målfiler är en vedertagen forensisk undersökningsprincip. Det är en fördel att DD har en inbyggd funktionalitet att genomföra denna hashsummering. Funktionen kan användas för att verifiera målfilens integritet.

- *DD har möjlighet att ställa trådprioritet så att processen att undanspara data i RAM kan genomföras snabbare*

Denna styrka kan användas för att processen datainsamling av RAM ska ges en hög prioritet. Schemaläggaren kommer att hantera de processer som körs i datorn. Om en process får hög prioritet kommer den tilldelas mer CPU-kraft. Resultatet blir att processen att undanspara data i RAM kan slutförs snabbare. Då data i RAM är flyktig är det önskvärt att processen genomförs så snabbt som möjligt, för att minimera eventuella dataförluster.

- *DD kan vara den enda möjliga metoden vid datainsamling (proportionalitetsprincipen)*

Att ta en dator i beslag vid husrannsakan på en arbetsplats kan beroende på brottstyp vara en oproportionerlig åtgärd. Enligt åklagare Håkan Rosvall beaktas särskilt om tredje man blir drabbad. Den term som närmast kan beskriva detta brukar förekomma i juridisk doktrin och omnämns som proportionalitetsprincipen. I fall då det anses vara oproportionerligt att ta datorer i beslag, för datainsamling och analys på annan plats, kan processer för datainsamling av driftsatta datorsystem RAM vara en tänkbar åtgärd (även att selektivt kopiera data från utvalda delar av hårddisken är tänkbart). Proportionalitetsprincipen borde även kunna tillämpas när datorsystem är affärskritiska och stor skada kan drabba ägaren av utrustningen om den tas ur drift. Istället för att stänga ner datorsystemen för att utföra datainsamling så kan datainsamlingen ske i det driftsatta datorsystemet. Att använda programvara såsom DD för datainsamling av RAM kan alltså vara en proportionerlig åtgärd då andra metoder inte kan motiveras med hänsyn till brottets art.

2. Hur kan vi minimera varje svaghet?

- *Datainsamling i driftsatta system sker inte i en kontrollerad och säker kontext*

Datainsamling av driftsatta datorsystem sker inte under samma förutsättningar som traditionella IT-forensiska undersökningar. Kontexten är inte kontrollerad och säker. Åtgärder måste vidtas för att minimera denna svaghet. Till exempel måste programmet som används för datainsamling vara dokumenterat. Det ska tydligt framgå vilka filer i måldatorns operativsystem som kommer att påverkas då programmet används. En utförlig dokumentation ska göras på ett kontrollerat sätt innan programmet kan användas operativt i verksamheten. Programmet bör vidare köras från ett read-only media (CD/DVD-rom). Det bör även dokumenteras i exempelvis protokoll för

husrannsakan vem det är som har utfört datainsamlingen av det driftsatta systemet och exakt vilka återgärder som vidtagits. Det sistnämnda för att säkerställa spårbarhet.

- *Tillämpning av DD för datainsamling i driftsatta datorsystem är relativt ny och dokumentation och erfarenhet saknas*

Hur effekten av denna svaghet kan minska är delvis besvarad i föregående fråga. Test och dokumentation måste vara omfattande innan programvaran tas i bruk. Personal måste utbildas och tränas i hanterandet av programvaran och processen att genomföra datainsamling i driftsatta datorsystem. Testscenarion som skapas för träning bör vara varierade och bör efterlikna den kontext som programvaran är tänkt att fungera i. Rutin i att använda programvaran bör skaffas genom träning i en kontrollerad miljö för att förhindra skada som kan orsakas av en felaktig hantering i en operativ verksamhet. Det måste gå att avgöra när programvaran kan användas och när det inte är möjligt att använda den. De begränsningar som programvaran har måste vara kända.

- *DD kräver administratörsrättigheter för att köras*

Det är inte känt hur denna svaghet kan minimeras. Svagheten utgör en begränsning vid användandet av programvaran DD för datainsamling av RAM i driftsatta datorsystem. Begränsningen förmodas bli än mer påtaglig vid lansering av nästa version av Windows nämligen Windows Vista. Windows Vista kommer jämfört med Windows XP förändra hanteringen av användarkonton och behörigheter. I Windows XP förekommer att ”normalanvändare” har administratörsrättigheter. Detta är dåligt ur säkerhetshänseende, men bra ur ett IT-forensiskt perspektiv, då det möjliggör för utredare att köra program såsom DD. I Windows Vista kommer säkerhetsfunktioner att vara inbyggda. Dessa hindrar ”normalanvändare” att starta med administratörsrättigheter, och kan ses som en åtgärd för att höja säkerhetsnivån. När ”normalanvändare” vill utföra systeminställningar eller installera program behöver de ange lösenord för att tillfälligt bli administratör, när åtgärden är utförd så återgår kontot till ett ”normalanvändarkonto” med lägre behörighet. Konsekvensen vid en husrannsakan blir troligen att administratörslösenord kommer att behövas för att använda program såsom DD för datainsamling.

3. Hur kan vi utnyttja varje möjlighet?

- *Att hitta relevant data för brottsutredningen som inte går att finna med traditionella metoder*

Traditionella IT-forensiska undersökningsmetoder har främst varit inriktade mot datainsamling och analys av data lagrat i hårddiskar hos beslagtagna datorer. Som tidigare nämnts kan det även finnas relevant data för en brottsutredning lagrat i RAM i driftsatta datorsystem. I de fall det går att genomföra datainsamling av den typen av data vid en brottsutredning bör man utnyttja den möjligheten. Programvara såsom DD, men även funktionalitet inbyggt i operativsystemet (CrashOnCtrlScroll) och hårdvarulösningar (Tribble, Firewire etc.) är exempel på olika sätt att genomföra datainsamling.

- *DD är öppen programvara och möjlighet finns att granska källkoden*

DD är öppen programvara och har granskningsmöjlighet, vilket är en fördel. Möjligheten kan användas för att granska källkoden och verifiera att programvaran verkligen utför de uppgifter och producerar de resultat som den utger sig för att kunna göra. Granskningsbarheten kan även bidra till ökad datakvalitet i den målfil som DD levererar. Genom att granska källkoden kan de

anrop som DD utför till operativsystemet kartläggas och dokumenteras. Programvarans beroende till operativsystemet kan genom granskning bli känd.

- *Lagringskapaciteten på hårddiskar är stor och ökar i ständig takt. Att genomföra analys av all data under en överskådlig tid är ogenomförbart. Förundersökningen ska bedrivas så snabbt som möjligt (skyndsamhetsprincipen) (Falk & Lindblom, 2005). Data i RAM kan ge strategisk vägledning och indikationer på var i hårddisken relevant data finns lagrad. Indikationer kring var i hårddisken IT-forensiker ska börja leta kan snabba på informationsanalysprocessen för stora datamängder.*

Denna möjlighet är en av fördelarna med att använda data som finns i RAM i samband med en IT-forensisk undersökning. Relevant data som kan finnas i insamlad data kan ge vägledning var IT-forensiker ska börja sökandet med de traditionella IT-forensiska metoderna. Detta kan göra informationsanalysprocessen effektivare.

4. Hur kan vi försvara oss mot varje hot?

- *Att kärnan levererar ett missvisande resultat på grund av rootkits*

Det enda sättet att helt eliminera detta hot är att finna sätt som gör att datainsamlingen kan fungera utan beroenden till måldatorns kärna. Med programvarulösningar som körs i måldatorns operativsystem kommer detta hot aldrig gå att helt eliminera. Hårdvarulösningar såsom Tribble har föreslagits som lösning för att ta bort beroendet till måldatorns kärna och operativsystem. Denna lösning skulle eliminera hotet att kärnan kan leverera en målfil med låg datakvalitet.

- *Att DD förvanskar data i måldatorns hårddisk. Integriteten kan påverkas, DD påverkar systemfiler i måldatorns operativsystem*

När data i driftsatta datorsystem undansparas med programvara kommer filer i måldatorns hårddisk att påverkas. Alla filer i datorns hårddisk innehåller data om tider som beskriver när filen skapades, ändrades och senast lästes (så kallade *MAC-times*). Systemfiler som främst hör till måldatorns operativsystem kommer att anropas av DD. Tider då filer senast lästes kommer att uppdateras och därmed förändras då DD körs. Detta är en förändring som påverkar integriteten hos filer i måldatorns hårddisk. Den främsta åtgärden man kan vidta för att minska effekten av detta hot är att på förhand karlägga och dokumentera vilka filer som påverkas vid användandet av DD. Det är ett dilemma att man måste offra viss integritet hos data i filerna i måldatorns hårddisk för att kunna undanspara data i RAM.

- *Data i RAM är flyktig och relevant data kan försvinna innan den har hunnit samlas in.*

Detta hot är svårt att hantera. I DD kan man ställa trådprioritet och därmed få processen datainsamling av RAM, att genomföras snabbare. Det behöver inte vara en nackdel att data förändras. Likväl som relevant data kan försvinna i RAM så är det även möjligt att ny data av lika eller högre relevans kan tillkomma. Det behövs mer kunskap om vad som har inträffat och när något har inträffat för att kunna ta beslut om det är viktigt att datainsamling ska ske snabbt, eller om det kan vara en fördel att avvakta en tid.

5.2. Riskanalys

Denna riskanalys har tre element. Dessa element kan formuleras i följande tre frågor.
Vad kan hända?
Hur troligt är det?
Hur värderar vi konsekvenserna?

Som tidigare nämnts är den främsta risken, vid användandet av programvara för datainsamling av RAM, att måldatorn kan vara infekterat av rootkits.

Vad kan hända?
Måldatorn kan vara infekterad av rootkits som gör att kärnan levererar otillförlitlig data.

Hur troligt är det?
Sannolikhetsuppskattningen är mycket svår att göra. Tillräcklig kunskap om verkligheten saknas. I riskanalyssammanhang brukar termen kalibrering användas. Med kalibrering menas hur väl sannolikhetsuppskattningen stämmer med verkligheten. Inga bra uppgifter finns om hur vanligt det är att en dator kan infekteras av rootkits. Kalibreringen i detta fall är därför svag beroende på okunskap om verkligheten.

Sannolikheten att en dator ska vara smittad av rootkits beror delvis på följande faktorer:

- Är datorn uppkopplad mot internet ofta och länge?
- Har datorn några säkerhetsfunktioner (antivirusprogram) som kan upptäcka och avlägsna rootkits?

Rootkits infekterar datorer vanligast genom internet. Om datorn är uppkopplad till internet under en lång tid och frekvent så ökar sannolikheten att datorn kan vara infekterad.

Det är vanligt att datorer har antivirusprogram. Det finns en mångfald av olika leverantörer av antivirusprogram av skiftande kvalitet. De största leverantörerna brukar ingå i produkttester av antivirusprogramvara för att se i vilken utsträckning dessa kan detektera förekomsten av och även avlägsna rootkits. Hur mycket man än testar antivirusprogramvara så finns alltid en sannolikhet att nya eller ovanliga rootkits kan undgå att bli upptäckta. Trots detta kan de faktum att datorn har ett antivirusprogram installerat bidra till den totala sannolikhetsbedömningen. Det är mindre sannolikhet att en dator är infekterad av rootkits om denna har ett antivirusprogram installerat jämfört med en identisk dator utan antivirusprogram, förutsatt att datorerna används lika.

Följande exempel belyser svårigheten med kalibrering ytterligare: Det är relevant att anta att det finns en anledning till att användaren har installerat ett antivirusprogram. Datorer med antivirusprogram installerat kanske i högre utsträckning är uppkopplade mot internet jämfört med datorer som aldrig eller sällan är uppkopplade.

Är datorn sällan eller aldrig uppkopplad mot internet och har datorn dessutom ett antivirusprogram av god kvalitet är sannolikheten låg att datorn kan vara infekterad av rootkits.

Är däremot datorn ofta eller alltid uppkopplad mot internet och saknar datorn dessutom antivirusprogram är sannolikheten hög att datorn kan vara infekterad av rootkits.

Hur värderar vi konsekvenserna?
Konsekvensen av att rootkits kan ha infekterat måldatorn är att data som kärnan levererar från måldatorn kan innehålla fel. En värdering av datakvaliteten är utanför omfattningen av denna uppsats. Anledningen är att uppsatsen har fokus på datainsamling och inte informationsanalyssteget. Som tidigare nämnts råder fri bevisprövning i svensk rätt och det är upp till rätten att värdera datakvaliteten och relevans hos de bevis som åberopas.

Det kan däremot nämnas att det framkom i intervju med åklagare Håkan Rosvall att de faktorer som påverkar bevisvärderingen är de processer som används för datainsamling. Att datainsamlingen skett i en kontrollerad kontext har betydelse.

Falk & Lindblom (2005) skriver om bevisvärdering i sin uppsats om *”Loggar som bevisning”*. Deras uppsats ger en utförlig förklaring och ett resonemang kring de faktorer som påverkar bevisvärdering. Författarna menar att all bevisning som presenteras i ett förundersökningsprotokoll har, eller kan ha, olika nivåer av bevisvärde.

Sammanfattningsvis kan konstateras att konsekvensen av att måldatorn infekterats av rootkits påverkar datakvaliteten och därmed bevisvärderingen. Om det kan bevisas att måldatorn innehåller rootkits kan konsekvensen bli att rätten ger bevisen ett lågt bevisvärde. Om det kan bevisas att måldatorn inte innehåller rootkits så kan bevisen få ett högre bevisvärde.

5.3. Att skapa tillförlitig programvara

I detta avsnitt följer en sammanställning hur man kan skapa tillförlitlig programvara som sedan kan användas vid datainsamling av driftsatta datorsystem. All programvara som används kommer att lämna spår efter sig i måldatorn som undersöks. Programvara som används ska påverka måldatorn i så liten utsträckning som möjligt. Det ska vidare vara dokumenterat och känt hur programvaran påverkar systemet. Tillförlitigheten är främst beroende av till vilken grad man förmår beskriva vad ett program gör när det används, samt dokumentation av de filer och bibliotek som påverkas. I en rättegång är det viktigt att denna dokumentation är korrekt och fullständig, annars kan bevisens värde minska.

5.3.1. Tillförlitighet för program i Windows

Windows har den egenskapen att källkoden inte är publikt tillgänglig och därmed inte granskningsbar. Applikationer som körs i Windowsmiljö använder gemensamma programbibliotek i operativsystemet. Programmen är således dynamiskt kompilerade och har beroenden till så kallade DLL-filer (dynamic link library). DLL är en samling programmoduler som används för olika tillämpningar. För utredningar innebär detta att program som körs på en dator som är igång kommer att påverka många filer i datorns operativsystem. Det är viktigt att känna till vilka dessa filer är och på vilket sätt filerna påverkas.

MAC-times (Modified, Accessed, Created) är en typ av metadata som beskriver när en fil senast förändrades, senaste tid för åtkomst av filen samt tiden när filen skapades (Casey E. 2004b). Med tid menas både datum och klockslag. En undersökning av MAC-tider på en dator kan ge kunskap om användarmönster, avsikten med vissa filer och hur ofta och när filernas innehåll har använts och förändrats.

Vid datainsamling i Windows kommer programmen som används för datainsamling att påverka MAC-tiderna. Data, till exempel senaste tid för åtkomst av en viss DLL-fil, kommer att förändras. Det gäller att på förhand veta vilka dessa filer är och dokumentera alla förändringar som sker i systemet. Vid en rättegång måste fullständig dokumentation presenteras, annars kan datakvaliteten ifrågasättas.

Dokumentationen av datainsamlingsprogrammens funktion och beroenden ska genomföras i en kontrollerad kontext. Datorerna bör vara installerade med känd och beprövad programvara och anslutning till internet och nätverk ska undvikas så att risken för virus och rootkits minimeras.

Sysinternals har skapat en rad användbara program för att bland annat analysera hur applikationer påverkar operativsystemet. Filemon (Sysinternals Filemon 2006) är exempel på en programvara som kan användas för att kartlägga vilka DLL-filer som applikationer använder. Regmon (Sysinternals Regmon 2006) är ett annat program som visar de register och nycklar som en specifik applikation påverkar. Filemon tillsammans med Regmon ger en överblick hur ett program påverkar operativsystemet. Gemensamt för programmen är att data som samlats in går att exportera till exempelvis Excel, så att utredaren kan dra fördel av sorteringsfunktionerna där. Det rekommenderas att dokumentationen berikas med hashsummor för varje DLL-fil som påverkas. Detta blir användbart då man i efterhand kan spåra förändringar samt verifiera att programmen fungerat som beräknat.

Sedan juli 2006 har Sysinternals köpts av Microsoft. De flesta program som tidigare funnits tillgängligt på Sysinternals hemsida finns numera tillgängliga på en hemsida som drivs av Microsoft[10]. Denna förändring ägde rum i början av november 2006. Förändringen har skapat stor debatt bland anhängare av Sysinternals programvaror. Det är ovisst vad förändringen kommer att innebära. Det som däremot står klart är Microsoft har valt att inte publicera källkoden till den programvara som skapats av Sysinternals. Denna källkod var tidigare tillgänglig på Sysinternals hemsida. Det är en nackdel att källkoden inte längre finns tillgänglig, främst för att granskningsmöjlighet saknas.

Process Monitor[11] är en programvara som har tillkommit efter det att Microsoft köpte Sysinternals. Funktionsmässigt är programvaran en sammanslagning av programmen Filemon och Regmon. Analys av sättet en programvaran påverkar operativsystemet och filer på hårddisken blir enklare med Process Monitor. En rad förbättringar är gjorda och ny funktionalitet har tillkommit. Funktioner för att sätta filter och göra sortering gör analysarbetet både tydligare och enklare.

[10] http://www.microsoft.com/technet/sysinternals/default.mspx
[11] http://www.microsoft.com/technet/sysinternals/processesandthreads/processmonitor.mspx

5.4. IT-forensiska undersökningsprinciper

Detta avsnitt behandlar rekommendationer som IT-forensiker bör tillägna sig. Bra metodik kan vägleda utredaren i IT-forensiska undersökningar av driftsatta datorsystem.

5.4.1. Använd beprövande och kända binärer

En utredare ska inte lita på de exekverbara filer (program) som finns i måldatorn (Adelstein F. 2006). Program som används vid undersökningen ska tillhandahållas på ett kontrollerat sätt. Dessa ska i första hand vara statiskt kompilerade, alternativt i andra hand inkludera de delade bibliotek som krävs av den exekverbara filen. Orsaken till detta är främst de risker som är förknippade med rootkits. Programmen ska härstamma från ett read-only medium, exempelvis CD-rom eller DVD-rom. Programmen kan kopieras till måldatorn (även om detta bör undvikas), man bör vara medveten om att det kommer påverka data i måldatorns hårddisk och därmed riskera att relevant data förändras eller förloras. En riskbedömning ska därför göras. Bedömningen bör väga risker för förändring av data (exempelvis förlust av raderade filer på hårddisken) genom att kopiera programmen till datorn mot risken att förlora all data i RAM om man inte vidtar den åtgärden. Det kan i vissa fall vara bättre att acceptera viss förlust av data, jämfört med att inte erhålla några data alls.

5.4.2. Beräkna kondensat (hashsummor) för varje bevis

När data har insamlats måste denna bevaras på ett sätt som möjliggör för utredaren att vid ett senare tillfälle kunna verifiera att inga data har förändrats (att integriteten har upprätthållits). Tillvägagångssättet är att för varje datafil beräkna ett kondensat (hashsumma), vanligen genom hashfunktionerna MD5 eller SHA-1. Kondensatet representerar ett fingeravtryck som kan beräknas från datafilen vid upprepade tillfällen och verifieras mot originalet för att fastställa integriteten. Om data överförs genom ett nätverk från måldator till en annan dator ska kondensat beräknas hos båda datorerna för att säkerställa att data inte har förändrats vid överföringen.

Såväl kondensaten som datafilerna ska förvaras på säkra platser. För att bevara integriteten brukar utredaren vanligen signera (underskrift, tid och datum) en pappersutskrift av kondensaten och arkivera dessa i säkert förvar.

5.4.3. Ordningsföljd av datainsamlingen beroende på flyktighetsnivå hos data

Som beskrivits i avsnitt 2.2 har nivå av flyktighet betydelse vid insamling av data. Viss data är mer flyktig än annan. Datainsamlingen ska börja med data som har högst grad av flyktighet (se Figur 3). Som exempel ändras nätverksanslutningar mer frekvent jämfört med genomsnittlig systemaktivitet.

Tiden det tar för datainsamling beror på vilken slags data som hanteras. Detta faktum komplicerar processen ytterligare. Datainsamling av RAM kan ge relevant data och data i RAM är dessutom flyktig vilket indikerar att insamling ska ske tidigt i processen. Emellertid kan processen ta ett tiotal minuter att genomföra och under den tiden kan annan relevant data förändras eller rentav försvinna. Exempel på relevant data som kan gå förlorad kan vara listor över processer som körs, filer som är öppna, nätverksanslutningar och dylikt. Trots det faktum att data i RAM

kontinuerligt förändras så kan även vissa minnesområden vara oförändrade under en längre tid (ibland flera dagar) i ett system med 1 Gbyte minne. En utredare måste mot bakgrund av detta vara införstådd i hela kontexten för att kunna fatta beslut rörande datainsamlingsprocessen.

5.4.4. Bevara fysiska bevis

Det är nära till hands att i sammanhanget enbart fokusera på relevant data som finns inne i datorn. Detta är en felaktig bild då det kan existera bevis av relevans även utanför datorn eller i dess omgivning. Det gäller för utredarna att tänka brett och använda erfarenhet och fantasi för att kunna genomföra bevissäkring i datorns omgivning.

För att bevara fysiska bevis i datorns omgivning bör några försiktighetsåtgärder vidtas. Tangentbordet hos måldatorn bör inte vidröras. Eventuella spår av fingeravtryck kan förstöras. Då datainsamling ska genomföras bör ett medtaget tangentbord användas. Att ansluta ett medtaget tangentbord påverkar dock måldatorn. Det bör på förhand vara dokumenterat vilka effekter som uppstår i måldatorn när medtagen utrustning ansluts. Vidare bör datorns kablar vara orörda under hela datainsamlingsförloppet. Datorns sammansättning beträffande anslutna kablar och kringutrustning bör alltid fotograferas, alternativt videofilmas. Dokumentation ska följa fastslagna rutiner. Vidare bör specialutbildad och erfaren personal undersöka datorns omgivning efter bevis. Exempelvis lappar med lösenordsanteckningar, böcker, disketter, CD, DVD, USB-minnen, backup media, mobiltelefoner och PDA.

6. Slutsatser och diskussion

I detta avsnitt sammanställs de slutsatser som gjorts. En diskussion kring slutsatserna följer. Slutligen presenteras förslag på framtida forskning inom området live computer forensics.

6.1. Slutsatser

Några av de slutsatser som gjorts sammanfattas i följande punktlista:

- Programvaran DD kan användas för att datainsamling av RAM i driftsatta datorsystem som använder operativsystemet Windows XP.
- Rootkits i måldatorn utgör den främsta risken vid användandet av programvara för datainsamling av RAM.
- Integriteten för data i måldatorns hårddisk kommer att påverkas vid användandet av programvara för datainsamling av RAM.
- Relevant data i RAM kan användas för att göra brottsutredningen effektiv.
- Relevant data i RAM kan leda till bevis som man med traditionella undersökningsmetoder inte kan hitta.
- Dokumentation av programvara som används för datainsamling av RAM har betydelse för datakvaliteten för den målfil som programvaran levererar.
- Hårdvarubaserade lösningar för datainsamling tycks vara den enda möjligheten till en datainsamlingsprocess som inte har beroenden till måldatorns operativsystem och kärna.

Programvaran DD kan användas för datainsamling av RAM i driftsatta datorsystem. De skäl som talar för att använda DD är det enkla formatet hos den målfil som programvaran skapar. Att använda DD i den textbaserade kommandotolken är att föredra. Programvaran kräver då minimalt med RAM hos måldatorn. De data i RAM som man önskar undanspara kommer endast i begränsad utsträckning att förloras på grund av att programmet självt behöver utnyttja minnet.

De skäl som talar emot att använda DD är främst att det finns beroenden till operativsystemet och kärnan. Kan man lita på de resultat som programvaran levererar? Hur stor är sannolikheten att insamlad data innehåller fel på grund av att kärnan har infekterats av rootkits? Det som ytterligare är en nackdel med DD är att programmet kräver administratörsrättigheter i måldatorns operativsystem för att fungera.

Ett av de kriterier som bör prioriteras är möjligheten att granska programvaran. Programvara innehåller beroenden till måldatorns operativsystem och kärna. Det är viktigt att kartlägga och dokumentera hur måldatorn påverkas när dessa program används. Process Monitor (Sysinternals) som beskrivits i avsnitt 5.3.1 kan användas för att registrera vilka filer som förändras då ett visst program körs. Process Monitor bör lämpligen användas i en kontrollerad och säker kontext för att kartlägga datainsamlingsprogrammets beroenden till måldatorn. Att använda Process Monitor för kartläggning av programvaran är inte tillräckligt. Med öppen programvara finns möjlighet att granska källkoden. Analys av källkoden kan ge svar på hur programvaran är uppbyggd och vilka funktioner som programvaran anropar.

Programvara som användas för datainsamling kan ifrågasättas i rätten. Detta gäller särskilt då det finns ett beroende till operativsystemet och kärnan, åtminstone så länge det finns en risk att kärnan eller operativsystemet är infekterade av rootkits. Domare och advokater saknar generellt tillräckliga kunskaper inom IT för att ifrågasätta digitala bevis. Test, dokumentation och certifiering av programvara kommer troligen att öka i takt med att kunskap om IT ökar i rätten.

Ett resultat som varit oväntat är att data i RAM inte tycks vara så flyktig som man tidigare trott (Farmer D. & Venema W. 2004). Data i RAM kan i driftsatta system vara oförändrad under relativt lång tid. Den allmänna uppfattningen har tidigare varit att data i RAM går förlorad om datorn startas om. Information från litteraturstudien visar att data i RAM kan vara oförändrad trots att datorn omstarts. Detta tycks gälla i synnerhet för Apple G4. Kanske kan denna upptäckt användas för att i BIOS avaktivera den instruktion som nollställer data i RAM.

Ett problem med programvara som används för datainsamling är att programvaran själv kräver minne i måldatorn. När programvaran utnyttjar minne går motsvarande mängd data förlorad i RAM, alltså data som man önskar samla in. Vilken plats i måldatorns minne som programvaran utnyttjar får betydelse. I RAM finns normalt reserverad plats som används för systemspecifika funktioner; exempelvis del av RAM som utnyttjas som videominne. Tänkbart är att låta programvaran allokera den delen av minnet för att undvika att relevant data går förlorad. Detta är en möjlighet för framtida forskning.

6.2. Diskussion

Målet med examensarbetet är att ta fram kriterier för val av programvara och processer som kan användas för datainsamling av RAM i driftsatta datorsystem. Dessa kriterier har identifierats, analyserats och presenterats i en utvärderingsmodell, SWOT-analys och riskanalys.

Uppsatsens resultat överrensstämmer väl med de förväntande resultat och antaganden som gjordes inledningsvis under arbetet. Resultatet är främst baserat på information från litteraturstudien. De artiklar som studerats har huvudsakligen varit publicerade i den vetenskapliga tidskriften Digital Investigation. Det är rimligt att anta att resultatet hade blivit annorlunda om uppsatsen helt, eller till största del, baserats på intervjuer. Området live forensics är relativt nytt och metoderna används sällan eller aldrig operativt. Detta var den främsta orsaken till att informationsinhämtningen nästan uteslutande baserats på litteraturstudier. En annan orsak är den sekretess som omfattar metoder som används vid kriminaltekniska undersökningar. Information från intervjuer skulle endast i allämna formuleringar kunnat återges. Information från artiklar som publicerats kunde används utan hinder av sekretess.

Det är rimligt att tänka sig någon form av certifiering för den programvara som användas operativt för datainsamling av RAM. Troligen kan Statens kriminaltekniska laboratorium (SKL) vara den myndighet som får till uppgift att genomföra en sådan certifiering. I synnerhet då metodutveckling omfattas av myndighetens verksamhet.

DD har begränsningar och några av dessa begränsningar har identifierats. DD kräver administratörsrättigheter för att kunna användas. Denna begränsning kommer förmodligen få konsekvenser vid datainsamling, i synnerhet när Windows Vista tas i bruk. Windows Vista

kommer att innehålla säkerhetsfunktioner som förhindrar användarkonton att i normalläge köras med administratörsrättigheter. Endast tillfälligt kommer användarkonton att kunna tilldelas administratörsrättigheter och för detta kommer lösenord krävas.

Eftersom datainsamling och analys av driftsatta datorsystem är en relativt ny företeelse så är det oklart hur data som insamlats kommer att användas. I informationsanalysfasen tolkas data till information som i sin tur kan tolkas till bevis. Även om informationen inte används som bevis så kan relevant information vägleda utredaren. Ledtrådar kan ges kring var på hårddisken man med traditionella metoder ska leta.

6.3. Framtida forskning

Behovet av tillgängliga och beprövade metoder för att kunna genomföra datainsamling i driftsatta datorsystem kommer troligen att öka i framtiden. Området är relativt nytt och få studier är gjorda. Nedan återfinns några förslag på framtida studier inom området live computer forensics. Förslagen har identifierats under arbetets gång och sådant material som tyvärr, främst av tidsmässiga begränsningar, inte kunnat ingå i denna studie.

En kartläggning av programvaran DD genom granskning av källkoden
DD är öppen programvara och källkoden är tillgänglig. Delar av källkoden för DD har granskats genom studier av källkoden i utvecklingsmiljön Microsoft Visual Studio. Av tidsmässiga begränsningar har inte hela källkoden kunnat granskas i denna studie. Genom att granska källkoden kan man kartlägga hur DD är uppbyggt och vilka beroenden som finns till operativsystemet. Denna kartläggning kan användas som input vid en eventuell framtida certifiering av programvaran.

Undersökning av hårdvarulösningar för datainsamling i driftsatta datorsystem
Forskning med inriktning mot hårdvarulösningar för datainsamling vore intressant. Tidigare resultat har exempelvis varit Tribble (se avsnitt 2.3). Experiment med datainsamling genom Firewire har nämnts i olika sammanhang och det finns anledning att undersöka den möjligheten närmare. Det finns fördelar med hårdvarulösningar. En fördel är att datainsamling kan utföras utan beroende till måldatorns kärna. Om detta beroende tas bort försvinner också den största risken till otillförlitlig data, nämligen att kärnan innehåller skadlig kod och levererar otillförlitlig data.

Undersökning av programvara för informationsanalys av de data som insamlats
När datainsamling är genomförd kan informationsanalysfasen påbörjas. En undersökning som utreder programvara och metoder för att genomföra informationsanalys av data vore önskvärd. Det finns ingen programvara i dagsläget som automatiskt kan genomföra en analys av data insamlad från ett driftsatt datorsystem. Vid analys av data måste olika programvaror för specifika syften användas.

Några studier är gjorda inom detta område och några resultat har publicerats. Till exempel Schuster (2006) som kartlägger en metod för att hitta processer och trådar. Schuster har skrivit ett program som heter PTFinder. Programmet skapades med avsikt att identifiera _EPROCESS och _ETHREAD strukturer i en undansparad målfil.

Farmer & Venema (2004) beskriver ett förfarande att identifiera text ur data insamlad från RAM. Denna text var sådan som nyligen hade krypterats. Programvaran Strings kan användas för att extrahera text ur en målfil. Metoden att upptäcka text som krypteras bygger på en svaghet i Windows som gör att data i filer som nyligen krypterats överlever i klartext i RAM en viss tid efter krypteringen.

Datainsamling genom hibernate
Det finns en teknik som kallas hibernate. Man kan enkelt beskriva det som att datorn försätts i ett viloläge med låg strömförbrukning. Finessen är att datorn kan starta mycket snabbare än vanligt och operativsystemet behåller även data kring de program som körs. Detta är användbart främst för bärbara datorer med begränsad tillgång till ström (vid batteridrift) samt där användare arbetar med datorn i omgångar. Hibernate sparar delar av data i RAM till en fil (vanligen hibernate.sys). Det är denna fil som är relevant ur ett kriminaltekniskt perspektiv. Förslaget är en studie kring möjligeter att använda denna fil för att forensiskt återställa en dator och sedan genomföra en informationsanalys.

Hibernate ser vid en första anblick ut att vara en bra metod för datainsamling. Emellertid har metoden vissa begränsningar. Hibernate undansparar vanligen inte data från oallokerade delar av minnet. Oallokerat minne kan innehålla relevant data för kriminaltekniska undersökningar. Ett ytterligare problem är att hibernatefilen lagras i måldatorns hårddisk. Detta är i strid mot vedertagna forensiska undersökningsprinciper.

Ordlista

Term	Definition
Behörighet	En användares tilldelade rättighet att använda en informationstillgång på ett specificerat sätt (SIS HB 550).
Data	Representation av fakta, begrepp eller instruktioner i form lämpad för överföring, tolkning eller bearbetning av människor eller av automatiska hjälpmedel (SIS HB 550).
Datakvalitet	Egenskap hos data som behandlas enligt specifikation och inte ändras eller påverkas av fel i databehandlingen (SIS HB 550).
Digital	Värden angivna med hjälp av sifferuttryck i ett sammanhang som rör hur data representeras och lagras (Svenska datatermgruppen).
Digitalt bevis	Någon eller alla digitala data som kan fastställa att brott har begåtts eller kan ge en koppling mellan ett brott och dess offer eller ett brott och dess gärningsman (Casey E. 2004a, översatt till svenska).
Information	Innebörd av data. Data måste tolkas för att information skall erhållas (SIS HB 550).
Informationskvalitet	Användbarhet av information, också med hänsyn till dess effekt, för en given användare och ett givet problem (SIS HB 550).
Integritet	Okränkbarhet med förmåga att upprätthålla sitt värde genom skydd mot oönskad förändring, påverkan eller insyn (SIS HB 550).
Hot	Möjlig, oönskad händelse med negativa konsekvenser för verksamheten (SIS HB 550).
Måldator	Den dator som är föremål för undersökning. Även den dator som är mål för en attack.
Målfil	Den datafil som programvara för datainsamling skapar.
Relevans	Överensstämmelse mellan användarens informationsbehov och tillgängliga data (SIS HB 550).

Riktighet	Egenskap att informationen inte obehörigen, av misstag, eller på grund av funktionsstörning har förändrats (SIS HB 550).
Risk	Ett statistiskt väntevärde för oönskade händelser.
Rootkit	Program för datorintrång.
Spårbarhet (Accountability)	Möjlighet att entydigt kunna härleda utförda aktiviteter i systemet till en identifierad användare (SIS HB 550).
Säkerhet (Security)	Egenskap eller tillstånd som innebär skydd mot risk i samband med insyn, förlust eller påverkan (SIS HB 550).
Säkerhetsfunktioner	Specifik teknisk egenskap hos ett system som svarar för viss del av säkerheten (SIS HB 550).
Tillförlitighet (Reliability)	Mått på i vilken grad systemet levererar den information av given kvalitet den säger sig leverera samt tilltro till nivån (SIS HB 550).
Tillgänglighet (Availability)	Möjlighet att utnyttja informationstillgångar efter behov i förväntad utsträckning och inom önskad tid (SIS HB 550).

Förkortningar

Förkortning	Betydelse
BIOS	Basic Input/Output System
CPU	Central processing unit
DD	Data definition
DLL	Dynamic link library
DMA	Direct memory access
FAU	Forensic Acquisition Utilities
GUI	Graphical user interface
PCI	Peripheral component interconnect

PDA	Personal digital assistants
RAM	Random access memory
RKP	Rikskriminalpolisen
SKL	Statens kriminaltekniska laboratorium

Referenser

Artiklar

Adams J. et.al (2006). *Gap Analysis: Judicial Experience and Perception of Electronic Evidence*. Journal of Digital Forensic Practice. Vol. 1. S 13-17

Adelstein F. (2006). *Live Forensics: DIAGNOSING YOUR SYSTEM WITHOUT KILLING IT FIRST*. Communications of the ACM. Vol. 49. S. 63–66.

Carrier B. (2006). *Risk of LIVE DIGITAL FORENSIC ANALYSIS*. Communications of the ACM. Vol. 49. S. 56–61.

Carrier B. & Grand J. (2003). *A hardware-based memory acquisition procedure for digital investigations*. Digital Investigation. Vol. 1. S. 50-60.

Dickson M. (2006). *An examination into MSN Messenger 7.5 contact Identification*. Digital Investigation. Vol. 3. S. 79-83.

Panda B. et al. (2006). *Next-Generation CYBER FORENSICS*. Communications of the ACM. Vol. 49. S. 44–47.

Schuster A. (2006). *Searching for processes and threads in Microsoft Windows memory dumps*. Digital Investigation. Vol. 3S. S. S10-S16.

Scientific Woring Group on Digital Evidence (SWGDE). Forensic Science Communication (2000). *Digital Evidence: Standard and Principles*. Vol. 2. Number 2.

Böcker

Brorsson M. (1999). *Datorsystem Program- och maskinvara.* Lund: Studentlitteratur. ISBN 91-44-01137-7.

Casey E. (2004a). *Digital Evidence and Computer Crime*. San Diego: Academic Press. ISBN 978-0-12-163104-8.

Casey E. (2004b). *Handbook of Computer Crime Investigation*. San Diego: Academic Press. ISBN 0-12-163103-6.

Dealtry T R. (1992). *Dynamic SWOT analysis*. Birmingham: Dynamic SWOT Associates. ISBN 0-9523007-0-2.

Farmer D. & Venema W. (2004). *Forensic Discovery*. Upper Saddle River, NJ: Addison-Wesley. ISBN 0-201-63497-X.

Holme I.M. & Solvang B.K. 1997. *Forskningsmetodik.* Lund: Studentlitteratur. ISBN 91-44-00211-4.

Nyberg R. 2000. *Skriv vetenskapliga uppsatser och avhandlingar med stöd av IT och Internet.* Lund: Studentlitteratur. ISBN 91-44-01000-1.

Rienecker L. & Stray Jørgensen P. 2004. *Att skriva en bra uppsats.* Lund: Studentlitteratur. ISBN 91-47-06217-7.

Sommerville I. (2004). *Software Engineering*. Essex: Pearson Education Limited. ISBN 0-321-21026-3.

Tanenbaum A. & Woodhull A. (1997). *Operating systems: design and implementation*. 2nd ed. Upper Saddle River, NJ: Prentice-Hall. ISBN 0-13-638677-6.

Østbye H. et al. 2004. *Metodbok för medievetenskap*. Trelleborg: Liber. ISBN 91-47-07350-0.

Internetlänkar

Carnegie Mellon University (2006). *Virtual Training Environment*. Hämtat från <https://www.vte.cert.org/vtelibrary.html>. Publicerat 25 februari 2006. Hämtat 26 oktober 2006.

The Electronic Evidence Information Center. <http://www.e-evidence.info/>

Gleason BJ & Fahey D. (2006). *Helix 1.7 for Beginners*. [PDF] Hämtat från <http://www.efense.com/helix/Docs/Helix0307.pdf>. Publicerat 7 mars 2006. Hämtat 31 oktober 2006.

Microsoft Knowledge Base article 254649. *Overview of memory dump file options for Windows Server 2003, Windows XP, and Windows 2000.* Hämtat från <http://support.microsoft.com/kb/254649>. Publicerat 30 oktober 2006. Hämtat 18 november 2006.

Microsoft TechNet. *Device\PhysicalMemory Object.* Hämtat från < http://technet2.microsoft.com/WindowsServer/en/library/e0f862a3-cf16-4a48-bea5-f2004d12ce351033.mspx?mfr=true>. Hämtat 19 november 2006.

Microsoft TechNet. *Windows Sysinternals.* Hämtat från <http://www.microsoft.com/technet/sysinternals/default.mspx>. Hämtat 26 november 2006.

Russinovich M. & Cogswell B. (2006). *Sysinternals Filemon for Windows v7.03.* Hämtat från <http://www.sysinternals.com/Utilities/Filemon.html>. Publicerat 13 juli 2006. Hämtat 26 oktober 2006.

Russinovich M. & Cogswell B. (2006). *Sysinternals Regmon for Windows v7.03.* Hämtat från <http://www.sysinternals.com/Utilities/Regmon.html>. Publicerat 13 juli 2006. Hämtat 26 oktober 2006.

Svenska datatermgruppen. Sökord: digital. Hämtat från <http://www.nada.kth.se/dataterm/index.html>. Hämtat 20 oktober 2006.

Vidas T. (2006). *Forensic Analysis of Volatile Data Stores*. [PDF]. Hämtat från <http://www.certconf.org/presentations/2006/files/RB3.pdf>. Hämtat 26 november 2006.

Vidström A. (2006). *Forensic memory dumping intricacies - PhysicalMemory, DD, and caching issues*. Hämtat från < http://ntsecurity.nu/onmymind/2006/2006-06-01.html>. Hämtat 18 november 2006.

Lexikon och standarder

D. Brezinsk. et al. (2002). *Guidelines for Evidence Collection and Archiving*. Network Working Group. RFC: 3227.

SIS HB 550 (2003). *Terminologi för Informationssäkerhet.* Stockholm: SIS Förlag AB. ISBN 9171625763.

Shirey R. (2000). *Internet Security Glossary*. Network Working Group. RFC: 2828.

Programvara

Forensic Acquisition Utilities (FAU) George M. Garner Jr.
Program: dd.exe
Länk: <http://users.erols.com/gmgarner/forensics/develop/> Hämtat 20 oktober 2006

Process Monitor v1.01
Länk:
<http://www.microsoft.com/technet/sysinternals/processesandthreads/processmonitor.mspx>. Publicerad 9 november 2006. Hämtad 26 november 2006.

Examensarbeten

Falk P. & Lindblom R. (2005). Stockholms universitet. Institutionen för data- och systemvetenskap. *Loggar som bevisning.*

Bilaga

Intervju med åklagare Håkan Rosvall

Intervjun genomfördes vid internationella åklagarkammaren Stockholm den 7 november 2006.

Hur utbildas domare och åklagare i frågor som rör internet och datorer?
Domare informeras i viss utsträckning. Vissa domare har skaffat kunskap av privat intresse.

År 1999 genomgick tio åklagare en teknisk utbildning. TCP/IP var huvudfokus. Fem åklagare jobbar idag särskilt med IT-brott.

Hur definieras IT-brott?
Dator är mål för brott eller dator är hjälpmedel för att begå brott.

Hur är medvetenheten kring risker som förknippas med digitala bevis bland domare och åklagare?
Medvetenheten är obefintlig. Man kan ej värdera riskerna om man inte kan tekniken.

Hur vanligt förekommande är det att digitala bevis åberopas i rätten?
Det är ett ständigt ökande antal ärenden.

Vad påverkar bevisvärderingen av ett digitalt bevis?
Hur systemet har tagits ner. Hur bevisen har hanterats sedan beslaget, hur bevisen förvarats, miljön vid spegling.

Vilka krav ställs på dokumentationen av de metoder och den teknik som används för insamling av digitala bevis?
Det finns inga färdiga krav idag. Det är upp till varje forensiker.

Hur vanligt förekommande är det att digitala bevis ifrågasätts i rätten?
Förbluffande sällan. Några exempel kan nämnas har förekommit. Fjärrstyrningar som invändning. Encase har också ifrågasatts vid något fall, men källkoden kan ju inte lämnas ut.

Är det möjligt att vi husrannsakan offra bevismaterial som enda utväg för att kunna säkra annan mer värdefull bevisning?
Ja. Skadeståndsansvar kan bli konsekvens om man ”förstör” någon information. Något skadeståndsanspråk har hittills inte förekommit.

Vilka juridiska begränsningar finns vid en husrannsakan då man vill säkra digitala bevis?
Proportionalitetsprincipen; intrång kontra nyttan. Normalt tar man inte med datorer från arbetsplatser. Hänsyn tas då tredje man kan drabbas.

Utdrag ur hjälpsidorna för DD

Usage: dd if=[SOURCE] of=[DESTINATION] [OPTIONS]

Copy a device or one or more files or streams, converting and formatting according to the options specified:

```
 bs=[BYTES]       Set 'ibs' and 'obs' equal to BYTES.
 conv=[KEYWORDS]  Convert the input as per the comma separated keyword list.
 count=[BLOCKS]   Copy only the specified number of input blocks.
 ibs=[BYTES]      Sets the input block size.
 if=[SOURCE]      Specifies the source for input; the default is stdin.
 obs=[BYTES]      Sets the output block size.
 of=[DESTINATION] Specifies the destination for output; the default is stdout.
 seek=[BLOCKS]    Skip the specified number of obs-sized blocks at start of output.
 skip=[BLOCKS]    Skip the specified number of ibs-sized blocks at start of input.

   -g --gather   Append multiple input files to a single output file.
   -a --append   Append input to the output file.
   -r, --recursive   Recursively search subdirectories for files to copy.  Valid only if 'if' specifies a
search pattern.
   --help        Display this help and exit.
   -v  --vebose  Output verbose information.
  --cryptsum [ALGORITHM]    Includes one or more cryptographic
                            checksums in the output.  "md2", "md4", "md5""
                            and "sha" or "sha1" are supported on all platforms
                            "sha_256", "sha_384" and "sha_512" are supported
                            on Windows Server 2003 and later.
      --cryptout [FILE] Write cryptographic checksum to the specified file.
      --verify      Verifies the cryptographic checksum of the output file.
      --sparse      Makes the output file sparse (ntfs only).
      --log [FILE]  Write log output to FILE.
      --lockin      Lock input file while copying.
      --lockout     Lock output file while copying.
      --volumelabel [VOLUME_LABEL]    Send output to a volume on a removable
                                      drive with the specified volume label.  If '--volumelabel' is
                                      specified, the volume name is prepended to the path specified
                                      by 'of'.
      --eject                 Dismount and, if possible, eject the volume specified
                        by the '--volumelabel' option.
      --localwrt        Enables writing output to a local fixed drive.
      --restore_access_times Restores file access times on the source.
      --locale [LANG]  Specifies the output locale.
      --seek [BYTES]   Skip the specified number of bytes at start of output.
      --skip [BYTES]   Skip the specified number of bytes at start of input.
      --count [BYTES]  Stop after acquiring the specified number of bytes.
```

--chunk [BYTES] Set the maximum size of the output file. If the output file
exceeds the specified size, the file is split into multiple
fragments of BYTES bytes in size.

BYTES may be suffixed: by xN for multiplication by N, by c for x1,
by w for x2, by b for x512, by KB for x1000, by KiB for x1024,
by MB for x1,000,000, by MiB for x1,048,576,
by GB for x1,000,000,000, by GiB for x1,073,741,824
by TB for x1,000,000,000,000, by TiB for x1,099,511,627,776

BYTES may be prefixed by "0x" or "x" to indicate a hexadecimal value.

Each KEYWORD may be:

noerror	Continue after read errors.
comp	Compress output.
decomp	Decompress input.

The following options may be used in conjunction with a search pattern
to select the files or streams that are to be processed:
-A, --attributes hashes files with specified attributes:

attributes	D Directories	R	Read-only files
	H Hidden files	A	Files ready for archiving
	C Compressed files	E	Encrypted files
	O Offline files	P	Sparse files
	S System files	~	Prefix meaning not
	T Temporary files		

--any Specifies how the -A --attribute option is to be
interpreted. With '--any' files or streams with
any one of the specified attributes will be processed.
The default is to hash files with all of the specified
attributes.

The following are used to select files based upon file times:
--modified [FILETIME] selects files based upon the file
modification time.
--accessed [FILETIME] selects files based upon the file access time.
--created [FILETIME] selects files based upon the file creation
time.

The format of the FILETIME string is specified according to the locale
of the current user. For example, 10:00PM June 6, 2003 is specified
as "6/10/2003 10:00PM" in the United States and "10/6/2003 10:00PM"
in most european countries. The file time string may be pre-pended by <, = or >
to search for file times that are less than, equal or greater than

the specified time string. The FILETIME string may include multiple conditions separated by a semi-colon (;). Multiple conditions are evaluated in pairs. An un-paired condition is evaluated individually.

The following may be used to select directories, files or streams based upon specified retular expressions:

--directoryfilter [EXPRESSION] selects directories based upon an expression.
--filefilter [EXPRESSION] selects files based upon an expression.
--streamfilter [EXPRESSION] selects alternate streams based upon an expression
--magicfilter [EXPRESSION] selects files or alternate streams based upon the binary contents at the start of the data stream as evaluated by an expression.

EXPRESSION may be any regular expression. Double quotes ("") may be used to prevent the command interpreter from splitting a single expression into two or more pieces. With respect to the --magicfilter option, EXPRESSION is limited to a regular expression that may be converted to a single byte character set.

The following may be used to select files or streams based upon the entropy of the initial data stream:

--entropy [THRESHOLD] selects files or streams whose initial data streams have an entropy greater than THRESHOLD.

The following may be used in conjunction with output in xml format:
--xml creates hash output in xml format.
--case [CASE NUMBER] includes the specified case number in xml output.
--evidence [EVIDENCE NUMBER] specifies one less than the initial evidence number.
--description [DESCRIPTION] includes an optional description in xml output.

The following option may be used to set the thread priority of the program:

--thread_priority [PRIORITY] sets the priority of the thread processing.

The thread priority may be set to any of the following values:
idle, lowest, belownormal, highest
abovenormal, timecritical

--ata_unlock [PASSWORD] unlocks an ATA drive using the provided password.
--ata_master specifies that the password provided with --ata_unlock is a master password.
--ata_hpa temporarily disables the ATA host protected area if it exists

and sets the starting offset to skip the user accessible bytes.
--ata_restore_configuration resets an ATA device configuration overlay (DCO) and restores the original drive configuration.