jamk.fi

Intrusion Detection During IT Security Audits

Mikko Vatanen

Master's thesis
November 2014

Master's Degree Programme in Information Technology
Technology, communication and transport

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Vatanen, Mikko	Master's thesis	3.11.2014
		Language of publication: English
	Number of pages: 94	Permission for web publication: x

Title of publication

Intrusion Detection During IT Security Audits

Possible subtitle

Degree programme

Master´s Degree Programme in Information Technology

Tutor(s)

Rantonen,Mika, Hautamäki Jari

Assigned by

Finnish Communications Regulatory Authority

In many public cases intruders have managed to remain undetected inside victim's network or systems a very long time. Sometimes it has taken over two years until detection was made and during that that time most of the targets have most probably been monitored and audited. Whether audit or monitoring process has failed, or used tools and processes don't cover intrusion detection.

This thesis aimed at narrowing the gap between IT security audit and intrusion detection by inspecting attack methods and signs of intrusions. Audit tools were then developed to cover the findings and were then tested on live environments. Current audit process and it's affects to audit results was analyzed and purpose was to find necessary changes. Thesis concentrated on finding anomalies from operating systems and network traffic within given guidelines of audit processes and applicable laws. Audit process and law was inspected from Finland's perspective, but technical research area was worldwide.

Keywords

Audit, Intrusion, Linux, Network, Privacy, Protocol, Windows

Miscellaneous

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Vatanen, Mikko	Opinnäytetyö	3.11.2014
	Sivumäärä 94	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Tunkeutumisen havainnointi tietoturva-auditoinneissa

Koulutusohjelma

Master´s Degree Programme in Information Technology

Työn ohjaaja(t)

Mika Rantonen, Jari Hautamäki

Toimeksiantaja(t)

Viestintävirasto

Monissa julkisuuteen tulleissa tapauksissa tunkeutujat ovat voineet toimia uhrin verkoissa tai järjestelmissä kauan aikaa. Joskus havainnon tekemiseen on mennyt jopa kaksi vuotta ja tänä aikana kohdetta on mitä luultavimmin monitoroitu ja auditoitu. Joko auditointi- tai monitorointiprosessi on pettänyt, tai käytettävät työkalut ja menetelmät eivät kata tunkeutumisen havaitsemista.

Lopputyö tähtäsi tietoturvallisuus auditoinnin ja tunkeutumishavainnoinnin välisen eron kaventamiseen tutkimalla hyökkäysmenetelmiä ja sen aiheuttamia jälkiä. Auditoinnissa käytettyjä työkaluja kehitettiin kattamaan löydetyt havainnot ja niitä testattiin oikeissa ympäristöissä. Nykyinen audit-prosessi ja sen haittavaikutukset auditointituloksiin tutkittiin ja tarkoitus oli löytää mahdolliset tarvittavat muutokset. Lopputyö keskittyi poikkeamien löytämiseen käyttöjärjestelmistä ja verkkoliikenteestä annettujen auditointi ohjeiden ja soveltuvien lakien asettamilla reunaehdoilla. Auditointi-prosessia ja lakia tutkittiin Suomen näkökulmasta, mutta teknistä osa-aluetta tutkittiin kansainvälisesti.

Auditointi, Tunkeutuminen, Linux, Tietoliikenne, Protokolla, Yksityisyys, Windows

Muut tiedot

ACRONYMS

AD	Active Directory
APT	Advanced Persistent Threat
ASLR	Address space layout randomization
C&C	Command&Control
CSV	Comma Separated Value
DEP	Data Execution Prevention
DGA	Domain Generation Algorithm
DLL	Dynamic Link Library
DNS	Domain Name System, Domain Name Service
EMET	Enhanced Mitigation Experience Toolkit
FICORA	Finnish Communications Regulatory Authority
GID	Group ID
GRE	Generic Routing Encapsulation
HKCC	HKey_Current_Config
HKCU	HKey_Current_User
HKCR	HKey_Classes_Root
HKLM	HKey_Local_Machine
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification/Identity/Identifier
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
ISACA	Information Systems Audit and Control Association
ISC	Internet Security Consortium
MAC	Media Access Control address
MSI	Microsoft Installer
NSA	National Security Agency
PID	Process Identifier
SAM	Security Accounts Manager
SFC	System File Check
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOR	The Onion router

UAC	User Account Control
UDP	User Datagram Protocol
UID	User ID
USB	Universal Serial Bus

CONTENTS

Tables

1. Introduction

Motivation

In most publicly reported cases, the attackers had already been inside the compromised systems for weeks, months, or even years before the intrusions were finally detected. Similar cases have been found in Finland and in other countries and organizations all over the world. One public example is from Japanese Finance Ministry that was infected with trojans. According to the news, the trojan was free to steal confidential data from January 2010 to November 2011. (Techworld, 2012). That is almost two years and probably plenty of information was lost.

Sometimes intrusions have been made by professionals, but in some cases there has simply been malware or trojan infections that has not been noticed by anyone. During this time systems have probably been monitored by administrators or even audited by internal or external organizations. Either there has not been proper monitoring by administrators, auditing included only paper review or technical audit focused only on compliance. It is possible that signs of intrusions were there and visible in the monitoring systems but were simply ignored by administrators or auditors for some reason. It is also possible that audit method and tools did not enable detection of intrusions or other malicious activity.

1.1. Threats and Mitigations

Some networks or information systems are disconnected from the Internet. This usually means that there is no direct interaction with the Internet, but information always flows there and back, through security gateways, removable devices or due to the actions of the users or administrators. Therefore even a closed system can be infected or misused.

Nowadays information systems that are connected to the Internet rely on security offered by firewalls, proxy/antivirus servers and Intrusion detection, IDS and Intrusion Prevention, IPS systems. If policies are well applied, logs are collected statistical reports about events generated from them; however, it is good to notice that perimeter defense is not always the key to detection or prevention of intruders.

> *In some cases, compromised servers are hosted on target organizations' networks after successful infiltration. This means that network defenses placed at the perimeter will not detect standard IXESHE (name of the APT) network traffic because communication occurs internally. The attackers can communicate through an alternate means with the internal C&C server in order to avoid detection. (Trendmicro 2012)*

Perimeter defense is efficient if the traffic flows through perimeter equipment that actually inspects the traffic instead of just acting as port filters and log collectors. Statistical analysis of traffic is a must, however, sleeping malware or sparsely communicating APT means that anomalies stay under the radar without raising alarms. Additionally, when low activity is combined with use of normal communication protocol features, the chances to detect advanced threats is almost zero. According to news and security conference subjects the point of interest is APT, Advanced Persistent Treat.

"It is a well-known fact that virus scanners do not find APT type malware. Based on CERT-FI's experience, most APT incidents are discovered either by accident or after a notification obtained from friendly third parties." (CERT-FI APT-ohje)

There are many articles and papers about APT detection. They usually provide examples for detection of specific APTs, however, the message is always the same: deeper statistical and technical analysis has to be made. (Trendmicro, 2012) If an attack is well made, it is typically not detected by security products like firewalls, antivirus, intrusion detection or intrusion prevention systems. Signs of the possible intrusion of

infections are not necessarily noticed without outside notification or help. Since intrusions have not been detected, attackers have plenty of time to complete their tasks.

> *Once they are inside the victim organization, attackers typically have plenty of time to reach their ultimate objective – whether that's stealing intellectual property or financial assets. The median number of days from the first evidence of compromise to when the attack was identified was 416 days. (Mandiant, 2013)*

According the report, the median number of days from compromise to detection can take over a year and in some cases many years.

1.2. Scope of Thesis and Implemented Research Method

The thesis is aimed at infiltration detection within main operating systems and to a passive network analysis. The applications and their configurations were left out, because it cannot be predicted what applications are installed beforehand, and the amount of application specific checks would be enormous.

To draft Proof of Concepts from the found hacking methods the following setups were inspected.

- Windows 7 SP1 64 bit, joined to domain
- Windows 8 64 bit, joined to domain
- Windows 2012 server 64 bit, Active directory with DHCP, IIS and DNS services
- Windows 2008 server R2 64 bit, Active directory with DHCP and DNS services

- Ubuntu server 12.04, Squid proxy
- Kali linux (debian),
- Ubuntu Desktop linux 12.04
- Centos Linux

Windows XP and Windows 2003 server were left out because they were running out of official support during the thesis writing process. The researched hacking tricks are presented as examples and do not include all possible methods that hackers use. The presented sample scripts and commands are meant to serve only as examples of different data gathering methods; however, they can be used to check similar details since the command or concept of the script is the same.

The research method implemented was an empirical study of system flaws, hacking methods and signs they leave. While different results were found they were compared to used audit process and tools to check if there already was a detection method. If a detection method was not found, an experimental method was used to test the issue on test environment and the method was developed in practice. The developed tools were then tested during actual audits.

2. Definition of Audit

2.1. Audit

International Information Systems Security Certification Consortium, (ISC)² defines audit as an independent review and examination of system records and activities that test for the adequacy of system controls, ensure compliance with established policy and operational procedures, and recommend any indicated changes in controls, policy, and procedures. Whether the question is about Information systems audit or security audit it is a matter of compliance.

2.2. Security Audit

The goal of security audit is to help the target system reach or maintain a certain security and trust level.

> *A security audit is a systematic evaluation of the security of a company's information system by measuring how well it conforms to a set of established criteria. A thorough audit typically assesses the security of the system's physical configuration and environment, software, information handling processes, and user practices. Security audits are often used to determine regulatory compliance, in the wake of legislation. (Searchcio.techtarget.com)*

Security audit might include penetration testing where a system or an organization is tested for their real ability to prevent and detect intrusions. This typically depends on the target whether it is a specific system or a whole IT infrastructure. Some systems have requirements that the security audit has to be passed successfully before it can be taken into production. However, the major part of the cases are systems that have never been technically audited or audited at all since the security culture has been missing.

2.3. Audit Process

An audit process can be carried out in many ways; however, certain frameworks have been developed to keep the audit methods and results comparable. The most famous of them is Control Objectives for Information and Related Technology (COBIT) framework created by Information Systems Audit and Control Association, ISACA. ISACA is an independent, nonprofit, global association. ISACA engages in the development, adoption and use of globally accepted knowledge and practices for information systems. One of the practices is information security audit and table 1 presents their classifications of audit..

Table 1 Broad classification of IS audit according to ISACA's web site

(The IS audit process, Isaca.org)

Audit target	**Controls**
Physical and environmental review	Physical security, power supply, air conditioning, humidity control and other environmental factors.
System administration review	Security review of the operating systems, database management systems, all system administration procedures and compliance.
Application software review	Application could be payroll, invoicing, a web-based customer order processing system or an enterprise resource planning system that actually runs the business. Review of such application software includes access control and authorizations, validations, error and exception handling, business process flows within the application software and complementary manual controls and procedures. Additionally, a review of the system development

	lifecycle should be completed.
Network security review	Review of internal and external connections to the system, perimeter security, firewall review, router access control lists, port scanning and intrusion detection are some typical areas of coverage.
Business continuity review	Existence and maintenance of fault tolerant and redundant hardware, backup procedures and storage, and documented and tested disaster recovery/business continuity plan.
Data integrity review	The purpose of this is scrutiny of live data to verify adequacy of controls and impact of weaknesses, as noticed from any of the above reviews. Such substantive testing can be done using generalized audit software (e.g., computer assisted audit techniques).

ISACA maintains an accepted concept of the audit; however, there are differences here between audit processes among countries and organizations.

2.4. Audit Process in Finland

In Finland, official audit guidelines are maintained by Finnish Communications Regulatory Authority, FICORA. These guidelines have to be used by companies or organisations that have gained an official approval to carry out information security audits within governmental organizations. (Ohje tietoturvallisuuden arviointilaitoksille). The guideline divides an audit process into two sections : Administrative and Technical.

Administrative Audit

The purpose of an administrative audit is to verify that a system is documented and to discover how the security responsibilities are organized. Table 2 below presents the required targets of an administrative audit.

Table 2 Targets of administrative audit

Audit target	Details
Interviews	At least following persons and/or groups are interviewed: maintenance staff, development and users
Documentation	At least following documentations are reviewed: network diagrams, system documentations, data flow diagrams, risk analysis etc.

Technical Audit

The purpose of a technical audit is to verify that the target is secure and in compliance with requirements. If there are specific security requirements like minimum password length, they are checked accordingly. Table 3 presents the required targets of a technical audit.

Table 3 Targets of technical audit

Audit target	**Details**
Passive analysis	Network traffic to and from the system being audited is captured and analyzed. Purpose is to verify network diagrams and system behavior as pictured in system documentation.

Audit of system configurations	Operating system, software, hardware and application configurations are reviewed. Configurations are checked to verify that systems implement the required security.
Active interface analysis	Operating systems, software or network interfaces are scanned for open ports, protocols and services. Vulnerabilities are discovered by using suitable vulnerability scanners or tools. Fuzzing is required at certain security levels or if there are availability requirements. Fuzzing means discovery of unknown vulnerabilities, coding errors and security loopholes in software, operating systems or networks within the system by inputting random data to the system in an attempt to make it crash.
Software security	Access controls of web applications, client and server applications or software are audited.
Encryption requirements and deployment	Target might have specific encryption requirements. Protocols, quality of encryption keys, key handling procedures and actual realization of encryption are audited.
Availability testing	Availability testing can be combined with fuzzing, but also includes review and possible tests of system recovery and backup plans.
Physical security	Implementation of physical security in structures, doors, windows and alarm systems are audited.

Security gateway	Communication between different information systems that have different security levels might have to be tightly controlled. One real life example of implementation is data diode, unidirectional security gateway. Finland has official guide for security gateway implementation (Yhdyskäytäväratkaisuohje, Viestintävirasto.fi).
Test of anomaly detection capabilities	Systems, people and organization are tested for capabilities and processes of detecting different kind of anomalies like viruses, data theft or attempts to break into the information systems.
TEMPEST	If required TEMPEST is measured against emanation leaks, including unintentional radio or electrical signals, sounds and vibrations.
Unknown devices	Physical review of target being audited to find extra devices or equipment.

The tools and methods for achieving the previous goals are not specifically guided except for TEMPEST where guidelines exist but they are not public. The execution order of a technical audit depends on the auditor, however, typically the administrative section is executed first. Table 4 is a sample list of commonly used audit tools and their purpose.

Table 4 Commonly used audit tools and their purpose

Tool	Purpose
Tcpdump, (http://www.tcpdump.org) Wireshark, (http://www.wireshark.org) Clarified network analyzer, (https://www.clarifiednetworks.com/Clarified%20Analyzer	Network traffic analysis and network topology analysis.
Microsoft Security Compliance Manager, https://technet.microsoft.com/en-us/library/cc677002.aspx	Compliance audit of Windows security settings.
Nessus (www.tenable.com/products/nessus) Nmap (www.nmap.org)	Portscanning and vulnerability detection over the network.
Radamsa https://www.ee.oulu.fi/research/ouspg/Radamsa Defensics, www.codenomicon.com/clarified/	Availability and software robustness testing.
Burp Suite, www.portswigger.net/burp/ OWASP Zed Attack Proxy (ZAP), https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project FuzzDB, https://code.google.com/p/fuzzdb/	Web application security and access control testing.
Sledgehammer, http://www.hammersource.com/Sledge_Hammers	Physical security
Protocol/Data testing or tunneling tools (Netcat, Hping, Scapy, Python)	Implementation of network security like security gateways, firewall filtering.

Test virus EICAR , www.eicar.org	Official antivirus and alarm testing file.
Suitable Denial of Service (DOS) tool Hping, Slowloris.pl, LOIC (Low Orbit Ion Canon) and variants	Availability testing and deployment of network and application security.
Manual testing, Selfmade scripts	Compliance and configuration verification.

2.5. Audit Results

Typical audit results consist of found vulnerabilities, flaws in used software or product or overall security implementation. Naturally there are different levels of flaws that can be presented from low to critical status. Some results are simply presented as pass or fail status if there are specific requirements for specific security controls. Administrative results are compared to technical results and vice versa to see if they co-operate. The results basically explain how well the target system is in compliance with its security requirements and if there are any risks to be analyzed and handled accordingly.

3. Basic Concept of Attacks

3.1. Why Attacks Happen

There are many reasons why attacks occur. The reasons can be political, hacktivism, terrorism, vandalism, or even accidents. Some of the stolen data is revealed to the public after intrusion, however, but for certain attackers the data is more valuable when it is kept in secret and the attack, if not revealed, can go on. Attacks, if not automated, usually use the following steps as described in subsections below.

3.2. Reconnaissance

The attacker tries to get as much useful information of the target as possible. The information is gathered from public sources like Internet, network scanning or by phishing the users or administrators or even by dumpster diving. The goal of reconnaissance is to gather enough information to execute the next phase and find weak points, which can be hardware, software, humans or processes.

3.3. Exploitation

Once weak points have been discovered, target systems or computers are directly exploited or with the help of user actions. The exploit could be delivered using email, web sites or portable media. The exploitation does not need to occur inside the target's network but when users are using a public network or are out of the physical security zone. In some cases the attacker creates a trust relationship with persons, which might ease the task of exploit delivery.

3.4. Persistence

The attacker tries to gain foothold on the target system and remain persistent. This can be achieved with trojans, backdoors, rootkits, or APT malware. The malware might be distributed to only a few target computers instead of a whole network.

> *A rootkit is a piece of software that enables the continued, privileged access to a computer, all the while hiding its presence from users and administrators. Although rootkits themselves might not be dangerous, the software or processes they hide almost always are. Unlike a virus, a rootkit gains administrative privileges to your machine. The biggest issue with rootkits is that once on a system, they are a challenge to detect and remove, because their main purpose is obfuscation. (Jack Wallen, 2010)*

Malware can also be memory based *(*Lucian Constantin, 2012) and self-destructing if it detects examination.

3.5. Operation

An attacker steals and uses existing accounts, which generates less noise in the form of logs. The attacker can also create extra accounts to the systems. The attacker can create new services, spread backdoor, manipulate existing services or use existing services to hop from one machine to another, which again generates less noise at the network level and logs.

3.6. Data Infiltration

The stolen data is transferred from victim computers to attacker systems with the help of network protocols or portable media. Exfiltration can go through several machines and data source is not directly communicating to the attacker. The machines

that intercept the stolen data are somewhere on the Internet and the intruder cannot therefore be easily identified. Intercepting a machine can be an other target that has been taken over to collect data from other targets.

All the previous steps could be carried out on the hardware level, which is much harder to detect.

> *Leaked NSA documents have revealed that also special hardware (chipsets, removable storage and cables that uses wireless signaling etc.) is used by advanced attackers. (*Sean Gallagher*,2013)*

Hardware investigation requires special tools and probably a laboratory. This process would also affect on availability of the system being audited. This thesis concentrates on operating system and to non-wireless network traffic.

4. Imperfections in Audit Process

Signs of Hacking and Audit Process

In the light of public and closed data about intrusions and methods that are taught in the hacking courses, there are missing actions in the audit process. All items in the following list do not not apply to all auditors. The list is built up from observation of audit work in Finland and contains the most typical flaws. Also, the audited target might already execute necessary actions to fulfill some of the following actions. Typically this is a scheduled full antivirus scan.

- Full antivirus checks are not necessarily run during audit
- Historical data (alerts, logs) are not reviewed
- Systems that do not have antivirus installed are not checked for viruses
- Passive network analysis does not include active search of anomalies
- Analysis of the data captured during passive analysis has to be done manually
- Gathered data is lightly analyzed not actually compared to anything
- Integrity of systems is required but typically not verified
- Overall capability of the search of anomalies inside the systems is missing. There is knowledge and information about anomalies; however, it is not fully used to enhance audit work quality.

5. Legal Matters

5.1. Finnish Privacy Law

Finland has it is own privacy law called Act on the Protection of Privacy in Electronic Communications. The objective of the Act is to ensure confidentiality and protection of privacy in electronic communications and to promote information security in electronic communications and the balanced development of a wide range of electronic communications services. (Act on the Protection of Privacy in Electronic Communications~~)~~

The act limits the possibility and ways to monitor network traffic and user actions where data is private. However, Section 20 (125/2009) of the act gives some rights to undertake necessary measures to ensure information security during an audit. Section 20 of the Act is quoted here in full.

> *Section 20 (125/2009) - Measures taken to implement information security*
>
> *(1) A telecommunications operator, value added service provider or corporate or association subscriber, or any party acting on their behalf has the right to undertake necessary measures referred to in section 2 for ensuring information security:*
>
> > *1) in order to detect, prevent, investigate and commit to pre-trial Investigation any disruptions in information security of communications networks or related services;*
> > *2) in order to safeguard the communications ability of the sender or the recipient of the message; or*
> > *3) in order to prevent preparations of means of payment fraud referred to in Chapter 37(11) of the Penal Code planned to be implemented on a wide scale via communication services.*
>
> *(2) Measures referred to in subsection 1 above may include:*
>
> > *1) automatic analysis of message content;*

2) automatic prevention or limitation of message conveyance or reception;
3) automatic removal from messages of malicious software that pose a threat to information security;
4) any other comparable technical measures.

(3) If it is evident due to the message type, form or some other similar reason that the message contains a malicious software or command, and automatic content analysis of the message cannot ensure the attainment of the goals referred to in subsection 1, the contents of a single message may be processed manually. The sender and recipient of a message whose contents have been manually processed shall be informed of the processing, unless the information would apparently endanger the attainment of the goals referred to in subsection 1.

(4) Any measures referred to in this section shall be implemented with care, and they shall be commensurate with the seriousness of the disruption being combated. Such measures shall not limit freedom of speech, the confidentiality of a message or the protection of privacy any more than is necessary for the purpose of safeguarding the goals referred to in subsection 1. Such measures shall be discontinued if the conditions for them specified in this section no longer exist.

(5) The Finnish Communications Regulatory Authority may issue further regulations to telecommunications operators and value added service providers on the technical implementation of the measures referred to in this section.

The law justifies an automatic analysis of data. Manual analysis is possible if automatic content analysis fails to reach it is goal. A good example of this is communication protocol that is not recognized automatically, but does something abnormal.

5.2. Private Data and Audit

Private data could be gathered in several ways. Passive network monitoring, which may reveal Internet browsing or emails is one way. An audit of the workstation or server can reveal personal files or Internet history. Even with the given rights of the Act, privacy has to be respected and only necessary actions can be taken during examination. There are basically two kinds of audit variations and in these cases the

approach to the audit has to be distinct. The first case is where target being audited only has system owner's data. With the system owner's permission the system can be thoroughly audited.

The second case is where there is a possibility that a target has private files, private Internet history or private network traffic. In that case permission to a thorough audit has to come from a user or users of the system. If there is no permission, normal audit process applies and following steps should be taken:

- No personal file checks, except general activities like virus scanning
- Only general file searches like virus.exe.
- No searching traces that could reveal personal activities. These include browsing history, typed URL's, last opened files etc.
- Traffic analysis has to be carried out as automatically as possible and if personal data is revealed and the user is identified, he or she has to be notified.
- Found network addresses or DNS names that can be treated as a conversation partner in private communication, have to be handled in a way that they cannot be connected to any specific user.

5.3. Responsibilities and Limitations

If an audit reveals system breaches or other anomalies, the auditor is only allowed to notify system owner. In Finland, it is the system owner's responsibility to inform the police about data breach and the police are in charge of the investigation. Even In case of detection of things like child pornography, when data is in a personal computer or in a personal folder, the auditor is not necessarily allowed to inform anyone due to privacy law.

Usually time and resources for the audit are limited. Tools being used have to be proven to work and in some cases have to be first inspected by the target organization. That means that tools and methods have to be not only readable, but also quickly customizable by the auditor. Audit tools are also not supposed to make any changes to the target system. Therefore special antivirus or integrity checks have to be run in “verify only” mode, which also protects from data destruction in case of false positives.

6. Adding Anomaly-finding Capability to Audit Process

6.1. Anomaly Finding Capability

If normal audit is executed mostly only static checks are made, and some basic historical data is investigated. If one also wants to search anomalies or to see that a system has been used and maintained as agreed, extra analysis has to be made. Signs can be found from the system itself (files, registry, configuration), log analysis or current state of network. Signs to look for come from the known intrusion methods or known tricks that are used to deceive administrators or users.

Ideas to anomaly detection can be adopted from forensics and malware analysis methods and from learned hacking methods. The difference between auditing and forensics is that during audit one does not actively search for evidence and try to maintain the integrity of the evidence at the same time.

6.2. Computer Forensics

During computer forensics, systems are searched for intrusion evidence and investigators try also to build a timeline of the intrusion. Disk imaging and memory dumps are used to maintain integrity of the original data. Data is then searched in offline environment. Table 5 presents definitions used by U.S. Federal Bureau of Investigation to delineate certain aspects of computer forensic science.

Table 5 Aspects of computer forensic science

Evidence source	***Explanation***
Data objects	*Objects or information of potential probative value that are associated with physical items. Data objects may occur in different file formats (for example, NTFS or FAT32) without alteration of the original information.*
Digital evidence	*Information of probative value that is stored or transmitted in digital form.*
Physical items	*Items on which data objects or information may be stored and/or through which data objects are transferred.*
Original digital evidence	Physical items and the data objects associated with such items at the time of acquisition or seizure. *Duplicate digital evidence. An accurate digital reproduction of all data objects contained on an original physical item.*
Duplicate digital evidence.	An accurate digital reproduction of all data objectscontained on an original physical item.

Incident Response: Computer Forensics Toolkit
Chapter: Catching the Criminal: The Basics of Computer Forensics

Despite of duplication of data the same methods are used in the Finnish audit. Duplication during audit would mean that also similar hardware is required to run the machines, and that is often not possible. However, virtualization allows the auditor to run the target in a similar environment without affecting online system.

6.3. Malware Analysis

Approach to Malware Analysis

There are two fundamental approaches to malware analysis: static and dynamic. Static analysis involves examining the malware without running it. Dynamic analysis involves running the malware. Both techniques are further categorized as basic or advanced. (Practical malware analysis).

Basic static analysis consists of examining the executable file without viewing the actual instructions (Practical malware analysis). Advanced static analysis consists of reverse-engineering the malware's internals by loading the executable into a disassembler and looking at the program instructions in order to discover what the program does (Practical malware analysis).

Basic dynamic analysis techniques involve running the malware and observing its behavior on the system in order to remove the infection, produce effective signatures, or both (Practical malware analysis). Advanced dynamic analysis uses a debugger to examine the internal state of a running malicious executable (Practical malware analysis).

Memory analysis and disk imaging are problematic due to today's technical development like the amount of used memory, Address Space Layout Randomization or implementation of disk redundancy. Additionally, most times the auditing is executed on live production environment. Therefore during an audit it is better to carry out similar checks as in computer forensics and look for signs of abnormal behavior in the past. Malware behavior could be detected from system logs or from passive monitoring of the network. Malware analysis goes to very basic level: Are there extra processes and do they communicate with other machines? The malware can also destroy itself automatically if it detects examination even if it is not memory based.

From auditing perspective anomaly detection can be divided into three categories:

Bad System Usage or Maintenance

Login methods reveal used protocols, login sources and used account details. This data can be combined with login times and dates. Activity in strange times or excessive use of administrative or strange accounts should be investigated.

USB device history reveals if data has been copied from the system or to system with approved portable media. Device history usually reveals if there have been unknown devices connected to the system.

Network connection history can be analyzed with passive monitoring and reviewing logs from firewall or proxy. DNS records are valuable if they have been saved since a connection attempt does not necessarily happen before resolving of the domain or hostname. The devices connected to the network can be traced with passive monitoring, DHCP logs and scanning MAC addresses in the network. MAC addresses can be counted and mapped to vendor lists.

Installed updates and their install dates report if a system has been updated or maintained as described. If the system has not been updated in a long time, the risk of intrusion rises. Virus and malware detection and update history reveal if the organization and employees have reacted to virus alerts.

Anomalies could be detected by searching specific changes to the system or by analyzing the system's history (logs), user or network traffic behavior. The checklist of specific changes builds up from the files or settings that the auditor or the public knows and that can be changed or added to the system so that the system becomes insecure. Keeping the list updated requires gathering data about new methods all the time. Material can be gathered from forensics courses and by following hacking sites.

Abnormal System Changes

Abnormal system changes mean settings or changes that should not exist within the system. In Windows, registry is the place where most of the operating system level changes occur. Linux configuration is mostly file based and therefore the approach to that operating system has to be different. Both systems have systems-files that can be modified or replaced. Missing logs or extra files within the system should also be investigated.

Extra user accounts can exist in almost every system. The location of the account however is not necessarily local Active directory, AD or extra user account can be added to local accounts so that it is not necessarily detected from central management.

Network analysis can reveal protocol misuses like binding services to strange ports or data tunneling. Binding services to uncommon ports is not wrong or uncommon; however, most of the protection systems like firewall tend to inspect protocols incorrectly when their behavior has been modified.

Behavioral Analysis

Behavioral analysis needs both special security knowledge and help from the system administrator who hopefully knows the system's normal state. Data might have to be compared to something like a list of bad Internet Protocol, IP addresses, file signatures or knowledge of the administrator before the decision of further action is made. The analysis can be automatic or manual depending on how the data is gathered.

Usage of existing user accounts, details of installed device drivers, list of system services or analysis system and application errors can be more easily analyzed with the help of a knowledgeable system admin. Other points of interests are analysis of in-

stalled software, overall analysis of network traffic and protocols, connection attempts to various domains or Internet Protocol addresses and strange activity time periods of users and the system. Collected results can and should be compared against each other where it is useful.

7. Example Tools Used for Analysis

Audit Tools

Common requirements for audit tools are that they are reliable, portable and can be modified. Target operating system also sets limitations since not all programming languages are directly supported. Therefore tools or scripting languages that are supported or developed by the operating system developers are probably the safest approach.

7.1. Powershell

Microsoft Powershell scripting tool is supported by all modern Microsoft operating systems. Powershell is also easily modifiable, understandable and can be easily checked by target organizations since it is becoming the de-facto tool for maintenance. In addition, Powershell can also execute or call external commands and it is not that way limited to its own command-lets. Windows PowerShell is Microsoft's task-based command-line shell and scripting language used as configuration management framework. Powershell is built on .NET Framework. (Technet, 2014)

Caveats of Powershell is versioning and dependence in modules if extra checks for specific services have to be made. That might require modules that are not necessarily installed. Since one cannot predict what modules are installed on the target, common commands or command-lets have to be used. Powershell can also export and convert collected data to different formats for reporting purposes.

7.2. Linux Audit Tool

Building Linux tools is harder because every Linux distribution and installation can have different package managers, shell environments and available commands. Also, software versions or directory structure and location of system files may differ. Therefore, basically Linux has to be audited with shell commands and differently depending on the distribution.

Statistically Linux has much less malware and is not that often used for workstation purpose (except homes). Linux/Unix systems are typically used as servers, which means that a more probable incident against them is data theft.

7.3. Passive Network Analysis Tool

In Finland official audit process already includes passive network analysis and it can be carried out with live capture and analysis on the fly or by saving data to hard-drive that can then be analyzed later. After the empirical research phase following requirements were found.

The tool should be able to read and analyze large packet capture files and also be able to do an online analysis on the fly. The tool should recognize and support as many protocols as possible, and manual protocol mapping should be possible. Filtering of data should also be possible by keyword or by source or destination IP address and the results should be possible to be directed to another file for further analysis. The auditor should also be able to filter traffic in a way that privacy is respected when required. The tool should be portable in a way that analysis can be carried out by using the target's own workstation or server. If the traffic is encrypted, the tool should be able to report certificates and encryption handshake options.

Tshark

TShark is a network protocol analyzer related to Wireshark. Both tools can achieve the same results. The difference is that Tshark is command line based and can be automated to process large amounts of files and data. Packets can be captured from live network and analyzed on-the-fly or read from previously saved capture files. TShark's native capture file format is `libpcap` format, which is also the format used by Tcpdump and various other tools. (Tshark manual, wireshark.org). `Libpcap` is an Application interface to low-level network traffic monitoring. Read filters in TShark also allow selection of packets to be decoded and written to another file (Clint P. Garrison, 2010)

Ssldump

`Ssldump` can handle traffic encrypted with Secure Sockets Layer or Transport Layer Security protocols. It can automatically identify SSL or TLS connections and decodes the records and displays them in a textual form. If provided with the appropriate keying material, it will also decrypt the connections and display the application data traffic. (Ssldump manual, rtfm.com)
`Ssldump` can decrypt the connections with appropriate keying material, but that feature should not be used unless specifically required and never against private data without permission.

8. Windows Anomalies

8.1. Windows Protection

Windows is known to have lot of malware dedicated to it. Even though Microsoft has developed ways to protect applications from attackers (Data Execution Prevention, Address Space Layout Randomization, Enhanced Mitigation Experience Toolkit) or user actions, it is not perfect. Many changes can be made to alter the operating system to hide intruder's tracks; however, if an intruder can stay in the system for many years it means that nobody is actually watching logs or monitoring abnormal behavior. This is understandable when the amount of systems is high and the number of employees is low.

8.2. Anomalies from Windows Logs

By default Windows stores login information in its log files. Security policy has to be specifically set so that both successful and unsuccessful logins are logged to the security log. Event ID and the method of parsing the event depend on whether local or AD login is used. Also alternate authentication methods, like smartcard, go to different Event ID. Data that can be collected from events are user names, login time, logon type, source IP address, hostname etc. However this requires parsing the contents of the actual events and knowledge of the environment. This information is valuable when audited target is for example company network with lot of machines.

Information required for parsing the logs depends on the Event ID and the inspected log file. For example, the security log contains information about login type, which is expressed as a number value instead of words. This number can be mapped to an actual method so the result is easier to evaluate. Event logs contain also other de-

tailed information (login name, time, login method as presented in Table 6 etc.) of the event and these details are listed as rows inside the actual event with every row presenting it is own detail value. Rows are numbered beginning from zero. In the example Powershell code “ReplacementStrings(value)” defines the offset from the beginning and the value is then given a human interpretable name. Negative value (-1) means countdown beginning from the last row.

An example of detailed event data from a failed login:

```
  <EventData>
    <Data Name="SubjectUserSid">S-1-5-18</Data>
    <Data Name="SubjectUserName">2012DC$</Data>
    <Data Name="SubjectDomainName">VALHALLA</Data>
    <Data Name="SubjectLogonId">0x3e7</Data>
    <Data Name="TargetUserSid">S-1-0-0</Data>
    <Data Name="TargetUserName">administrator</Data>
    <Data Name="TargetDomainName">VALHALLA</Data>
    <Data Name="Status">0xc000006d</Data>
    <Data Name="FailureReason">%%2313</Data>
    <Data Name="SubStatus">0xc000006a</Data>
    <Data Name="LogonType">2</Data>
    <Data Name="LogonProcessName">User32 </Data>
    <Data
Name="AuthenticationPackageName">Negotiate</Data>
    <Data Name="WorkstationName">2012DC</Data>
    <Data Name="TransmittedServices">-</Data>
    <Data Name="LmPackageName">-</Data>
    <Data Name="KeyLength">0</Data>
    <Data Name="ProcessId">0xec4</Data>
    <Data
Name="ProcessName">C:\Windows\System32\winlogon.exe</Data
>
    <Data Name="IPAddress">127.0.0.1</Data>
```

```
    <Data Name="IPPort">0</Data>
  </EventData>
</Event>
```

Table 6 List of actual login methods

from `<Data Name="LogonType":</Data>` (Audit logon events, technet.microsoft.com)

Method number	Translation	Explanation
2	Interactive	A user logged on to this computer.
3	Network	A user or computer logged on to this computer from the network.
4	Batch	Batch logontype is used by batch servers, where processes may be executing on behalf of a user without their direct intervention.
5	Service	A service was started by the Service Control Manager.
7	Unlock	This workstation was unlocked.
8	NetworkCleartext	A user logged on to this computer from the network. The user's password was passed to the authentication package in its unhashed form. The built-in authentication packages all hash credentials before sending them across the network. The credentials do not traverse the network in plaintext (also called cleartext).

9	NewCredentials	A caller cloned its current token and specified new credentials for outbound connections. The new logon session has the same local identity, but uses different credentials for other network connections.
10	RemoteInteractive	A user logged on to this computer remotely using Terminal Services or Remote Desktop.
11	CachedInteractive	A user logged on to this computer with network credentials that were stored locally on the computer. The domain controller was not contacted to verify the credentials.

The method number could be mapped to human readable form with the following example function:

```
function LogonTypesAndNames {
        Param($LogonTypeNumber)
        switch ($LogonTypeNumber) {
        0 {"System"; break;}
        2 {"Interactive"; break;}
        3 {"Network"; break;}
        …
        …
        …
        13 {"CachedUnlock"; break;}
        default {"Unknown"; break;}
        }
        }
```

In the function the value of parameter `LogonTypeNumber` is matched to the name of the Logon type.

8.3. Different Logins from Security Log

When investigating Windows logs both successful and unsuccessful logins from local system and AD should be reviewed. The usage of administrative accounts is especially interesting. Other logins (successful or failed) that can be parsed with the same method by changing the queried event id (Description of security events, support.microsoft.com) can be reported similarly, but possibly with different offsets as in example.

Event ID that means successful login.
Event ID: 4624

Event ID that means unsuccessful login.
Event ID: 4776
Following is an example output of one login event that was done locally. The results were directed to a format separated by a comma. This example also includes the name of the process that was used by the system.

```
"12.2.2014
9:44","Interactive","Negotiate","administrator","2012DC",
"C:\Windows\System32\winlogon.exe","2012DC.VALHALLA.root"
```

If a list of the above kind of events is opened in spreadsheet program like Excel, data can then be filtered according to the used account name or source IP address or something else. Information, however, is not accurate in the means of used login account being used and used Event ID. A live test showed that non-AD computer that used exchange mail account (AD account) appeared in the AD servers local authenti-

cation log. The point is not, however, where the log can be found but what can be found, since the information included username, hostname, IP address and timestamp. In this case, users using machines that are not centrally managed can be found.

Logins Without Security Log

In case security policy is not set, successful logins can be found in the system log with event ID 7001. However, the system log gathers plenty of other events thus the available login history might not be long and it seems that only successful logins are logged.

```
$UserProperty = @{n="User";e={(New-Object Sys-
tem.Security.PrincIpal.SecurityIdentifier
$_.ReplacementStrings[1]).Translate([System.Security.Prin
cIpal.NTAccount])}} $TypeProperty =
@{n="Action";e={if($_.EventID -eq 7001) {"Logon"} else
{"Logoff"}}}
$SourceProperty = @{n="Source";e={$_.Source}}
$MachineProperty = @{n="Machine";e={$_.MachineName}}

Get-EventLog System -Source Microsoft-Windows-Winlogon -
ea SilentlyContinue | select $UserProper-
ty,$TypeProperty,$TimeProperty,$SourceProperty,$MachinePr
operty
```

In the code the values from log files are mapped to according properties. These properties are then used to filter data for the output. The output differentiates logon and logoff events.

8.4. Errors from System and Application Log

Errors from system and application logs indicate system or software instability, which can be a sign of normal badly made software, however, also a sign of malware. Errors can also reveal misconfigurations inside the system that might mean that security controls are not protecting or cannot protect the system as designed.

Application Errors:

`Get-evenlog` cmd-let is told to fetch error type messages from application log. The application log contains information about application startup procedures and their behavior.

```
Get-Eventlog application | where { $_.EntryType -eq 'Er-
ror' } | select time,source,message
```

Time of the event, process name and error message are selected to be included in the output.

System Errors:

Get-evenlog cmd-let is used to fetch error type messages from the system log.

```
Get-Eventlog system | where { $_.EntryType -eq 'Error' }
| select time,source,message
```

Time of the event, process name and error message are selected to be included in the output.

8.5. Windows Boot Process

When Windows machine starts, Windows Boot Manager is first loaded from boot sector. Information is then loaded from Boot Configuration Data Store and the actual loading of operating system starts. Actions that the system takes during Windows boot can be analyzed from logs if they exist. The easiest way to get this data is to execute *Get-WinEvent* from powershell. This data can be converted to CSV with ConvertTo-csv parameter. The output lists what listeners have started, the applications that have been allowed or disallowed to run, or executed scripts and their sources.

8.6. USB Device History

Windows stores information and history of connected USB devices in the registry under `HKLM:/SYSTEM/CurrentControlSet/Enum/USBSTOR`. `Get-Itemproperty` cmd-let is used to values from the registry path.

```
Get-ItemProperty -erroraction SilentlyContinue -path
HKLM:/SYSTEM/CurrentControlSet/Enum/USBSTOR/*/*  | select
FriendlyName
```

If information is not cleared from registry, all connected USB devices can be listed. `FriendlyName`, however is not the best filter since some USB sticks tend to use the same name. In that case more precise device ID can be fetched from registry instead.

8.7. User Accounts

The Win32_UserAccount class contains information about user accounts on a Windows system. Namespace value is root/CIMV2. The used example does not print all

values from every user, only the selected ones.

```
Get-wmiobject -class "Win32_UserAccount" -namespace
"root\CIMV2" -erroraction SilentlyContinue | select Cap-
tion,AccountType, Descrip-
tion,Domain,FullName,PasswordChangeable,PasswordExpires,P
asswordRequired,Status
```

If command is run in Windows domain controller, all known AD account information is reported. Therefore the results list might be long and due to the contents it should be handled carefully. Also, if there is AD federation applied the user list from other domains is also reported. Extra user accounts should naturally be investigated.

8.8. Windows Driver Information

Windows Driver Files

By default Windows installs driver files to *c:\windows\system32\drivers.* Drivers are not always hardware device drivers, but can also be software of virtual device drivers. In that case the path can be something else and a typical example of this is anti-virus software, which usually loads its drivers from its own installation path.

List of Drivers

DOS command *driverquery* is used for getting the driver information that in the example includes name of the driver, if it is loaded and the file path.

```
$scriptdrivers = { driverquery /v }
$DriverQuery = Invoke-Command -scriptblock $scriptdrivers
-erroraction SilentlyContinue | Select-object 'Display
Name','Start Mode','Path'
```

Antivirus products and virtualization drivers often load their drivers from their own paths and make an exception. An example sign of intrusion is that a driver is placed to '<drive letter>:\System Volume Information' folder. This hidden folder is used by Windows for such operations as providing System Restore functionality, and the permissions on the folder, by default, grant only the SYSTEM account access to the folder.

Signing of Driver Files

64-bit Windows default policy requires that drivers are signed (Driver signing policy, msdn.microsoft.com). However, this is not the case in 32-bit Windows or if the policy has been modified to allow such behavior. Again dos command *driverquery* is used to fetch signing information.

```
$scriptdrivers2 = { driverquery -si }
$DriverQuery2 = Invoke-Command -scriptblock $scriptdriv-
ers2  -erroraction SilentlyContinue | Select-object 'De-
viceName','InfName','IsSigned','Manufacturer'
```

The name of the device, actual driver filename, signing status and manufacturer information are selected to the output in the example.

Driver File Dates

In the example driver information is queried with driverquery command and extra characters are stripped out for convenience. The output is then written to driver-list.txt file. Then *get-content* cmd-let is executed on every line of the driverslist.txt (driver file location).

```
Function DriverDates {
$DList = Invoke-Command -scriptblock $scriptdrivers -
erroraction SilentlyContinue  |% {$_ -replace
"\\\?\?\\"}| ConvertFrom-Csv | Select-object 'Path'|out-
file "$filepath\driverlist.txt"
$DriverPaths = get-content "$filepath\driverlist.txt"
foreach ($line in $DriverPaths) { Get-ChildItem $line -
erroraction SilentlyContinue | Select-Object -Property
FullName,Lastaccesstime,LastWriteTime}
                                }
```

`Get-content` cml-let can also be used to get information about other specific files.

8.9. Specific Anomalies from Windows Registry

The Windows operating system implements ”Image file execution” feature inside the user-mode part of CreateProcess. Image file execution takes place when a user clicks the icons that execute commands. After the operating system has determined what executable is going to be started it checks the registry key for a debugger. If it finds one, it will launch the debugger instead of the application. (Gregg Miskelly,2005, msdn.com)

By assigning debug option like cmd.exe to certain image file execution options, it is possible to bypass user authentication and system is accessed as with SYSTEM privileges.

```
Get-ChildItem
HKLM:HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\"Windows
NT"\CurrentVersion\"Image File Execution Options"\ -
recurse | foreach {
Get-Itemproperty -Path $_.PsPath } | Select-Object
PSPath,Debugger
```

Some of the possible file targets are: Sethc.exe, Magnify.exe, Login.scr, Utilman.exe, but also almost any executable such as notepad.exe can be defined. All positive results should be investigated.

Internet Explorer Settings

Internet explorer can be set to bypass SSL certificate checking and bypass overall caching of the data. If these changes are applied, there might not then be signs of malware communication in cache or in index.dat file, which keeps Internet explorer's browsing history. The setting can be made at user or system level.

User Level

```
Get-ItemProperty
HKCU:HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\Current
Version\"Internet Settings"\
```

System Level

```
Get-ItemProperty
HKLM:HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Curren
tVersion\"Internet Settings"\
```

Bypass settings can also be chosen to be there, thus the results should be verified with administrators.

AlwaysInstallElevated

Microsoft Windows operating systems come installed with a Windows Installer engine, which is used by MSI packages for installation. Setting specific registry keys orders Windows Installer to use system permissions when it installs the application on the system. (Margosis, 2011) In a normal environment applications not distributed by the administrator are installed using the user's privileges and only managed applications get elevated privileges. For a package to use elevated privileges the registry name "AlwaysInstallElevated" must exist in both keys with a *dword* value of 1.

```
Get-ItemProperty -erroraction SilentlyContinue
HKLM:HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windo
ws\Installer\
```

```
Get-ItemProperty -erroraction SilentlyContinue
HKCU:HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Window
s\Installer\
```

If the results indicate that both keys have `"AlwaysInstallElevated"=dword:00000001` value set, the inspection has to be made. A normal user

account can use this feature if both results return value 1. An intruder could use only user level accounts and still control the machine with specifically build MSI files.

Hidden User Accounts

Windows operating system has a feature called “Special account”. Local user names can be defined in the registry under SpecialAccounts key. User is then hidden from login screen’s user selection and control panel user menu.

```
Get-ChildItem -erroraction SilentlyContinue
HKLM:\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\"Windows
NT"\CurrentVersion\WinLogon\SpecialAccounts\ -recurse |
Select-Object Name,PSChildName,Property,ValueCount
```

Any results returned should be investigated. Some organizations use this feature to hide service accounts.

Cloned Administrator Accounts

Administrator accounts can be cloned in a way that a user reported with user rights only operates actually with administrative rights but does not appear in the administrator’s group. Registry values reside in the part of the Security Accounts Manager file, SAM that is not normally accessible to the local administrator. However, this part of the SAM is accessible to software running as SYSTEM account. The concept is to take a low-privileged user account in the SAM and to populate its F value with the data from the built-in Administrator account’s F value. (Joel Scambray, Stuart Mcclure, 2008)

Registry values can be inspected from following registry path
`HKEY_LOCAL_MACHINE/SAM/SAM/Domains/Users`

The automatic detection of this can be difficult since registry editor has to be run as a system. After gaining system level access, the method exports the registry keys for each user and manually compares the F and V values for each user to those of the built-in Administrator account.

8.10. Replacement or Manipulation of System Files

Certain system files can be replaced to gain system level access or to bypass a login prompt. Such example files are Sethc.exe, Magnify.exe, Login.scr and Utilman.exe. For example if `Magnifier.exe` is replaced with `cmd.exe`, clicking magnifier in the login screen runs the command prompt with system rights instead of actual magnifier. This can be used as a backdoor to the system when graphical desktop is used. Another concern are trojanized system binaries. With this technique, the malware patches bytes of a system binary to force the system to execute the malware the next time the infected binary is run or loaded. Typical targets are system binaries used frequently in normal Windows operation like DLLs. (Sikorski and Honig, 2012). The system files can be modified by malware or manually and gain different results.

Msv1_0.dll

File msv1_0.dll is part of the Windows's authentication package and this library file is responsible for verifying user's authentication. Path to file is Windows/system32/The.file.msv1_0.dll and it can be modified with a hex editor. By making specific changes to this file, which depend on the installed operating system architecture, intruder can log on to Windows with any existing account on the Windows without a password. However, if Microsofts's Encrypting File System is used, user's encrypted files cannot be opened. The modification requires taking ownership of the file and copying it outside the system for editing. *(Varsalone,McFadden, 2012)*

Detection of manipulated *msv1_0.dll* file or other system files can be achieved with System File Checker utility. System File Checker is a Windows built-in utility that allows users to scan for corrupted system files in Windows but only checks Windows own Dynamic Link Library files and the third party files remain unchecked. Utility also allows restoration of corrupted files from a cached copy. However, to prevent unwanted system changes from occurring */verifyonly* parameter is used in the command. This way detected corruptions or files changes are only logged to *%WINDIR%/Logs/CBS/CBS.log* file.

```
Invoke-expression –command “sfc /verifyonly”
```

Scanning usually takes from 3 to 15 minutes, depending on the system resources.

Search of Specific Files

If there are files that are considered dangerous or there is a specific list of files that has to be searched due to some feature of recent malware. The files can be searched with following command.

```
Get-ChildItem -recurse -path c:\ -Include psexec.exe
```

Example file *Psexec.exe is* utility that enables executions of programs on remote systems without preinstalling an agent. The recent versions encrypt all communication between local and remote systems, including the transmission of command information such as the user name and password under which the remote program executes. (Windows sysinternals, 2014). Psexec has actually been used in controlling machines without detection, however, it is just presented here as an example.

8.11. Windows Services

From anomaly perspective services are in key position. An intruder can gain persistence in the system by adding or modifying services. Windows allows tasks to run without their own processes or threads by using services that run as background applications; the code is scheduled and run by the Windows service manager without user input. At any given time on a Windows operating system, several services are running. Service running as SYSTEM in not necessarily a bad setting, since administrative privileges are needed to install such service. However, for malware writers this is convenient, because SYSTEM account has more access than administrator. (Sikorski, Honig, 2012)

When checking processes as simple information as a missing description can be a sign of bad process. Also, start-up path of the process, actual executable behind the service and loaded dynamic link libraries (DLL) provide useful information.

Running Services, Stopped Services and Their Start-up Paths

Windows services are either running or at stopped state. Services can be set to start automatically or manually by user. *Get-WmiObject* cmd-let is used to get service information from the system. In the example script results are sorted in a way that running services are printed first.

```
Get-WmiObject win32_service  -erroraction SilentlyContin-
ue | Select
Name,State,DisplayName,Pathname,ServiceType,StartMode,__P
ATH,Description,Startname | sort-object State
```

Loaded DLL Files Behind Services

The service usually has an executable that is running. This executable might be dependant on a dynamic link library files within the system and they must be loaded to memory as well. If it is known that specific DLLs are used for DLL injection, a list of executables and their loaded library files can be checked.

```
$scripttasklist1 = { tasklist /M }
$Servicedll = Invoke-Command -erroraction SilentlyContin-
ue -scriptblock $scripttasklist1  | Select-object 'Image
Name','PID','Modules'
```

As a result, many DLLs are listed and the list itself can be useless unless there is some indication of intrusion or malware activity. This is due to the fact that knowing all library files without a comparable list is probably impossible.

Service Credentials

Services run under given credentials and these credentials have certain user rights. Services should not run with too wide rights and therefore services with too high privileges (like SYSTEM) should be inspected. Service names, process id numbers and used user accounts are printed with DOS command `tasklist /V`.

```
$scripttasklist2 = { tasklist /V }
Invoke-Command -erroraction SilentlyContinue -scriptblock
$scripttasklist2 | Select-object 'Image Name','PID','User
Name'
```

Note: *tasklist* system executable can be trojanized and therefore the results might not be accurate. They can be compared with the results of port scanning and the information that the registry offers for services.

Services That Are Not Launched From Typical Locations

Windows protects folders like ”Program files” with User Account Control and access rights. An intruder or malware might install service to some other location like user folder or TEMP directory to go around these restrictions. It is possible that some services like SQL server has been installed to a different partition, which makes false positive.

```
Function StrangeServicePaths {
Get-WmiObject Win32_Service -erroraction SilentlyContinue
|
Where { $_.PathName -NotMatch '^\"*C\:\\Windows\\.+' } |
Where { $_.PathName -NotMatch '^\"*C\:\\Program
Files\\.+' } |
Where { $_.PathName -NotMatch '^\"*C\:\\Program Files
\(x86\)\\.+' } |
Select-object
@{Name='DisplayName';Expression={$_.DisplayName -join ';
'}},
@{Name='PathName';Expression={$_.PathName -join '; '}}
```

Another interesting service path could be *`System Volume Information`* folder, which is a hidden system folder that the *System Restore Tool* uses to store its information and restore points. Usually there is a System Volume Information folder on every partition of the computer. Since the folder is hidden by default it can be used to hide malware executables.

8.12. Network Connections

Services can be so called network services where the process is listening to the network and waiting for client connection. A process can also be a client that connects to another system over the network. Currently open network connections can be listed by executing *netstat* command.

```
$scriptnetstat = { netstat -an }
Invoke-Command -scriptblock $scriptnetstat -erroraction
SilentlyContinue | select-string LISTEN-
ING,ESTABLISHED,"TIME-WAIT" | select Line,Pattern
```

Strange port and Internet Protocol addresses should be investigated. However, results are not accurate because rootkit or malware can hide itself from netstat command. Processes and ports can be compared to service details, the network scans the results and the results of passive analysis.

8.13. Startup Programs

Following registry keys usually contain programs or program paths that are automatically run during system startup without requiring user interaction. Malware could use this feature to make it start at every startup and gain persistence.

```
HKLM\ SOFTWARE \Microsoft\Windows\CurrentVersion\Run

HKLM\ SOFTWARE \Microsoft\Windows\CurrentVersion\RunOnce

HKLM\ SOFTWARE
\Microsoft\Windows\CurrentVersion\RunOnceEx

HKLM\ SOFTWARE
\Microsoft\Windows\CurrentVersion\RunServices

HKLM\ SOFTWARE
\Microsoft\Windows\CurrentVersion\RunServicesOnce
```

Items under Run or RunServices keys are executed everytime a user logs in. RunOnce and RunServicesOnce are executed only once when the user logs in and then registry key is deleted. When item under RunOnceEx registry key is executed, a dialog box is displayed while the items contained in the registry key are being processed.

In Windows Vista, Windows Server 2008, Windows 7 and Windows Server 2008 R2 the RunOnceEx registry key does not exist by default.(support.microsoft.com)

8.14. Installed Software

Depending on the processor and operating system architecture, and on audit perspective, there are basically three software types installed on the Windows system: 32-bit programs, 64-bit programs and MSI packages. 32 and 64-bit programs are typically installed to different directories in the hard drive and information about them is stored in different locations of Windows registry.

MSI is an installer package file format used by Windows. MSI files can be used for installation, storage, and removal of programs. The files are contained in a package, which is used with the program's client-side installer service, an .EXE file, to open and install the program. (MSI , techtarget). For MSI package installation Windows Installer package is needed on the computer.

List of Installed MSI Lackages

A list of installed MSI packages can be queried using powershell with Get-WmiObject cmd-let and an example command lists name of the software, version number, name of the package, date of installation and vendor information. MSI packages are com-

mon and therefore the system administrator probably knows best what packages are normal and what are not.

```
Get-WmiObject win32_Product -erroraction SilentlyContinue
| Select Name,Version,PackageName,Installdate,Vendor |
Sort Installdate -Descending
```

List of Installed 32 bit Programs

Windows stores uninstall information of installed 32 bit programs in following registry path:
`HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`.
Powershell's `Get-ChildItem` cmd-let can be used to get values and information under certain registry path.

```
Get-ChildItem
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
-Recurse -erroraction SilentlyContinue | Select-object
PSChildName,PSPath
```

List of Installed 64 bit Programs

Windows stores uninstall information of installed 64-bit programs in the following registry path:
`HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall.`
Since the registry path is different from 32-bit programs extra query has to be made.

```
Get-ChildItem
HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersi
on\Uninstall -Recurse -erroraction SilentlyContinue | Se-
lect-object PSChildName,PSPath
```

If the system has a long history, programs propably have been installed, upgraded and uninstalled. Data pertaining to programs that are (or were at one time) installed on a system can also be found in the following registry locations:
`SOFTWARE\Microsoft\Windows\CurrentVersion\AppPaths`

8.15. Dynamic Registry Searches

Windows registry stores a great deal of information that can be static or dynamic. Following examples show how to search IP addresses by using optional regular expressions from registry. Malware might store IP addresses in clear text in Windows registry. Windows also stores history about Web addresses that users have typed or opened.

IP and Web Addresses From Registry

Windows registry stores IP addresses and Web addresses set by software or operating system. Addresses can be statically set for later use or collected as connection history. In the example script the wanted values are defined with regular expression language. The problem with Internet Protocol addresses is that their form can be similar to software version numbers. Internet protocol v4 addresses consist of four decimal numbers, each ranging from 0 to 255, separated by dots. Therefore, the address 1.2.3.4 can also be software version number. The example script searches from HKEY_LOCAL_MACHINE registry hive which contains most of the windows configuration data.

```
Function RegistryIP-HKLM {
Get-ChildItem HKLM:\ -rec -erroraction SilentlyContinue |
foreach {
$CurrentKey = (Get-ItemProperty -Path $_.PsPath)
select-string "\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b" -
input $CurrentKey -AllMatches | foreach {($_.matches)|
get-unique |select-object -unique Value}
                              }
                                                        }
```

By using the following regular expression filter instead web addresses can be searched and listed similarly.

```
select-string "\b(ht|f)tp(s?)[^ ]*\.[^ ]*(\/[^ ]*)*\b"
```

These searches can also be run against HKEY_CURRENT_USER, HKEY_CURRENT_CONFIG and HKEY_CLASSES_ROOT registry hives.

Searching connection history can be against the privacy law without a user's permission since Windows also stores the user's browsing history.

9. Linux Anomalies

Linux

Concept of Linux is that everything is done with files. Configurations are stored in files, commands and processes are executed with files. Instead of having files with specific extensions, executables are defined with special permissions given to the file. Due to the nature of security implementation of files and filesystem, Linux auditing requires more manual processing than Windows. Plenty of file permissions have to be verified and configurations interpreted. An intruder can take advantage of file security and manipulate Linux filesystem in many ways.

The difficulty in Linux security are rootkits that are hard to detect. For example, Linux rootkit LRK replaces valid common command binaries like *chfn, chsh, crontab, du, find, ifconfig, inetd, killall, login, ls, netstat, passwd, pidof, ps, rshd, syslogd, tcpd, top, sshd,* and *su* with trojanized versions. These versions give false information if they are executed and therefore, if possible, it would be good to get the same information from two different sources.

9.1. File Permissions in Linux

Stat is a Linux command that gives detailed information from wanted files.
The following is an example of an output of command $*stat /root/foo.txt.*
The explanations of the results are presented in Table 7.

```
"File: `/root/foo.txt'
Size: 1927      Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d    Inode: 1050715     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2014-02-12 21:02:26.897153312 +0200
Modify: 2014-02-12 21:02:25.409180584 +0200
Change: 2012-02-12 21:02:25.417180437 +0200
Birth: -"
```

Table 7 Explanations of Stat command results. (Unix Stat Command)

Fieldname	***Value***	***Explanation***
File	`/root/foo.txt'	Absolute path name of the file
Size	*1927*	File size in bytes
Blocks	*8*	Total number of blocks used by this file
IO Block	*4096*	IO block size for this file
Regular file	Regular file	Indicates the file type. Available file types: regular file. (ex: all normal files), directory. (ex: directories), socket. (ex: sockets), sym-bolic link. (ex: symbolic links.), block special file (ex: hard disk), character special file. (ex: terminal device file).
Device	801h/2049d	Device number in hex and device number in decimal

Inode	1050715	Inode number is a unique number for each file which is used for the internal maintenance by the file system
Links	*1*	Number of links to the file
Access	(0644/-rw-r—r--)	Access specifier displayed in both octal and character format.
Uid	(0/ root)	File owner's user id and user name are displayed
Gid	(0/ root)	File owner's group id and group name are displayed
Access	2014-02-12 21:02:26.897153312 +0200	Last access time of the file
Modify	2014-02-12 21:02:25.409180584 +0200	Last modification time of the file.
Change	2012-03-12 21:02:25.417180437 +0200	Last change time of the inode data of that file.

File Permissions in Octal Format

This information about the file is displayed in the Access field of the previous example command. Table 8 presents the values for read, write and execute permission in Unix.

Table 8 Meaning of octal values

Value	Meaning
4	Read permission
2	Write permission
1	Execute permission

Numbers can be added up to get other kind of permissions. Table 9 presents meaning of added up values.

Table 9 Meaning of added up octal values.

Value	Sum	Meaning
7	4+2+1	Read/Write/Execute
6	4+2	Read/Write
5	4+1	Read/Execute
4	4	Read
3	2+1	Write/Execute
2	2	Write
1	1	Execute

File Permission in Character Format

This information about the file is displayed in the Access field of the example command

- **File Type:** First bit of the field mentions the type of the file.
- **User Permission:** 2nd, 3rd and 4th character specifies the read, write and execute permission of the user.
- **Group Permission:** 5th, 6th and 7th character specifies the read, write and execute permission of the group.

- **Others Permission:** 8th, 9th and 10th character specifies the read, write and execute permission of the others.

(thegeekstuff,2009)

The results of *ls* or *stat* command cannot be fully trusted. An intruder can modify Linux file dates to look like they were not modified lately. Information can therefore be misleading. For example, date modifications can be made with *touch –r* command.

```
Touch –r existing_filename_with_wanted_timestap target_file
```

The command replaces the modification timestamp of target files with source files modification timestamp. This looks normal when *ls* command is used; however, the change timestamp is updated to current date, which can be verified with stat command.

9.2. Anomalies From Linux Logs

Linux logging configuration is checked during audit as a normal procedure.
Often *last* command is used to run the report of previously logged on users and their usernames. Missing log files or changed log configuration file can be a sign of intrusion. By default in most distributions logs are archived every 4 weeks. To extract the archived files, disk space and proper commands have to be available. Some commands can parse archived file on-the-fly; however, they do not necessarily exist within the system. To print the contents of a file one could use *"cat file"* command. But if a file is archived and packed one could use *"zcat log.gz"* instead. *Zcat* and similar commands are not always available, and therefore the system being audited decides which is the better method.

Sudoers

In Linux or Unix operating systems sudo is a program that allows users to run programs with the security privileges of another user. Rights that *sudo* user can have are full rights (superuser), or limited rights to specific commands. Accounts that have used their sudo rights can be found in the auth.log file by searching lines containing word *"sudo"* from log file.

```
grep sudo /var/log/auth.log
```

Execution dates, usernames and actions can be listed.

USB Devices

In Linux attached USB device appears in the */var/log/dmesg* file.
For USB device history the file can be inspected by searching lines that contain the word USB.

```
grep USB /var/log/dmesg
```

Typically the date, product id, vendor id and friendly names are found; however, just like in Windows a friendly name is not always available, instead the result might be just a device id in numeric format.

9.3. Loaded Drivers and Modules

Linux kernel module is a compiled code that can be inserted into the kernel at runtime. Common commands to do so in live system are *insmod* or *modprobe*. A driver is a code that runs in the kernel to talk to some hardware device. A driver may

be built statically into the kernel file on disk. A driver may also be built as a kernel module so that it can be dynamically loaded later. Loaded drivers and modules can be verified in couple of ways. One is running *lsmod* command which return statuses of loaded kernel modules. With *modinfo* command one can check specific module or driver details. Another way is to check contents of */proc/modules* folder. There results can be compared with each other. One can also verify dates and paths of the loaded driver files just like in windows.

Driver Information Comparison

This method uses *lsmod* command to get full information loaded modules and outputs them to text file. Full output of results is valuable if further inspection is needed.

```
$lsmod | grep -v Module > drivers_from_lsmod_full.txt
```

To list of only the module names, *lsmod* command is used and only the module name field is selected and outputted to text file. This could also be achieved by parsing module names from previous text file that contains the full output.

```
$lsmod | grep -v Module| cut -d ' ' -f1 > drivers_from_lsmod.txt
```

This method gets the module information directly from the filesystem. *Cat* command is used to fetch the contents of the */proc/modules/* folder. Only the first field (module name) of output is selected.

```
$cat /proc/modules | cut -d ' ' -f1 > drivers_from_proc.txt
```

Diff command finds differences between the lines of two or more files. Here contents of previous commands are compared with each other. Normally there should not be any differences.

```
$diff drivers_from_lsmod.txt drivers_from_proc.txt
```

Driver Details

Getting driver information with one command requires two phases. First, module names are extracted from *lsmod* results.

```
Module (name)                  Size               Used by
nls_utf8                      12416               1
nls_cp437                     12417               1
vfat                          17157               1
…
```

The output lists installed kernel modules, their sizes and state of usage.

Next, results are used as input to *modinfo* command that gives details of the module.
Command $*modinfo nls_utf8* gives following details from module named *nls_utf8.*

```
filename:       /lib/modules/3.10-3-686-
pae/kernel/fs/nls/nls_utf8.ko
license:        Dual BSD/GPL
depends:
intree:         Y
vermagic:       3.10-3-686-pae SMP mod_unload modversions
686
```

Information contains location of the module file, license information, dependencies on other modules or libraries and kernel information.

The full command is presented as follows and the output is directed to text file.

```
modinfo `lsmod | grep -v Module| cut -d ' ' -f1` > driver_details.txt
```

The examination requires a manual review of the output files. A simple missing description of the driver can be a sign of anomaly.

Driver Paths

The previous *modinfo* output already contains file paths and they can be collected with the following command. The purpose is to use results in the next command that gets the driver file details such as file dates.

```
cat driver_details.txt| grep file | cut -d ' ' -f8 > driver_paths.txt
```

Driver and Module Filedates

Driver files are fed to *stat* command, which gives detailed information of the files. Function runs *stat* command to every file listed in the previous output file.

```
function driver_file_dates {
while read line
do
    stat $line
done < driver_paths.txt
}
driver_file_dates > driver_file_dates.txt
```

9.4. Change and Creation Dates of Specific Files

Important or otherwise interesting file dates and rights can be verified with *stat* command. The command lists access rights, last access date, last modification date, and change date. At least the following files should be checked since they control user accounts and their rights:

```
Stat /etc/passwd
Stat /etc/shadow
Stat /etc/group
Stat /etc/sudoers
```

Logging settings and location of the configuration files depend on used log daemon: *syslog, syslog-ng, rsyslog* (remote logging). These files are usually keys to disruption of the logging.

```
Stat /etc/rsyslog.conf
Stat /etc/syslog.conf
```

The installed and used service configurations can be checked accordingly.

9.5. Hidden or Abnormal Files

In Linux, the files can be hidden from a user by adding dot (.) in front of the file. Normal file listing command *ls* does not show them unless parameter *–a* is given. Some of the hiding techniques rely on humans' way to interpret visual data. If *ls –la* command is executed, Linux and Unix list two directories in the beginning of the output:

```
drwxr-xr-x  2 root root   4096 Mar 12 23:06 .
drwxr-xr-x 29 root root 344064 Mar 12 22:41 ..
```

Single dot means current directory. Two dots refer to the previous directory in the directory tree.

The find command could be limited with *–xdev* parameter. That way the command does not escape to external shares. To find all hidden files or directories (any file-name that starts with a dot) the following command can be used. Typically there are plenty results if the whole filesystem is searched.

```
$find / -name ".*"
```

The user can be visually distracted to miss file or directory names due to the file listing method where the output already has "." and ".." file names.

An example is as follows: three dots for a file name.

```
$find / -name "..." -print 2> /dev/null
```

Example : dot and space for a file name is given below.

```
$find / -name ". " -print 2> /dev/null
```

9.6. File Discovery

Files can also be hidden to directories where people do not usually look. Directory might not be expected to have any normal files and therefore is not normally routinely searched. One directory example is */dev*. To find regular non-device files hidden in */dev*:

```
find /dev –type f –print 2> /dev/null
```

However, there might be acceptable files like */dev/MAKEDEV.*

Setuid and Setgid Shells Owned by Root

Setuid and *setgid* programs run with the privileges of the file owner or a group. Intruder who has gained root privileges may place a copy of */bin/bash* in a different location and use the *chmod* command with the *u+g+s+x* parameter to set the *setuid* flag on it. After this is done, the intruder may log in again with a non-root account and simply run the setuid's bash executable to obtain a root shell and root password is not needed.

Command that finds *setuid* files.

```
$find / –perm –4000 –type f –print 2> /dev/null
```

Command that finds *setgid* files.

```
$find / –perm –2000 –type f –print 2> /dev/null
```

Both commands by default return files that have *setuid* or *setgid* flag on. Instant unauthenticated root level access is offered by copied shell binaries like sh,

bash etc. However, the file names of these shells can also be renamed to something else.

Orphaned Files

If the user is removed from the system, there might be files that old User ID still owns. These files are not necessarily automatically deleted when the account itself is removed and might contain data that should not be available to everyone. To find the paths of orphaned files, *find* command with the *–nouser* argument can be used.

```
sudo find / -xdev -nouser
```

9.7. Cron

Cron or crontab is a Linux equivalent to scheduled tasks in Windows. Scheduled tasks can also be used by users if allowed. Therefore every user can have their own scheduling configuration and a check has to be made per user basis. The check can be made by extracting usernames from /etc/passwd file and checking if crontab has a configuration for that username. Root accounts crontab has to be checked separately if command is run under root account.

Crontab command with *–l* parameter lists current users (account command is being run as) scheduled tasks. The output is directed to text file.

```
crontab -l >> user_crontabs.txt
```

Crontab command with *–u* parameter shows given usernames scheduled tasks. Usernames are first fetched from */etc/passwd* file and crontab is then queried for every configured username.

```
function cron {
for user in $(cat /etc/passwd| cut –f1 –d:);
do crontab –u $user –l|grep –v '^#';
done
}
```

Under Ubuntu or Debian, it is possible to view crontab by opening folder */var/spool/cron/crontabs/* and then a file for each user is there. But that is only for user-specific *crontab* and system *cron* has to be checked separately. RedHat has similar */var/spool/cron* directory. Checking the cron via these files is easier if user accounts are centrally managed.

Cat command can be used to list contents of the files.

```
cat /var/spool/cron/crontabs/*
cat /var/spool/cron/*
```

9.8. Rhosts and Hosts.equiv

.rhosts file (hidden file) defines which remote hosts can invoke certain commands on the local host without supplying a password. This can be used to access the operating system later on.

```
$find / –name .rhosts
```

All results are bad, even from normal security perspective.

/etc/hosts.equiv that provides the same functionality as .rhosts
The `/etc/hosts.equiv` file contains a list of hostnames and users that are considered trusted. Services such as rlogin, rsh, rcp, and rcpmd use this file to determine trusted entities.

9.9. PATH Variables

When a user executes a command, the command shell searches for the location of the command in the list of directories contained in the *PATH* environment variable. If an attacker plants a trojanized binary to users home folder and ads "." to the environment's PATH variable , the command is executed from there instead of the normal path. For the operating system "." stands for current directory. It is also good to check all *PATH* variables from all users since binaries can be put to other directories as well.

File */etc/profile* contains Linux system wide environment and startup programs. It is used by all users with bash, ksh, sh shell. It is usually used to set PATH variable, user limits, and other settings for user. It only runs for login shell. Large changes or application specific changes are configured under */etc/profile.d/* directory.

Other options for variables are *.bash_profile* file in user's home directory, which is run during user login. Another option is *.bashrc* which is run when the user opens new terminal windows after login. With sh shell similar files are *.login* and *.cshrc.*

9.10. Linux Processes

Linux processes and their paths can be checked in ways that can be compared. One way is running *ps –ef* command, which prints process id number (PID) and process paths. PID numbers can also be found in /proc directory, where every PID number is its own directory. The amount and existence of PID numbers can be compared within these two outputs.

If Linux system is rootkitted, processes can be hidden from commands that list processes or commands refuse to print certain processes. A typical trojanized command is actually *ps*.

PID List

Process ID, PID list can be retrieved with *ps* command. PID is a numerical value that every running process get from the operating system or kernel. Every process has it's own PID number. Following command get PID numbers of running processes and writes them to a text file.

```
ps –ef |awk '{print $2}'|grep –v PID |sort > pids_ps.txt
```

PIDS from /Proc

Directory */proc* is not only a directory, but also a virtual filesystem. Kernel uses it as a kind of control and information centre and therefore process information is also stored there.

```
ls –l /proc |awk '{print $9}' |grep [1-999999] |sort >
pid s_proc.txt
```

If these two lists are compared with each other, there are always few differences. One reason is that *ps –ef* command itself appears in the process output.

Startup Paths

The startup path of processes can be collected by executing *ps –ef* command and printing only the wanted column that contains the path variable.

```
ps -ef | awk '{print $8}'
```

Paths can be also examined from */proc/PIDNUMBER* directory which usually has a link file called exe that is linked to the actual file execution path.

False Services

An intruder may add an entry to the *inetd.conf* file or the *xinetd.d* directory to cause *inetd* or *xinetd* to serve a remote shell on a specific port. This can be done by configuring *inetd* or *xinetd* to execute */bin/bash* or any other command shell. Once this is accomplished, an attacker may telnet to the remote port to gain a remote shell. For example, an attacker may place the following contents into the file */etc/xinetd.d/* domain:

```
service domain
{
        socket_type     = stream
        wait            = no
        user            = root
        server          = /bin/bash
        server_args     = -i
}
```

After the service is restarted, the attacker telnets to port 53 and gets a root shell. However, this does not work in all modern systems. The contents of the service files should be inspected.

10. Passive Analysis

Purpose of Analysis

The purpose of network analysis is to verify that the target being audited works as intended and there are no design flaws. From anomaly perspective it is interesting to discover communication partners and what is being communicated. Machines could be a part of a botnet, or some malware is exfiltrating data out of the network.

Data exfiltration means unauthorized transfer of data from a computer. Exfiltration can be done with a variety of protocols and typically a usable protocol depends on what kind of Internet connectivity is possible. If Internet browsing is possible, data exfiltration is easier and hacker or malware have more transfer options to choose from. The transfer could be done for example with different hypertext transfer protocol (http) methods (for example posts), domain name service (DNS) queries or email. Data exfiltration can be carried out in a way that normal protocol behavior is retained.

Communication partners can be inside the target's network, in partners' network or in the Internet. IP addresses and DNS names that come from internal network are easier to track and analyze than outside ones.

> *Sadly, cybercriminals use increasingly sophisticated methods of communication such as Domain Generation Algorithms (DGA) designed to evade detection in the growing noise of web traffic and to prevent the takedown of a botnet. DGAs are algorithms used by malware that generate domain names, which then serve as rendezvous points with their controllers. They are used as a method to restore communication when a controller is offline. (Barry Weymes,2013)*

Communication is not necessarily taking place during audit. Therefore historical data like proxy, DNS or firewall logs that could be parsed, could offer valuable information. However, the format of the log files cannot be predicted which means that the parsing method has to be developed on-site. Log analysis might also be against privacy law.

10.1. Traffic Capture

Network traffic mirroring is carried out during the audit so that all traffic that comes from or goes to audited target is captured. Typically whole packets (with snaplength 0) are captured and saved for further investigation. The capturing time is usually 12-24 hours and is usually limited by disk space. Found IP addresses are mapped and IP conversation partners are semi-automatically drawn to a map. Semi-automatically means that for example a product, Clarified analyzer, lists IP addresses; however, those addresses have to be manually given an identity in a graphical map. The analyzer then automatically draws connecting lines between IP addresses that were carrying out conversation or tried to do so during capture period. If there is specific data that needs to be analyzed, it has to be done manually with Wireshark. If there is plenty of traffic and connections, analysis is time consuming.

Following are examples how this process can be speeded up and simultaneously important data from anomaly perspective is collected. In the examples *captured.pcap* is the file being analyzed. Live capture could be carried out by replacing *"-r capture.pcap"* with *"-i Interfacename".* The same things in the examples given can be done with Wireshark; however, when processing multiple capture files automatically with less resources, Tshark is better.

10.2. Detected IP Addresses

A list of detected IP addresses can be enlisted with the following command. The command is piped to *uniq* program so that every IP address is listed only once. List can be compared to list of unwanted or dangerous IP list. If tunneling methods like Generic routing encapsulation (GRE) is detected, the line in file contains two or more IP addresses separated with a comma: Tunnel point addresses and tunneled addresses.

```
tshark -r capture.pcap -T fields -e ip.dst ip.src |sort
|uniq
```

10.3. Protocols

Following command lists detected protocols from pcap file. Protocol detection depends on software version and is not necessarily accurate if services are using non-standard ports.

```
tshark -p -r capture.pcap -qz io,phs
```

If there are strange protocols, details can be investigated with Wireshark or with *tshark :* `tshark –r capture.pcap –R protocolname`

If the protocol and expected port do not match traffic can be decoded again.

Example:
`tshark –r capture.pcap -d tcp.port==8888-8890,http`
will decode any traffic running over TCP ports 8888, 8889 or 8890 as http.

10.4. Conversation Partners

List of IP addresses that communicate with each other with transmission control protocol (TCP) or User Datagram Protocol, UDP can be listed with following commands. The command makes a list of address pairs that were communicating during capture and both protocols have to be reported with different commands.

TCP streams

```
tshark –n –p –r capture.pcap –qz conv,tcp
```

UDP streams

```
tshark –n –p –r capture.pcap –qz conv,udp
```

10.5. Connected Devices

Every Ethernet based device has a MAC address that is tied to network card or port. A list of seen unique MAC addresses can be obtained with following command.

```
tshark –p –r capture.pcap –T fields –e eth.dst eth.src
|sort |uniq
```

The list of addresses can be compared to the vendor list and unwanted devices might be detected. The sample list can be downloaded at http://anonsvn.wireshark.org/wireshark/trunk/manuf .

10.6. Keyword Search

Packets can be parsed by searching certain keywords (like in this example hook.js). Additionally, hex values can be searched. The output is then written to a new file that can be analyzed separately.

```
tshark –r capture.pcap tcp contains hook.js –w key-
word.pcap
```

With –w option all packets that contain the wanted keyword are written to key-word.pcap file for further analysis. If there is plenty of data to analyze, this seems to be practical.

10.7. Http Traffic

Http and https are the most common protocols being used for Internet browsing. What differentiates http and https from other protocols, is their amount of parameter and usage options. And from the data infiltration perspective, the options does not have to be used in a designed way. For example both *POST* and *GET* methods can be used to send or receive data.

Http Hosts

A list of unique http domains can be listed with following command. List can be then compared to list of dangerous domain or host lists.

```
tshark –p –r capture.pcap –T fields –e http.host |sort |
uniq
```

Http Posts

Http posts can reveal activity that is using http posts to send data out. A great amount of posts to same destination, or to dangerous destination should be investigated. Also, posts that are sending files (Documents, pictures etc.) might indicate ongoing data exfiltration.

```
tshark -p -r capture.pcap -n -R
http.request.method=="POST" |sort
```

Http Requests

Http requests and optionally, their received response can be listed as follows. Data can be used to analyze strange requests. However, this might reveal private data.

```
tshark -r capture.pcap -V -R "http.request ||
http.response"
```

User-Agents

A user-agent is software (a software agent) that is acting on behalf of a user.
Web browsers, email clients and system components have often their own agent names.

Format of a typical user-agent *(sans reading room,2013)*

"User-Agent" ":" 1(product | comment)*

Example user-agent *(sans reading room,2013)*

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

Detected user-agents can be listed with the following command and strange user-agent names should be investigated. User-agent data is first sorted out alphabetically and then every user-agent is reported only once.

```
tshark -r capture.pcap -T fields -e http.user_agent |sort
|uniq
```

Http Requests With Empty User-agent and IP Address

Software or malware might not report their user-agents while communicating. However some antivirus vendors do this on purpose so false positives are possible.

```
tshark -r capture.pcap -R "http.request==1 and
http.user_agent" -T fields -e http.user_agent -e ip.addr
|sort |uniq
```

Previous missing user-agent reporting can be combined with address the user-agent is communicating to.

```
tshark -r capture.pcap -R "http.request.method==”POST”
and http.user_agent" -T fields -e http.user_agent -e
ip.addr |sort |uniq
```

Strange user-agents or requests that are missing user-agent parameter should be investigated.

10.8. DNS Queries

It is important to list DNS names that are requested even if there is no Internet connection. For example, malware might be trying to contact some IP and is first trying to resolve DNS name first. Another aspect is data infiltration where DNS protocol can be used even if there is no direct Internet access (only DNS resolving is possible).

Requested DNS domain names and responses can be listed with following command.

```
tshark –r capture.pcap -2 –R dns.resp.addr –T fields –e
dns.qry.name –e dns.resp.addr
```

Many requests to the same domain can indicate malicious activity. Calculating the amount of queries to the same domain eases the task of detection.

This list can also be compared to the list of dangerous domain and host information.

10.9. Encryption

Command and Control traffic can be encrypted to make it harder to detect and to look valid. Negotiated encryption methods are extra information, however, certificate information can be used to track the usage of stolen or dangerous certificates.

Following command dumps all IP addresses that used SSL or TLS encryption and show their encryption negotiation data.

```
ssldump –N –r capture.pcap >> ssl_negotiations_raw.txt
```

The list of detected cipher suites can be parsed from previously collected data. Previous ssl_negotitations_raw list is large so it is easier to filter interesting data from it

first and then look for details from the large file. Again, every cipher suite is reported only once.

```
grep cipherSuite ssl_negotiations_raw.txt |sort |uniq >>
ssl_ciphersuites_raw.txt
```

Certificate Information From Captured Traffic

Handshake certificate information can be reported with the following command. This does not print the actual certificates but certificate names and paths. Old/revoked or strange certificates should be inspected.

```
tshark -r capture.pcap -R ssl.handshake.certificate -V >>
certificates_raw.txt
```

11. External Sources for Data Comparison

Data comparison to external sources, especially with IP addresses is important. However, lists are not accurate and addresses proven friendly today can be dangerous tomorrow. Therefore extra historical data (firewall, proxy, DNS etc.) are valuable, however, recent data should also be compared to external sources later. A home country's own IP addresses are not necessarily safe either.

> *In an attempt to better evade detection, cybercriminals are increasingly configuring their command and control infrastructure in such a way that initial malware callbacks communicate with a server located in the same country as the newly infected machines.*
> *(Malware C&C Servers Found in 184 Countries)*

Tor network is not only a network that offers privacy but is also used by criminals and employees. Tor networks' exit node addresses can be downloaded at:

`http://exitlist.torproject.org/exit-addresses`
`https://www.dan.me.uk/tornodes`

Few services offer almost real-time data about sites, domains and IP addresses that spread malware intentionally or unintentionally.

The complete malware domain list database in CSV format is as follows:

`http://www.malwaredomainlist.com/mdlcsv.php`

A list of active IP addresses is presented below as follows:

`http://www.malwaredomainlist.com/hostslist/IP.txt`
`http://shadowserver.org/wiki/`
`https://zeustracker.abuse.ch/blocklist.php`

https://spyeyetracker.abuse.ch/blocklist.php
https://palevotracker.abuse.ch/blocklists.php
http://www.spamhaus.org/drop/drop.txt

When comparing results to these sites, it is good to remember that data comes somewhat behind these services. Therefore, the comparison might be more valuable if it is done a few days later.

12. Results

Methods and tools of this thesis were tested during normal audits, and results varied from “everything is ok” to a start of incident management process. Usually findings are old user accounts, a variety of unknown USB devices, bad system update scheduling, undefined logging policies, misconfigurations that slow down systems and applications that phone home. Finding APT type malware requires luck if time for analysis is short, since there might not be any activity at all.

Even though Powershell is common in all modern Windows operating systems, differencies exist. For example, if one wants to check the creation time of registration keys of USB device history, the same method does not necessarily work with all server versions. This means that the tool should check its environment before execution or different versions of the tools should be made for different Windows versions. Log analysis also take 100% of the CPU which can slow down other operations on the server. Remote auditing (ps-remoting) is not recommended, since after testing it was noticed that it causes plenty of extra events and traffic to target systems. This might pollute the actual audit results. Also, examples in this thesis do not have export options defined. The actual Proof of Concept tool written actually converts everything to HTML and CSV format.

Linux systems are mostly used as servers, and if there are many files or shares, doing multiple file searches (added with ones that are done during normal audit) can be time consuming. However, it is the machine that does the work and the waiting time can be used to inspect something else. It is also good practice to run a log analysis before other systems commands that are used during the audit because they sometimes fill the log.

Passive analysis with Tshark is relatively easy. If there is only one or two files to be analyzed Wireshark does the same job but not automatically. It was also noticed that

capture size per file should be around 2 GB or smaller with 8GB memory. The analysis takes memory and one Tshark command uses one CPU core. The largest data amount that was analyzed was 900 pcap files of 500MB size. The computer analyzed this data for about 15 hours (during the night); however a great deal more data was extracted than the presented examples in this thesis (for example cookies).

Following is example from real company that was audited with the presented methods. This was also a case where the order of the actual steps taken during the audit reconsidered.

Running Full Virus Scan

Full virus scan to all computers (normally not executed during audit) crashed several computers and alarmed about trojans. The infected machines had 2-4 trojans (Java exploits). During an offline scan from external media the same machines reported 18-22 trojans.

Passive Analysis

Passive analysis indicated plenty of traffic to the Internet from infected machines. Few machines not infected according to antivirus were doing http posts to external servers. The posts were used to send files out of the company network without user actions. The posts occurred every now and then, and therefore the amount of traffic was low and did not generate any alarms about suspicious activity. The collected user-agents reported unwanted software.

Vulnerability- and Port Scans

The lesson learned was that running vulnerability or port scans before operating system audit has an effect on system audit results. Most of the scanners try a different kind of access and authentication methods to systems, which generates logs. If the logging policy is not defined before scans, some of the logs are possibly overwritten or lost.

Anomalies From Operating Systems

Getting user names from AD indicated that there was an old AD federation to the old doain. What was interesting was that old accounts were still usable under different domain name, however forgotten. With the help of log analysis it was possible to track if any of the accounts was used by someone and the data was already available. Computer login scripts were modified to add an admin user with specific password to a local computer at every boot. The login script was encoded VBS and therefore readable by default. The alwaysinstallelevated registry key was used to ease installations from admin accounts. Therefore there were no logs about this activity.

Log Analysis

The log analysis works; however, some results were polluted because network scanning and virus scanning were carried out before the operating system analysis. Also, the logging policy was too loose in a way that some of the events were overwritten due to file size limits.

If a Linux system is doing command recording, log files have to be analyzed first. Otherwise, this record is full of audit commands and the history can be lost.

13. Conclusions

It can take years until system intrusions are revealed and attackers might have remained undetected for a long time. From that point of view, should systems that have already been in production be inspected differently than in normal audit? It is possible to detect anomalies during audit, if there is enough knowledge of the methods that attackers use, data is collected in a way to be easily analyzed and compared to private or public sources of intelligence. In Finland Finnish privacy law and usually the time available to the audit limits chances to detect intrusions or other anomalies. Due the time limits and amount of data, great deal of automation has to be used but a skilled professional can usually point the difference between human made and machine made actions better. Due to the privacy law data cannot be collected or analyzed as thoroughly as might be required to detect threats like APT and therefore automation should be used. And to get results from automation, specific information like APT signatures are needed. However, a simple sign of anomaly like strange account or network activity can give a clue. Also, the order of executed audit functions matters. From audit process perspective operating systems and their logs should be audited first before active actions like vulnerability scanning or virus scanning are executed. It is possible that data such as logs are lost or overwritten, because system might not be compliant already or it is misconfigured. Before taking any technical actions, auditor should verify that there will not be any damage to existing log data.

Not all malware or forensics methods can be adopted to audit process, since they are a wide and dynamic area of research. However, basic actions can be taken without affecting audit costs or people's privacy. The methods in this thesis are just examples of ways to check certain kind of data and actually just samples of actual tests, and these samples are only for mainstream operating systems and networks. Every audited target could also be analyzed for intrusions whether it is a firewall, switch or any other special equipment. Most of the firewalls and big switches actually have Linux or BSD as a operating system and therefore the same security checks should apply instead of just checking configurations.

Knowledge to this process comes from news and hacking courses and from other information sources. It would be good that this knowledge would automatically be used to enhance audit process and tools. However, It's not always necessary to develop exact detection mechanism to certain attacks by hand. System's behavior can be inspected instead and automatic analyzers or tools are more suitable when one is trying to detect malware. It is also good the advantage on existing monitoring systems like IDS or antivirus systems and let hem do the work and just verify the results.

It is valuable to include anomaly searching to audit process, real life experience has proven that. However, privacy law is not quite clear in this area and permissions to do certain inspections have to be received and confirmed by an auditor.

References

Act on the Protection of Privacy in Electronic Communications. PDF document on Finlex website. Accessed on 13. January 2014. Retrieved from http://www.finlex.fi/en/laki/kaannokset/2004/en20040516.pdf

Andress, Winterfeld. 2014. Cyber warfare Techniques, Tactics and Tools for Security Practitioners. USA:Syngress.

Audit logon events. Page on Microsoft technet website. Accessed on 14 April 2014. Retrieved from http://technet.microsoft.com/en-us/library/cc787567%28v=ws.10%29.aspx

Audit logon events. Page on Microsoft technet website. Accessed on 27 December 2013. Retrieved from http://technet.microsoft.com/en-us/library/cc787567%28v=ws.10%29.aspx

Bejtlich R. 2013. The Practice of Network Security Monitoring. San Francisco:No Starch Press Inc.

Carvey H. 2007. Windows forensic analysis, USA:Syngress.

Carvey H. 2011. Windows registry forensics, USA:Syngress.

Constantin L. 2012. Java-based Web attack installs hard-to-detect malware in RAM. Article on infoworld.com. Accessed on 19 March 2014. Retrieved from http://www.infoworld.com/d/security/java-based-web-attack-installs-hard-detect-malware-in-ram-188960

Danchev D. 2013. Cybercriminals experiment with Tor-based C&C ring-3-rootkit empowered SPDY form grabbing malware bot. Blog on webroot.com. Accessed on 17 March 2014. Retrieved from http://www.webroot.com/blog/2013/07/02/cybercriminals-experiment-with-tor-based-cc-ring-3-rootkit-empowered-spdy-form-grabbing-malware-bot/

Davidoff, Ham. 2012. Network Forensics Tracking Hackers through Cyber-space. Prentice Hall

Definition: Security audit. Page on techtarget website. Accessed on 6 February 2014. Retrieved from http://searchcio.techtarget.com/definition/security-audit

Description of security events in Windows 7 and in Windows Server 2008 R2. Page on Microsoft website. Accessed on 16 May 2014. Retrieved from http://support.microsoft.com/kb/977519

Description of the RunOnceEx Registry Key. Page on Microsoft support website. Accessed on 17 March 2014. Retrieved from http://support.microsoft.com/kb/310593

Donohue b. 2013. Malware C&C Servers Found in 184 Countries, Blog on threathpost.com website. Accessed on 12 January 2014. Retrieved from http://threatpost.com/malware-cc-servers-found-in-184-countries/

Driver signing policy. Page on Microsoft msdn website. Accessed on 1 April 2014. Retrieved from http://msdn.microsoft.com/en-us/library/windows/hardware/ff548231%28v=vs.85%29.aspx

Erickson J. 2008. Hacking The Art of Exploitation, 2nd Edition, San Francisco:No Starch Press Inc.

Gallagher S. 2013. Your USB cable, the spy: Inside the NSA's catalog of surveillance magic. Article on arstechnica.com. Accessed on 2 January 2014. Retrieved from http://arstechnica.com/information-technology/2013/12/inside-the-nsas-leaked-catalog-of-surveillance-magic/

Garrison P. 2010. Digital Forensics for Network Internet and Cloud Computing, Chapter 2 Capturing Network Traffic, page 46. Burlington:Syngress.

Hale Ligh, Adair S, Hartstein B, Richard M. 2011. Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code, Indianapolis:Wiley Publishing Inc.

John E Dunn, 2012, Japan's Finance Ministry uncovers major Trojan attack, Article news.techworld.com, Accessed on 28 December 2013. Retrieved from http://news.techworld.com/security/3371587/japans-finance-ministry-uncovers-major-trojan-attack/

Kovacs E. 2013. Researchers Find Way to Detect Direct Memory Access Malware. Article on news.softpedia.com. Accessed on 28 December 2013. Retrieved from http://news.softpedia.com/news/Researchers-Find-Way-to-Detect-Direct-Memory-Access-Malware-386671.shtml

Mandiant® Releases Annual Threat Report on Advanced Targeted Attacks. Page on Mandiant Website. Accessed on 2 November 2013. Retrieved from https://www.mandiant.com/news/release/mandiant-releases-annual-threat-report-on-advanced-targeted-attacks1/

Margosis A. "AlwaysInstallElevated" is Equivalent to Granting Administrative Rights. Blog on technet.com website. Accessed on 28 December 2013. Retrieved from http://blogs.technet.com/b/fdcc/archive/2011/01/25/alwaysinstallelevated-is-equivalent-to-granting-administrative-rights.aspx

Miskelly G. 2005. Inside 'Image File Execution Options' debugging. Blog on msdn.com website. Accessed on 12 December 2013. Retrieved from http://blogs.msdn.com/b/greggm/archive/2005/02/21/377663.aspx

MSI file format. Page on Techtarget website. Accessed on 12 December 2013. Retrieved from http://whatis.techtarget.com/fileformat/MSI-Installer-package-Microsoft-Windows

Natarajan R. 2009. Unix Stat Command: How To Identify File Attributes. Blog on Thegeekstuff website. Accessed on 12 February 2014. Retrieved from http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/

Negus C. 2010. Linux bible 2010 edition. USA:Wiley publishing.

Ohje tietoturvallisuuden arviointilaitoksille. PDF document on Viestintavirasto website. Accessed on 15 March 2014. Retrieved from https://www.viestintavirasto.fi/attachments/Ohje_tietoturvallisuuden_arviointilaitoksille.pdf

Ohje tietoturvallisuuden arviointilaitoksille. PDF document on Viestintävirasto website. Accessed on 25.2.2014. Retrieved from https://www.viestintavirasto.fi/attachments/Ohje_tietoturvallisuuden_arviointilaitoksille.pdf

SANS education material. Page on SANS website. Accessed on 1 April 2014. Retrieved from http://www.sans.org/reading-room/whitepapers/detection/http-header-heuristics-malware-detection-34460

Sayana, Article isaca.org, Accessed on 15.2.2014. Retrieved from http://www.isaca.org/Journal/Past-Issues/2002/Volume-1/Pages/The-IS-Audit-Process.aspx

Scambray, Mcclure. 2008. Hacking Exposed Windows: Windows Security Secrets & Solutions 3rd edition. USA:Mcgraw-Hill.

Schrenk M. 2012. Webbots, Spiders and screen scrapers, 2nd edition. San Francisco:No Starch Press Inc.

Schweitzer D. 2003. Incident Response: Computer Forensics Toolkit, page 2. Indianapolis: Wiley Publishing

Scripting with Windows PowerShell. Page on Microsoft techent website. Accessed on 5 October 2014. Retrieved from http://technet.microsoft.com/en-us/library/bb978526.aspx

Sikorski. Honig. 2012. Practical malware analysis, San Francisco:No Starch Press Inc.

Ssldump homepage. Page on Rtfm website. Accessed on 4 November 2013. Retrieved from http://www.rtfm.com/ssldump/

The IS audit process. Page on Isaca website. Accessed on 15 March 2014. Retrieved from http://www.isaca.org/Journal/Past-Issues/2002/Volume-1/Pages/The-IS-Audit-Process.aspx

Tshark manual page. Page on Wireshark website. Accessed on 2 January 2014. Retrieved from http://www.wireshark.org/docs/man-pages/tshark.html

Unix Stat Command: How To Identify File Attributes. Page on Thegeekstuff website. Accessed on 12 March 2014. Retrieved from http://www.thegeekstuff.com/2009/07/unix-stat-command-how-to-identify-file-attributes/

Varsalone, McFadden. 2012. Defense against the Black Arts. CRC Press.

Villeneuve,Bennett. 2012. Detecting APT Activity with Network Traffic Analysis. PDF document on Trendmicro website. Accessed on 2.11.2013. Retrieved from http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-detecting-apt-activity-with-network-traffic-analysis.pdf

Wallen J. 2010, Five tips for dealing with rootkits. Blog on techrepublic.com. Accessed on 1.3.2014. Retrieved from http://www.techrepublic.com/blog/five-apps/five-tips-for-dealing-with-rootkits/

Weymes B. 2013. DNS anomaly detection: Defend against sophisticated malware. Article on net-security.org website, Accessed on 5 November 2013. Retrieved from https://www.net-security.org/article.php?id=1844

Windows sysinternals. Page on Microsoft technet website. Accessed on 1 March 2014. Retrieved from http://technet.microsoft.com/en-us/sysinternals/default.aspx

Yhdyskaytavaratkaisuohje. PDF document on Viestintavirasto website. Accessed on 20 March 2014. Retrieved from https://www.viestintavirasto.fi/attachments/Yhdyskaytavaratkaisuohje.pdf

Zalewski M. 2005. Silence On The Wire - A Field Guide To Passive Reconnaissance And Indirect Attacks. San Francisco:No Starch Press Inc.