OULUN YLIOPISTO
UNIVERSITY of OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Ossi Herrala

SECURE DEPLOYMENT STORY FOR CHALLENGING ENVIRONMENTS

Bachelor's Thesis
Degree Programme in Electrical Engineering
Aug 2017

Herrala O. (2017) Secure Deployment Story for Challenging Environments. University of Oulu, Degree Programme in Electrical Engineering. Bachelor's thesis, 29 p.

ABSTRACT

Network based automatic installation of an operating system is and has been a crucial method to manage masses of computers. Protecting every step of the installation is important. Ability to trust the installation system to complete operating system installation safely and to produce a secure installation is an important step in the information security life cycle.

This thesis reviews past and current state of network based automatic installation systems based on what protocols they use. Then it continues to identify risks to those protocols and to study how the situation could be improved by using cryptography with Transport Layer Security (TLS) protocol and digital signatures. This thesis shows how these two existing technologies can be used to provide a more secure installation system.

A proof of concept implementation using TLS and digital signatures is specified, developed and then compared to an already existing and publicly available installation system. Finally the proof of concept implementation is tested and shown to detect and prevent man-in-the-middle attacks.

Keywords: network protocol, operating system, installation system, network security

Herrala O. (2017) Turvallinen käyttöönotto haastavissa ympäristöissä. Oulun yliopisto, sähkötekniikan tutkinto-ohjelma. Kandidaatintyö, 29 s.

TIIVISTELMÄ

Automaattinen käyttöjärjestelmän asennus verkon yli on ollut ja on edelleen tärkeä tapa hallita isoja määriä tietokoneita. Jokaisen asennusvaiheen suojaaminen on tärkeää. Mahdollisuus luottaa asennusjärjestelmän toimivan turvallisesti ja tuottavan turvallisen asennuksen on tärkeää tietoturvan elinkaaressa.

Tämä kandidaatintyö käy läpi miten asennusjärjestelmät ovat toimineet ennen ja nykyisin sen perusteella mitä protokollia ne käyttävät. Seuraavaksi tunnistetaan prokotolliin liittyviä riskejä ja miten tilannetta voisi parantaa käyttämällä salausta (Transport Layer Security, TLS) ja digitaalisia allekirjoituksia. Kandidaatintyö osoittaa, että näitä kahta olemassaolevaa teknologiaa voi käyttää tuottamaan turvallisemman asennusympäristön.

Asennusjärjestelmästä määriteltiin ja toteutettiin soveltuvuusselvitys hyödyntäen TLS-protokollaa ja digitaalisia allekirjoituksia. Tätä järjestelmää verrattiin julkisesti saatavilla olevaan asennusjärjestelmään. Lopuksi toteutettua järjestelmää testataan ja todetaan sen havaitsevan ja estävän välimieshyökkäykset.

Avainsanat: verkkoprotokolla, käyttöjärjestelmä, asennusjärjestelmä, verkon tietoturva

TABLE OF CONTENTS

FOREWORD

I have been writing software, and designing, building, operating and tearing down Linux and UNIX systems for two decades. Installing the operating system has been an important part of the life cycle for Linux and UNIX systems.

This thesis was motivated by the need for modernizing operating system installation over Internet. It is a big subject with lots of little details to work on. This thesis is just a small but important part of it.

I hope my work with this thesis can contribute back to everyone who has made my life so much easier with their installation systems.

I would like to thank my supervisor Prof. Juha Röning for the opportunity to do this thesis for Oulu University Secure Programming Group (OUSPG). Huge thanks to Christian Wieser for continuously taking the time to sit down with me for followup and pushing me forward.

Thanks to everyone who attended OUSPG Open 2016[1] and especially to everyone who participated in the Thesis Review session, and read my work and gave feedback. Truly appreciated.

Thanks to my dad and sister for taking the time to read my work and giving valuable feedback. Hugs to my spouse Ella, who not only read and commented on my work, but also endured the process of me writing it while also finishing her master's degree.

[1] https://github.com/ouspg/ouspg-open

ABBREVIATIONS

BOOTP	Bootstrap Protocol (IETF)
CIA	Confidentiality, Integrity, Availability
DHCP	Dynamic Host Configuration Protocol (IETF)
DNS	Domain Name System (IETF)
DNSSEC	Domain Name System Security Extensions (IETF)
FTP	File Transfer Protocol (IETF)
GPG	The GNU Privacy Guard
HTTP	Hypertext Transfer Protocol (IETF)
HTTPS	HTTP over TLS
IETF	Internet Engineering Task Force
IP	Internet protocol (IETF)
IPMI	Intelligent Platform Management Interface
InfoSec	Information Security
MitM	Man-in-the-Middle
NFS	Network File System (IETF)
NSA	National Security Agency
PXE	Preboot Execution Environment
RARP	A Reverse Address Resolution Protocol (IETF)
RFC	Request for Comments (IETF)
RSA	Public-key cryptosystem named after Ron Rivest, Adi Shamir and Leonard Adleman
TFTP	Trivial File Transfer Protocol (IETF)
TLS	Transport Layer Security (IETF)
UDP	User Datagram Protocol (IETF)
URL	Uniform Resource Locator
USB	Universal Serial Bus
USENIX	The Advanced Computing Systems Association
initrd	initial ramdisk

1. INTRODUCTION

Loading an operating system into computer remotely over the network ("network booting", "diskless booting") has been used for decades. Network booting can be used to bootstrap operating system installation ("network installation") or it can be used for diskless nodes to load the operating system and to run it using a disk provided by server [1].

Usually network installation systems are built to serve a single organization (e.g. a single business) inside their own networks (business intranet) to achieve repeatable and homogeneous installations. Preferably, the installation is expected to be as easy and as fast as possible, and to require as little as possible human interaction.

Many Linux distributions offer a "net install" which is a small image used to boot the computer into a state where rest of the installation software and packages can be downloaded directly from Internet. After downloading files the installer is executed. It runs a set of steps, and finally a fresh operating system installation is completed on the computer.

This thesis has four parts. Introduction looks at the history and current state of installation systems (how it has been done in the past and how it works currently) and then identifies what network based risks there are for an installation system. The next chapter contains design principles for a proof of concept installation system which tries to mitigate against the identified risks. Then the proof of concept implementation is compared against another installation system. Finally in the conclusions and discussion chapter the findings are summarized and recommendations are given for new research and development, and how the security of installation systems could be improved further.

The purpose of this thesis is to identify network protocol based risks in current installation systems and to show how cryptography can be used to improve security and to eliminate network based risks.

1.1. Information Security

One definition of information security (InfoSec) is given in Finnish legislation, Government Decree on Information Security in Central Government (681/2010), which states that "information security means administrative, technical and other measures and arrangements to comply with secrecy obligations and restrictions on use related to information, as well as to ensure access to information and its integrity and availability" [2].

The Finnish national security audit criteria ("Information security audit tool for authorities" or "Katakri" for short) further divides information security into three separate divisions called security management, physical security and information assurance [3].

Katakri's security management is about how information security should be built-in into an organization from management down to each individual person. Security management contains subjects like the organization's security principles, security management tasks and responsibilities, risk management, continuity and personnel security.

As the name implies, physical security consists of protecting information from physical access. "The aim of physical safeguards is to deny surreptitious or forced entry by intruders, to deter, impede and detect unauthorised actions and allow for segregation of personnel in terms of access to Classified Information on the need-to-know basis” [3].

Katakri’s information assurance category contains information security requirements for electronic information.

1.1.1. Information assurance

Katakri’s information assurance requirements are divided into four separate sections: communications security, systems security, data security and operations security.

Communications security is about computer networks, devices connected to such networks and networks connected to other networks.

Systems security is about access controls, privileges and authorizations when using computers and computer networks. Further when using the systems, proper audit logging, protection against unwanted software, and incident detection and recovery are required.

Data security is about keeping the secrets secret when the data is either stored somewhere or moved from one place to another.

Operations security contains day to day tasks for managing the information processing environment life cycle, for example change management, backups and software vulnerability management. Operations security also contain requirements for the handling and transfer of classified information.

1.1.2. This thesis in Information Security landscape

Security of an installation system can be place into the information assurance section of Katakri. Further, it can be categorized under operations security. In operations security, the role of an installation system is at the very beginning of information life cycle management. A secure installation system takes care of setting up an appropriate and properly secured operating system installation to a computer which then can be trusted.

The implementation of an installation system needs to take care of the requirements in communications, systems and data security to be able to achieve a secure installation.

In this thesis it is assumed that proper management security and physical security are already in place.

1.2. Role of Humans

Information security is not only a technical issue. Since humans operate computers, networks and software, information security is also a people problem [4][5]. Roughly the people problem can be divided into two categories: psychological attacks against humans (social engineering) and “getting things done”.

Social engineering is an attack against human psychology and cognitive biases. Attacks like phishing or pretexting are used to exploit human weaknesses and the willingness to provide information like user names, passwords or credit card data. A person under a social engineering attack might think she is not providing anything sensitive or harmful, but the attacker could use bits and pieces of information from multiple attacks to gain whatever she is looking for [6][5].

Another social engineering attack worth mentioning is tailgating or piggybacking [7]. In case of successful tailgating or piggybacking, the attacker gains physical access to premises which can then lead to e.g. stealing of information or assets like computers, physical keys, ID badges and money. Or, the attacker might be planting hardware like a physical key logger, listening device or USB (Universal Serial Bus) memory with malware.

In contrast to a social engineering attack against humans is the “getting things done” scenario where human is just trying to get through the day without any intention to do anything malicious. An employee has something which needs to be done, maybe under stress and pressure, and technical information security measures like a pop-up window alarming user about something or even a window asking for user password, are a distraction from the task at hand. However, such pop-ups are so common that humans are constantly used to click “OK” to continue without even reading nor thinking about the reason or content of such notifications [5].

In the case of installation systems, the “getting things done” is the more probable information security issue. Imagine an integration engineer with a tight schedule working in the customer’s premises trying to get computer systems up and running. Any problem or obstacle increases stress and anxiety, and just “getting things done” without worrying about information security risks becomes more likely.

1.3. Involved Protocols

Multiple network protocols have been developed and used to allow booting using IP (Internet Protocol) networks. Early published standards include RARP (“A Reverse Address Resolution Protocol”, RFC903, published 1984 [8]) and BOOTP (“Bootstrap Protocol”, RFC951, published 1985 [9]) which could be used to allow “a diskless client machine to discover its own IP address” [9], and TFTP (“Trivial File Transfer Protocol”, RFC783, published 1981 [10]) which “may be used to move files between machines on different networks implementing UDP.” [10].

Later RARP and BOOTP were superseded by DHCP (“Dynamic Host Configuration Protocol”, RFC1531, published 1993 [11]), while TFTP became superseded by NFS (“Network File System”, RFC1094, published 1989 [12]), which “provides transparent remote access to shared files across networks” [12]. PXE (“Preboot Execution Environment” [13]) is a specification from Intel Corporation to standardize the preboot environment for network booting. In some cases TFTP or NFS or both can be replaced with HTTP (“Hypertext Transfer Protocol” [14][15]).

1.4. Identifying Risks

The CIA triad [16] divides network security into three elements: confidentiality, integrity and availability [5].

Confidentiality means that the sender of the message encodes the content so that only a receiver can decode it and see the message. Confidentiality can be achieved using encryption. Encryption is the process of encoding a message using a secret key so that decryption is only possible with the correct secret key. Keys used for encryption and decryption might not be the same. Confidentiality can be achieved in a network for example by using the TLS protocol [17] to encrypt network traffic.

Integrity control guarantees that a message cannot be modified during transfer. It consist of two parts: Non-repudiation and authenticity.

Non-repudiation ensures proof of integrity and the origin of data. This is usually achieved with using authentication and integrity control. Digital signatures [18][19] can provide non-repudiation. Standards such as S/MIME [20] or OpenPGP [21] can be used for digital signatures.

Authenticity ensures that the receiver, transmitter or both parties determine they are communicating with another intended party before exchanging any confidential messages. The TLS protocol provides means to verify the authenticity of communicating parties [17].

Availability means that systems are up and operational. Perfect security could be achieved by turning everything off, but such systems cannot be used. It is important to ensure availability so that network services can be used. Availability can be achieved by allocating enough human and computing resources to operate the services such as facilities, computer systems and networks.

Risks can be identified in all components from hardware to operating system vulnerabilities. Table 1 lists some common known attacks which could be targeted towards network booting or network installation systems.

All components in Table 1 are susceptible to issues with availability. For example, if the network or one or more components are not available, the whole stack of components is inoperative.

Table 1. Roles and risks of various components used in operating system installations over network

Component	Role	Risks
HTTP	File transfer	confidentiality, integrity
DNS	Name service	non-repudiation
NFS	File transfer	confidentiality, integrity
TFTP	File transfer	confidentiality, integrity
DHCP	Address resolution	non-repudiation

DHCP and DNS (Domain Name System, RFC1035 [22]) protocols could be used to redirect (“hijack”) future communications towards malicious services [23][24].

DHCP is commonly used to assign an IP address (Internet Protocol address) to a client and to give it various bits of information such as the IP address of a TFTP or DNS server. A malicious DHCP server could take over the following TFTP and DNS communications.

DNS has many uses, but commonly it is used to translate a host name into an IP address. A malicious DNS server could redirect future communications towards malicious services.

TFTP, NFS and HTTP are used to transfer files between a client and server. A malicious or compromised file server could be used to deliver malicious files to a client which upon execution could compromise the client operating system installation or even infect the hardware the operation was performed in.

There has been efforts to secure DHCP and DNS. However, that requires the network, clients and servers to be configured to take these security measures into action. But the risks can be detected by other components (e.g. using TLS's server authentication, and digital signatures), so there is no need for changes to network configuration. The installation can be done securely in any network and if something malicious is detected, the installation process can halted.

Hardware (e.g. a physical server or laptop) and peripherals (e.g. displays, keyboards, mice, removable medias) can have backdoored firmware [25]. The backdoors could have been installed already at the factory or the firmware has been infected with some malware that has been previously executed on the machine. Mitigations for risks against malicious hardware is out of scope for this work.

1.5. Current state

Software deployment technologies [26], securing virtual machines [27] as well as cloud computing security challenges [28][29] have been widely studied. However, the network installation of an operating system is still much the same as in the 1980s, and it still forms the basis for operating system installation.

Alpine Linux's PXE Boot HOWTO [30] summarizes the current situation:

> Alpine can be PXE booted starting with Alpine 2.6-rc2. In order to accomplish this you must complete the following steps:
>
> 1. Set up a DHCP server and configure it to support PXE boot.
> 2. Set up a TFTP server to serve the PXE boot loader.
> 3. Set up an HTTP server to serve the rest of the boot files.
> 4. Set up an NFS server from which Alpine can load kernel modules.
> 5. Configure mkinitfs to generate a PXE-bootable initrd.

Alpine Linux's documentation was chosen as an example because of their claim that it is "for power users who appreciate security, simplicity and resource efficiency" [31]. Similar setup is required for other Linux distributions like Red Hat [32], and for Microsoft Windows [33].

As can be seen, the whole process still relies on old protocols DHCP, TFTP, HTTP and NFS developed around 1980–1990. However, these protocols provide no security and should not be used in networks.

TFTP, NFS and HTTP protocols could be replaced with HTTPS (HTTP over TLS) where TLS (Transport Layer Security Protocol [17]) provides communications security using cryptography and authentication of one or both communicating parties.

DHCP is the standard protocol to centrally manage IP addresses for clients. It is difficult to replace, so its shortcomings need to be countered with other means.

DNS is also a vital protocol to the Internet. It provides translation from host names to IP addresses and back (and other name services). DNS is also a standard protocol. Work to protect DNS traffic has been done (DNSSEC, RFC4035 [34]) and DNSSEC is slowly getting a foothold to protect DNS communications. A risk for DNS in installation systems arises from man-in-the-middle attacks. Without DNSSEC it is possible to continue to use DNS and use other means outside of DNS protocol to verify that DNS is working as it should.

Shortcomings of all of these protocols and how to mitigate against the risks are discussed later.

1.6. Challenging environments

Computer networks are not safe nor secure [35]. Internet is likely the lest safe among networks. Only one compromised device in a network is required to make the whole network unsafe. Connections in the Internet do not observe national borders and travel through different areas of laws and regulations. Protocol packets are passed from one Internet service provider to another. On every step of the connection's path someone might be listening or even altering the connection to their own agenda. It might be a governmental body (like NSA's PRISM program [36]), a criminal organization that has gained a foothold somewhere in the network or simply a curious individual just being able to do so.

Same problems can also be present in networks like business intranets where both governmental and criminal organizations might have gained a foothold to operate in. In USENIX Enigma 2016 conference Rob Joyce, Chief of Tailored Access Operations in National Security Agency describes how his team infiltrates networks and moves there laterally to gain what they are after [37]. Therefore, intranets should be treated with same level of mistrust as Internet.

1.7. Mitigation

Risks can be mitigated by using trusted media, secure communication channels and cryptographically signed files.

A boot environment should be loaded from trusted media, for example using prebuilt USB mass media. This media contains software and files to safely load next steps required to load the operating system kernel and other files safely over the network.

Network communication is done using HTTPS with X.509 certificate pinning. This authenticates the remote server and makes it harder to perform man-in-the-middle attack against the connection. If a secure channel cannot be established, the boot process should be halted.

Signed files are used to ensure the authenticity of files used for booting. For example many Linux distribution mirrors only provide files via HTTP or FTP servers which are susceptible to man-in-the-middle attack. If the signature check fails, the boot process should be halted.

1.8. Comparison to virtualization and cloud

Installing an operating system into a virtual machine (in a cloud or other virtualization platform) enjoys many benefits compared to installation to a physical hardware. Virtualization provides easier “programmable” access to every state of virtual machine installation from setting up the machine itself and its parameters (like processors, memory amount, disk space, network) to pre-building ready operating system images (machine images) to be booted in the cloud. This is called “Infrastructure as Code” or IaC [38].

Infrastructure as Code can be achieved for example with tools like Packer [39] which can be used to build machine images, and Terraform [40] to set up virtual machines and launch machine images to produce running virtual machines. Both tools use a simple description language where operations can be specified and then run using the tool itself.

When building machine images, it is possible to have operating system installation files on local disk so that no network access is required. Many operating systems also provide means to download the installation files beforehand and to verify their authenticity. Or in the case of cloud environments, it is possible that the user can use pre-made machine images provided by the cloud provider.

However, physical computers require physical access in order to be able to plug in devices and cables, to turn power on and to control the first stages of startup before the operating system is running.

There are remote management solutions like Intelligent Platform Management Interface (IPMI), which make it possible to remotely control physical hardware. IPMI is a specification of interfaces for monitoring and controlling physical computer hardware remotely via network. Using IPMI it is possible to instruct a computer to turn on or off, and to control BIOS settings [41]. However, using IPMI still requires physical connections to be made and IPMI to be properly configured.

2. IMPLEMENTING SECUDEP

To see if it is possible to use HTTPS and digital signatures, a simple proof of concept implementation of an installation system called *secudep* was implemented. Source code can be found from secudep's project site on GitHub [1].

Secudep's implementation has three main design principles: ease of use, ease of deployment and security. Deploying a new installation system should be easy so that it encourages building small, easy to update and easy to maintain setups. Ease of deployment might also attract people to develop new use cases and applications on top of an already existing system. With the implemented solution there should be no need to have a monolithic and centralized installation system, and instead designs can shift more towards personal or per application installation systems.

An installation system should help the end user achieve a fresh installation of operating system as well as install applications as easily, smoothly and as fast as possible. Most of the decisions required for installation should be made beforehand and automatized as much as feasible.

Security is a more difficult design principle to tackle. An installation system should concentrate on selecting and enforcing safe defaults, and to guide the user to make safe choices.

This proof of concept implementation uses public-key cryptography to digitally sign files so that the authenticity of those files can be verified. Regenerating new key material should also be encouraged when updating signature files. This renders old installation systems unusable and forces updating of installation media (for example USB mass media).

As an example, one security design principle is to halt the installation process when a security measure detects an anomaly. An example of such an anomaly could be an active man-in-the-middle attack. If the user is given a choice to continue, she usually does so, probably without understanding or investigating what caused the issue, and thus rendering the security measure useless and allowing the attack.

This implementation borrows lots of ideas and lessons learned from boot.foo.sh [42] and from the installation system used by Faculty of Information Technology and Electrical Engineering in University of Oulu.

2.1. Tools

Secudep uses iPXE [43] as network boot firmware. iPXE is a PXE [13] implementation with additional features such as support for booting via HTTP [15] protocol. Support for HTTPS can also be compiled in.

Secudep's iPXE binary build is done inside container using the Docker [44] software containerization platform to achieve repeatable builds with managed dependencies. Docker is a platform to easily build operating system level virtualization [45] containers. Instead of virtualizing the hardware like Xen or KVM, containers use the operating system's namespaces to separate containerized applications from each other. Docker is not mandatory for producing the build.

[1] https://github.com/ouspg/secudep

Python programming language, bash shell scripts and OpenSSL are used to build individual parts of the system.

2.2. Setting up the installation system

Setting up the installation system using secudep consists of the following steps. After the list, all of the steps are explained further.

1. Generate digital signing keys
2. Collect HTTPS servers' X.509 certificates for public key pinning
3. Build iPXE bootable media
4. Write configuration file
5. Generate contents for deployment

Digital signing keys are generated when deploying the installation system. A private key is used to produce the signatures and a public key is embedded into the installation image.

The installation system is deployed to an existing HTTPS server. Thus this server's X.509 certificate can be fetched and embedded into the installation image. This is now the only X.509 certificate to be trusted, and no other HTTPS server can be used.

After the digital signing keys have been generated and X.509 certificates have been fetched, it is possible to build the bootable installation media. This installation media file is written e.g. to a USB memory, and can be used to launch the operating system installation in a computer.

A configuration file binds things together. It specifies the HTTPS server, where the keys and certificates are, what operating systems can be installed, and where the required files can be found.

After all other steps are done, the files to be deployed on the HTTPS server can be generated. This step fetches all of the required files, calculates digital signatures, creates various boot scripts, and builds a directory structure which should then be mirrored on the HTTP server.

Future work on secudep should simplify these steps even further. Digital signing keys could be automatically generated if missing, X.509 certificate collection could be automated based on secudep's configuration file. iPXE media build could also be done every time contents for deployment are generated.

2.3. Deploying

Everything needed for the installation system to operate (from the server side) is generated under one directory. This directory can then be published on the HTTPS server. The URL for the installation system files is configured in secudep's configuration file.

2.4. Security

Secudep makes it as easy as possible to use public key pinning for HTTPS hosts and digital signatures to verify authenticity of files.

iPXE is configured to require trusted files. A file is trusted only after its signature is verified successfully. This requirement cannot be turned off once it is turned on.

3. CASE STUDIES

Three case studies were performed. The case studies build an arch from studying how a current state-of-the-art installation system works towards testing the promise of secudep to making a more secure installation system.

The first case study looks into an already existing installation system to verify what protocols are used in the process. This is done to verify what is written in the introduction chapter about the current state of the art.

Next case study compares results from the first case study with the implementation details of secudep. The purpose is to compare how an already existing installation system differs from secudep.

Third and last case study looks into secudep's promise to make a more secure installation system. This is done by simulating the attack scenarios, for example man-in-the-middle attack, and observing how secudep behaves.

3.1. Case Study 1: Identify Protocols

3.1.1. What was studied

The purpose of this case study is to identify network protocols used in an online installation system system. This study also verifies the involved protocols which were described in the introduction chapter.

A service called boot.foo.sh [42] is used as an installation system. Boot.foo.sh was chosen because it is an open service known to be used for automatic installations in enterprises and it has been an inspiration to this thesis to make installation system safer to use.

Boot.foo.sh is used to install the CentOS 7 Linux operating system. CentOS Linux is a community driven effort to provide a free alternative to Red Hat Enterprise Linux (RHEL). CentOS is built using RHEL source code. Red Hat has 67 % market share of the market for Linux distributions according to Gartner's analysis [46].

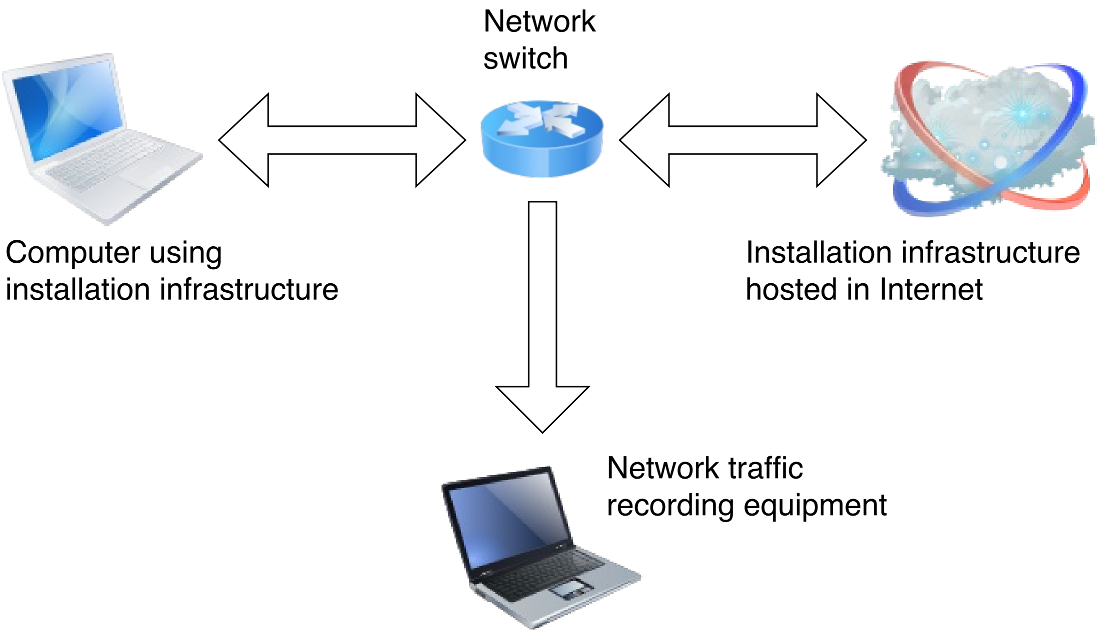
3.1.2. How it was done

The installation was done using a virtual machine. VirtualBox was chosen as virtualization software because it is free, open source and has an easy to use network traffic recording functionality.

With VirtualBox's network traffic recording it is possible to get network traffic captured for the whole lifetime of a virtual machine. The capture is saved as a standard PCAP file which can later be opened in a network protocol analyzer for investigation. Figure 1 has the typical traffic capturing setup with a computer using the installation system, another computer recording the traffic, a network switch to arrange traffic flows and the Internet containing the installation system in use.

The traffic capture was then analyzed using Wireshark network protocol analyzer. Wireshark is a free and open source network protocol analyzer which has the capability to help an expert user analyze many different network protocols and their internals.

Figure 1. Network traffic recording setup.



Traffic analysis was done by hand, looking at the captured traffic recording and identifying the protocols used.

3.1.3. Results found

Traffic recording was 788 megabytes of network traffic containing over 883000 network packets. The Recording contains spans just over nine minutes. This time span contains all of the network traffic from the startup of the virtual machine to the end of operating system installation.

Table 2. Table of found protocols and their roles. DS in table means Digital Signatures.

Step	Protocol
Address resolution	DHCP
Name resolution	DNS
Boot menu	HTTPS
Kernel and initrd	HTTP
Kickstart	HTTP
Installation files	HTTP (DS)

Summary of the protocols used during various steps of the installation process can be found from Table 2.

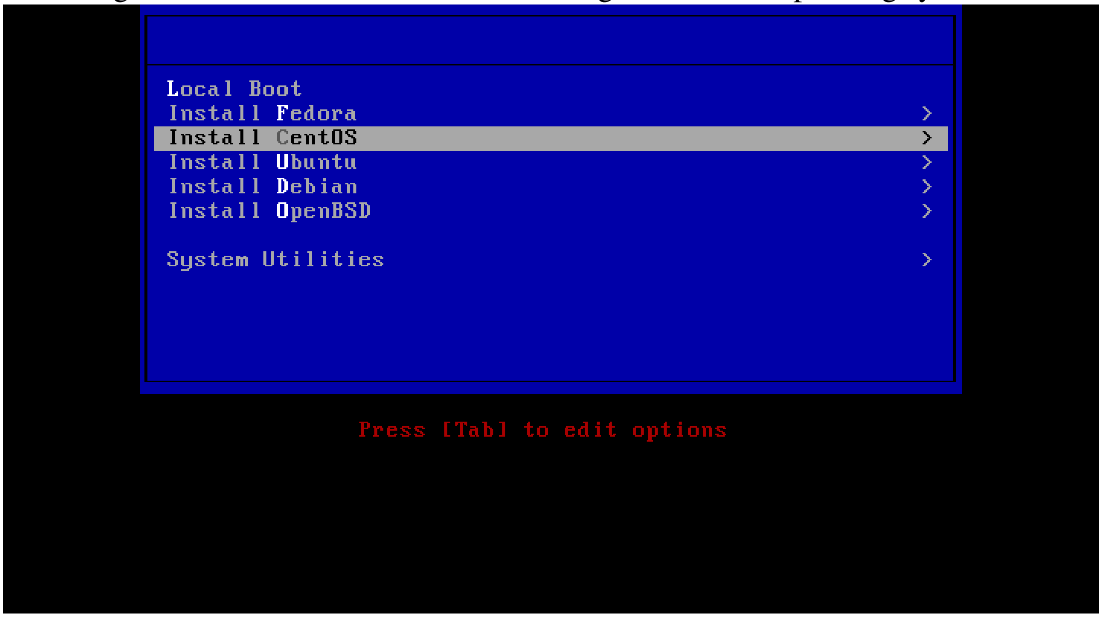
Installation steps are identified and named for each installation system used. The steps are discussed in chronological order of appearance as found from traffic recording.

3.1.4. Analysis of results

“Address resolution” is the first step and its purpose is to get the IP address and DNS server addresses for the system to be installed. DHCP is the standard protocol for this, and it was found to be used also in this case.

“Name resolution” is used to translate host names into IP addresses to communicate with other servers. DNS protocol is used for name resolution needs.

Figure 2. boot.foo.sh boot menu showing selection of operating systems.



“Boot menu” is used to display the operating systems available for installation. Boot menu from boot.foo.sh can be seen in Figure 2. Boot.foo.sh uses HTTP protocol to fetch various files needed to display the boot menu.

“Kernel and initrd” are the files needed to launch Linux installation. These two files are downloaded over the Internet, the kernel is executed and it continues the boot process. HTTP was used to communicate with a CentOS 7 mirror to fetch the needed files.

“Kickstart” is a CentOS specific file for automating unattended installation. It is a set of instructions downloaded and executed by the installation process. The kickstart file is downloaded by software inside the initrd system, so at this point the control of installation is already switched over to the CentOS operating system’s own installer. HTTP was used to communicate with boot.foo.sh server to fetch the kickstart file.

“Installation files” are the contents of the operating system to be installed. The files are downloaded and extracted to the local hard drive for installation. The operating system installer is usually trusted to verify the digital signatures of the downloaded content before extracting the files into the hard drive. For example CentOS uses OpenPGP [21] (“GPG”) signatures. The CentOS documentation [47] states that

> “Each stable RPM package that is published by CentOS Project is signed with a GPG signature. By default, yum and the graphical update tools will verify these signatures and refuse to install any packages that are not

> signed, or have an incorrect signature. You should always verify the signature of a package prior to installation. These signatures ensure that the packages you install are what was produced by the CentOS Project and have not been altered by any mirror or web site providing the packages."

However, before installation files with related GPG digital signatures can be verified an attacker could have replaced the kernel or initrd from previous step with a compromised version and thus the finished installation could become compromised. Previous steps must be protected before the operating system's own installation files in order to enable the installer's own protections.

3.2. Case Study 2: Comparing boot.foo.sh and secudep

3.2.1. What was studied

This case study compares implementation details of secudep to an already existing installation system solution which was studied in case study 1. The purpose of this is to see the differences between the network protocols used by these two systems.

3.2.2. How it was done

The results from case study 1 was used as a base, and then implementation details about secudep were compared against the base.

3.2.3. Results found

Table 3. Comparison between how boot.foo.sh and secudep use protocols. DS in table means Digital Signatures.

Step	boot.foo.sh	secudep
Address resolution	DHCP	DHCP
Name resolution	DNS	DNS
Boot menu	HTTP	HTTPS (DS)
Digital signatures	N/A	HTTPS
Kernel and initrd	HTTP	HTTP (DS)
Kickstart	HTTP	HTTPS
Installation files	HTTP (DS)	HTTP (DS)

Results of comparing boot.foo.sh and secudep can be found from Table 3. Boot.foo.sh results are same as in case study 1. The differences between boot.foo.sh and secudep are discussed next.

3.2.4. Analysis of results

Address and name resolution steps are identical in both systems. As discussed in the introduction chapter, these protocols are standards and therefore difficult to change.

”Boot menu” is used to display the operating systems available for installation. HTTP used by boot.foo.sh is susceptible to man-in-the-middle attack. Secudep uses HTTPS (HTTP over TLS) with signed files to remediate this issue.

Secudep uses digital signatures that are fetched over HTTPS. This is a step missing from boot.foo.sh.

Kernel and initrd are the files needed to launch the Linux installation. Both boot.foo.sh and secudep systems use the HTTP protocol. Again HTTP is susceptible to man-in-the-middle attacks. HTTP is used because the files are fetched from CentOS's official mirror over the Internet. Secudep uses digital signatures to verify downloaded content. After kernel and initrd are downloaded and digital signatures are verified, the execution is handed over to the kernel. This means that secudep cannot provide digital signatures for any following files.

3.3. Case Study 3: Testing attacks against secudep

3.3.1. What was studied

This case study consists of simulated attacks against an implementation of secudep. Secudep's main defense against attacks is the use of encryption (TLS) and digital signatures.

Table 3 contains a list of protocols involved in the operating system installation process.

The first two protocols, DHCP for address resolution and DNS for name resolution, are insecure and susceptible for example to man-in-the-middle attacks. Loading the boot menu over HTTPS with a digital signature check should validate that DHCP and DNS are not tampered with and that the installation can proceed further.

After the boot menu step is done, secudep loads kernel and initrd over an unsecured HTTP connection. A man-in-the-middle attack could change the kernel or initrd files to something else, but digital signature verification should notice that and prevent running possibly malicious content.

3.3.2. How it was done

Secudep boot media contains at least two public keys. One is for digital signature verification, and one or more are for verifying HTTPS connections. More public keys are loaded over a HTTPS connection while the boot progresses.

Testing that these verifications work can be done by either omitting the public key from secudep boot media or serving the wrong public key from secudep's HTTPS server.

Table 4 lists the steps during the boot process and what verification methods are used to defend against attacks.

Table 4. Attacks and their defenses during each installation step. DS in table means Digital Signatures.

Attack against	Protocol	Defense
Address resolution	DHCP	Verification done in boot menu
Name resolution	DNS	Verification done in boot menu
Boot menu	HTTPS (DS)	Certificate and digital signature verification
Digital signatures	HTTPS	Certificate verification
Kernel and initrd	HTTP (DS)	Digital signature verification
Kickstart	HTTPS	Certificate verification
Installation files	HTTP (DS)	Operating system takes control

DHCP and DNS man-in-the-middle attacks can be detected when X.509 certificate verification fails, and HTTP man-in-the-middle attacks can be detected when code signing verification fails.

Table 5. Different files used for verification during each step, where they are used and where they are located. DS in table means Digital Signatures.

File	Used for	Where it is located
DS certificate	Verify digital signatures	in bootable media
X.509 for HTTPS	Verify HTTPS connection(s)	in bootable media
Boot menu (DS)	Verify boot menu is not changed	File on HTTPS server
Kernel (DS)	Verify kernel is not changed	File on HTTPS server
Initrd (DS)	Verify initrd is not changed	File on HTTPS server

Table 5 lists the files used during each step in the installation process. Any failure in verification should halt the installation process.

3.3.3. Results found

Table 6. Results of testing secudep's implementation with simulated attacks.

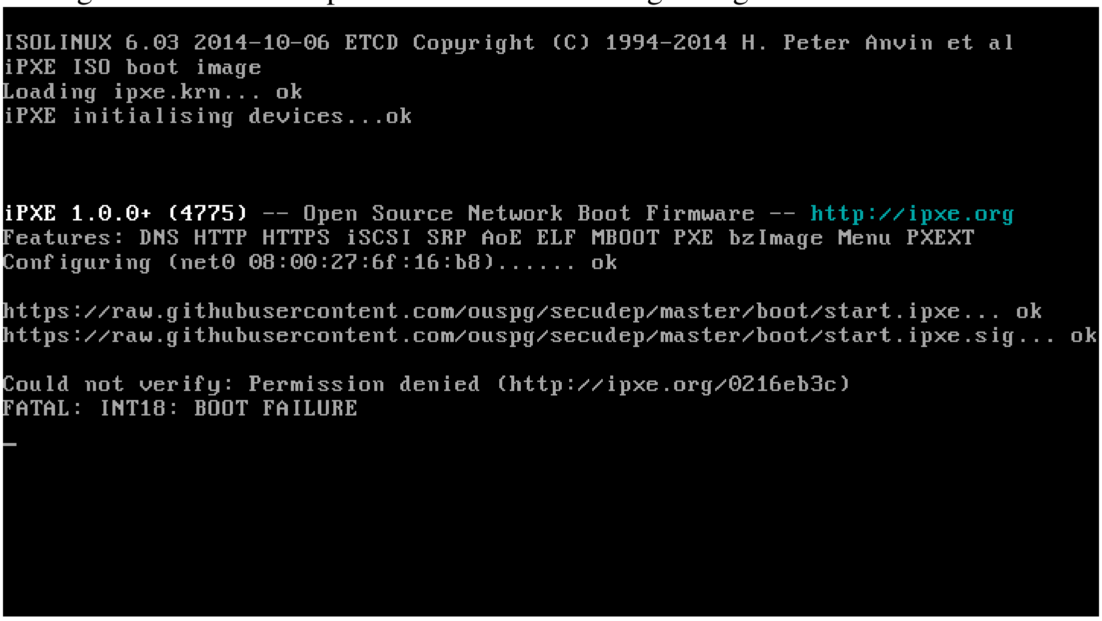
File	Halts installation	iPXE error code
DS certificate	True	0216eb3c
X.509 for HTTPS	True	0216eb3c
Boot menu DS	True	0227e13c
Kernel DS	True	0227e13c
Initrd DS	True	0227e13c

Five different tests were made, with their results displayed in Table 6. Every simulated attack was noticed and the installation was halted. Table also gives iPXE's error code for each tested case.

3.3.4. Analysis of results

Five different tests were made by breaking one verification step at a time and trying to perform the installation. The result was observed and materials collected.

Figure 3. Installation process is halted when digital signature verification fails.



Two distinct iPXE error codes were found while conducting the tests. One example of these error is shown in Figure 3. The installation process could not verify the digital signature. The process was halted and it could not proceed further.

The first error code, "0216eb3c" is documented[1] on the iPXE web page as "Error: No usable certificates". This matches what was tested. In the test a wrong certificate was provided, so the error given is correct.

The second error code "0227e13c" is documented[2] on the iPXE web page as "Error: RSA signature incorrect" with additional notes stating:

> This error indicates that an RSA signature was found to be incorrect.
>
> Things to try:
>
> 1. Check that all certificates are correct.
> 2. If you are verifying a digital signature using the imgverify command, check that you are using the correct signature file.

[1] http://ipxe.org/err/0216eb3c
[2] http://ipxe.org/err/0227e13c

This matches what was tested. Either a wrong RSA signature was given in test, or the file was changed so that the RSA signature verification should fail. This error message is correct.

This case study tested only the most obvious security issues. More sophisticated attacks might exploit the implementation weaknesses in iPXE and other software and hardware. Thus this case study is not proof of perfect security, but shows that at least some cases of attacks can be detected and reacted on.

3.4. Analysis of Case Studies

Three separate case studies were conducted. Case studies identified how a current state-of-the-art installation system operated, and ended in proving that the technologies used in secudep can be used to detect and prevent man-in-the-middle attacks. Security analysis or security testing of a state-of-the-art installation system implementation was not performed.

Comparison between secudep and another installation system showed how secudep introduces cryptography (TLS and digital signatures) into the installation process to protect the installation. Simple testing shows that secudep is capable of preventing attacks using TLS and digital signatures.

4. CONCLUSION

This thesis first reviewed what network based risks could face an installation system and then studied what kind of means could protect the initial phases (before the operating system kernel takes control of execution) of the installation process using encryption and digital signatures.

Protecting every step in network communications is important, and protecting installation systems is no exception. This thesis has shown that it is possible to take a step further towards more secure installation systems by using two technologies: encryption and digital signatures.

More secure systems can be built step by step by combining simple individual components without the need for designing whole new systems and technologies. Replacing old components (like TFTP, NFS and HTTP protocols) with newer but already existing ones (HTTPS protocol) and increasing the use of digital signatures are small steps to take in order to gain a big benefit in security. Furthermore, when using HTTPS it is possible to use different authentication schemes to hide installation scripts (kickstart files, etc.) which otherwise would be visible to the Internet.

Linux distributions and other open source operating systems use OpenPGP or other digital signature methods to protect the installation packages from outside tampering, which is a really good and important thing to do. Some Linux distributions also protect the package database metadata with digital signatures, but some have that functionality turned off by default. Maybe mirrors at some point could take a step forward and enable HTTPS so files like kernel and initrd, and package database metadata could be securely downloaded?

The initrd file also contains a public key to verify digital signatures. Is the initrd file downloaded and verified so that the embedded public key can be trusted by the installation process?

More testing and verification should be performed for iPXE and its TLS implementation and digital signature capabilities. This was intentionally left out from this thesis.

5. REFERENCES

[1] Anvin H.P. & Connor M. (2008) X86 network booting: Integrating gpxe and pxelinux. In: Linux Symposium, p. 9. URL: https://www.kernel.org/doc/ols/2008/ols2008v1-pages-9-18.pdf.

[2] Government decree on information security in central government (681/2010). URL: http://www.finlex.fi/en/laki/kaannokset/2004/en20040516.pdf.

[3] (2015) Katakri - Information security audit tool for authorities. URL: http://www.finlex.fi/en/laki/kaannokset/2004/en20040516.pdf.

[4] Parsons K., McCormac A., Butavicius M. & Ferguson L. (2010) Human factors and information security: individual, culture and security environment. Tech. rep. URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a535944.pdf.

[5] Anderson R.J. (2008) Security Engineering, 2nd edition. URL: https://www.cl.cam.ac.uk/~rja14/book.html.

[6] Greavu-Serban V. & Serban O. (2014) Social engineering a general approach. Informatica Economica 18, pp. 5–14. URL: https://www.researchgate.net/publication/273708458_Social_Engineering_a_General_Approach.

[7] Fairbrother R. (2014) Insider network attack investigation. Discovery, Invention & Application URL: http://commerce3.derby.ac.uk/ojs/index.php/da/article/view/62.

[8] Finlayson, Mann, Mogul & Theimer (accessed 26.5.2016) RFC903: A reverse address resolution protocol. Tech. rep. URL: https://tools.ietf.org/html/rfc903.

[9] Croft B. & Gilmore J. (accessed 26.5.2016) RFC951: Bootstrap protocol (BOOTP). Tech. rep. URL: https://tools.ietf.org/html/rfc951.

[10] Sollins K.R. (accessed 26.5.2016) RFC783: The TFTP protocol (revision 2). Tech. rep. URL: https://tools.ietf.org/html/rfc783.

[11] Droms R. (accessed 26.5.2016) RFC1531: Dynamic host configuration protocol. Tech. rep. URL: https://tools.ietf.org/html/rfc1531.

[12] Nowicki B. (accessed 26.5.2016) RFC1094: NFS: Network file system protocol specification. Tech. rep. URL: https://tools.ietf.org/html/rfc1094.

[13] Berners-Lee T., Fielding R. & Frystyk H. (accessed 18.6.2016) Preboot execution environment (pxe) specification version 2.1. Tech. rep. URL: ftp://download.intel.com/design/archives/wfm/downloads/pxespec.pdf.

[14] Berners-Lee T., Fielding R. & Frystyk H. (accessed 26.5.2016) RFC1945: Hypertext transfer protocol – HTTP/1.0. Tech. rep. URL: https://tools.ietf.org/html/rfc1945.

[15] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P. & Berners-Lee T. (accessed 25.8.2016) RFC2616: Hypertext transfer protocol – HTTP/1.1. Tech. rep. URL: https://tools.ietf.org/html/rfc2616.

[16] Perrin C. The CIA triad. Tech. rep. URL: http://www.techrepublic.com/blog/it-security/the-cia-triad/.

[17] Dierks T. & Rescorla E. (accessed 30.6.2016) RFC5246: The transport layer security (TLS) protocol version 1.2. Tech. rep. URL: https://tools.ietf.org/html/rfc5246.

[18] Diffie W. & Hellman M. (2006) New directions in cryptography. IEEE Trans. Inf. Theor. 22, pp. 644–654. URL: http://dx.doi.org/10.1109/TIT.1976.1055638.

[19] Goldwasser S., Micali S. & Rivest R.L. (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17, pp. 281–308. URL: http://dx.doi.org/10.1137/0217017.

[20] Ramsdell B. & Turner S. (accessed 29.8.2016) RFC5751: Secure/multipurpose internet mail extensions (S/MIME) version 3.2: Message specification. Tech. rep. URL: https://tools.ietf.org/html/rfc5751.

[21] Callas J., Donnerhacke L., Finney H., Shaw D. & Thayer R. (accessed 29.8.2016) RFC4880: OpenPGP message format. Tech. rep. URL: https://tools.ietf.org/html/rfc4880.

[22] Mockapetris P. (accessed 25.8.2016) RFC1035: Domain names - implementation and specification. Tech. rep. URL: https://tools.ietf.org/html/rfc1094.

[23] Green I. (2005) DNS spoofing by the man in the middle URL: https://www.sans.org/reading-room/whitepapers/dns/dns-spoofing-man-middle-1567.

[24] Ornaghi A. & Valleri M. (2003) Man in the middle attacks. In: Blackhat Conference Europe. URL: https://www.blackhat.com/presentations/bh-europe-03/bh-europe-03-valleri.pdf.

[25] Swierczynski P., Fyrbiak M., Koppe P., Moradi A. & Paar C. (2016) Interdiction in practice—hardware trojan against a high-security usb flash drive. Journal of Cryptographic Engineering , pp. 1–13URL: https://www.hgi.rub.de/media/crypto/veroeffentlichungen/2016/05/12/hardware_trojan_fpga.pdf.

[26] Carzaniga A., Fuggetta A., Hall R.S., Heimbigner D., van der Hoek A., & Wolf A.L. (accessed 29.8.2016) A characterization framework for software deployment technologies. Tech. rep. URL: http://scholar.colorado.edu/csci_techreports/806/.

[27] Garfinkel T. & Rosenblum M. (2005) When virtual is harder than real: Security challenges in virtual machine based computing environments. In: Proceedings of the 10th Conference on Hot Topics in Operating Systems - Volume 10, HOTOS'05, USENIX Association, Berkeley, CA, USA, pp. 20–20. URL: http://dl.acm.org/citation.cfm?id=1251123.1251143.

[28] Owens D. (2010) Securing elasticity in the cloud. Commun. ACM 53, pp. 46–51. URL: http://doi.acm.org/10.1145/1743546.1743565.

[29] Hashizume K., Rosado D.G., Fernández-Medina E. & Fernandez E.B. (2013) An analysis of security issues for cloud computing. Journal of Internet Services and Applications 4, pp. 1–13. URL: http://dx.doi.org/10.1186/1869-0238-4-5.

[30] (accessed 26.5.2016), PXE boot howto - Alpine Linux. URL: https://wiki.alpinelinux.org/wiki/PXE_boot.

[31] Team A.L.D. (accessed 26.5.2016), Alpine linux - about. URL: https://alpinelinux.org/about.

[32] Red Hat I. (accessed 25.1.2017.), Red hat 7 installation guide: Preparing for a network installation. URL: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Installation_Guide/chap-installation-server-setup.html.

[33] Lindsay G. (accessed 25.1.2017.), Configure a pxe server to load windows pe. URL: https://technet.microsoft.com/en-us/itpro/windows/deploy/configure-a-pxe-server-to-load-windows-pe.

[34] Arends R., Austein R., Larson M., Massey D. & Rose S. (accessed 25.8.2016) RFC4035: Protocol modifications for the DNS security extensions. Tech. rep. URL: https://tools.ietf.org/html/rfc4035.

[35] Ward R. & Beyer B. (2014) Beyondcorp: A new approach to enterprise security. ;login: Vol. 39, No. 6, pp. 6–11.

[36] Bowden C. (accessed 9.7.2016) The US surveillance programmes and their impact on EU citizens' fundamental rights. Tech. rep. URL: http://www.europarl.europa.eu/thinktank/en/document.html?reference=IPOL-LIBE_NT(2013)474405.

[37] Joyce R. (accessed 9.7.2016) Disrupting nation state hackers. Tech. rep. URL: https://www.usenix.org/conference/enigma2016/conference-program/presentation/joyce.

[38] Spinellis D. (2012) Don't install software by hand. IEEE Software 29, pp. 86–87. URL: http://www.spinellis.gr/pubs/jrnl/2005-IEEESW-TotT/html/v29n4.html.

[39] HashiCorp (accessed 1.2.2017.), Hashicorp: Introduction to packer. URL: https://www.packer.io/intro/.

[40] HashiCorp (accessed 1.2.2017.), Hashicorp: Introduction to terraform. URL: https://www.terraform.io/intro/.

[41] (accessed 5.2.2017), Intelligent platform management interface specification, second generation. URL: https://www-ssl.intel.com/content/dam/www/public/us/en/documents/specification-updates/ipmi-intelligent-platform-mgt-interface-spec-2nd-gen-v2-0-spec-update.pdf.

[42] Mäkinen T. (accessed 4.6.2016), boot.foo.sh installation automation. URL: http://boot.foo.sh/.

[43] (accessed 4.9.2016), iPXE - open source boot firmware. URL: http://ipxe.org.

[44] (accessed 4.9.2016), Docker is the software containerization platform. URL: https://www.docker.com/.

[45] Soltesz S., Pötzl H., Fiuczynski M.E., Bavier A. & Peterson L. (2007) Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. SIGOPS Oper. Syst. Rev. 41, pp. 275–287. URL: http://doi.acm.org/10.1145/1272998.1273025, uRL: https://pdfs.semanticscholar.org/bf36/1962667e2cf7d8c2fde3186012cc8df87cb2.pdf.

[46] Gartner I. (accessed 11.2.2017), Vendor rating: Red hat. URL: https://www.gartner.com/doc/reprints?ct=150106&id=1-26VHVSW.

[47] (accessed 24.8.2016), CentOS GPG keys - how CentOS uses GPG keys. URL: https://www.centos.org/keys/.