



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
January 2017

ABSTRACT

- **FIXME:** remove this list
- **Background information** (present tense)
- **Principal activity** (past tense/present perfect tense)
- **Methodology** (past tense)
- **Results** (past tense)
- **Conclusions** (present tense/tentative verbs/modal auxiliaries)

Network based automatic installation of operating system is and has been a crucial method to manage masses of computers. Protecting every step of the installation is important to be able to trust that the network installation system completed the installation so that the operating system and configuration is what the user wanted.

This thesis reviews history and current state of network based automatic installation systems based on what protocols do they use. Then it continues to identify risks to those protocols and see how the situation could be improved by using cryptography using Transport Layer Security (TLS) protocol and digital signatures.

Proof of concept implementation using TLS and digital signatures is compared to two other installation systems.

Keywords: sample, keywords

Herrala O. (2017) FIXME: Turvallinen käyttöönotto haastavissa ympäristöissä.
Oulun yliopisto, sähkötekniikan tutkinto-ohjelma. Kandidaatintyö, 26 s.

TIIVISTELMÄ

FIXME Esimerkki tiivistelmä

Avainsanat: esimerkki, sanoja

TABLE OF CONTENTS

ABSTRACT

TIIVISTELMÄ

TABLE OF CONTENTS

FOREWORD

ABBREVIATIONS

| | |
|--|-----------|
| 1. INTRODUCTION | 6 |
| 1.1. Information Security | 6 |
| 1.1.1. Information assurance | 7 |
| 1.1.2. This thesis in Information Security landscape | 7 |
| 1.2. Involved Protocols | 7 |
| 1.3. Risks | 8 |
| 1.4. Current state | 9 |
| 1.5. Challenging environments | 10 |
| 1.6. Mitigation | 10 |
| 1.7. Comparison to virtualization and cloud | 11 |
| 2. IMPLEMENTING SECUDEP | 12 |
| 2.1. Tools | 12 |
| 2.2. Setting up installation system | 12 |
| 2.3. Deploying | 13 |
| 2.4. Security | 13 |
| 3. CASE STUDIES | 14 |
| 3.1. Case Study 1: Identify Protocols | 14 |
| 3.1.1. What was studied | 14 |
| 3.1.2. How it was done | 14 |
| 3.1.3. Results found | 15 |
| 3.1.4. Analysis of results | 16 |
| 3.2. Case Study 2: Comparing boot.foo.sh and secudep | 17 |
| 3.2.1. What was studied | 17 |
| 3.2.2. How it was done | 17 |
| 3.2.3. Results found | 17 |
| 3.2.4. Analysis of results | 18 |
| 3.3. Case Study 3: Testing attacks against secudep | 18 |
| 3.3.1. What was studied | 18 |
| 3.3.2. How it was done | 18 |
| 3.3.3. Results found | 19 |
| 3.3.4. Analysis of results | 20 |
| 4. DISCUSSION AND CONCLUSION | 22 |

FOREWORD

FIXME FOREWORD

This L^AT_EX-template has been used by various people at department since the late 1990's, and has slowly improved over time. It is still somewhat rough at the edges, but hopefully will be helpful in reducing some of the pain involved in writing a diploma thesis.

Contributors to the template include Mika Korhonen (original author), Pekka Pietikäinen, Christian Wieser and Teemu Tokola. If you make any improvements to this template, please contact ouspg@ee.oulu.fi, and we will try to include them in further revisions.

The template was updated during the summer of 2013 by Juha Kylmänen.

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

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

Many Linux distributions offer “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 then finally a fresh operating system installation is completed on the computer.

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

The work done is a practical hands on study of how installation systems work and how to improve the security of these systems.

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 organization from management down to each individual person. Security management contains subjects like organization’s security principles, security management tasks and responsibilities, risk management, continuity and personnel security.

Physical security as the name implies 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 has information security requirements for electronic information.

1.1.1. Information assurance

Katakri's Information assurance requirements are further 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 places to places.

Operations security (FIXME: vulnerability management, backups, remote use and management, change management)

1.1.2. This thesis in Information Security landscape

Security of installation infrastructure can be put under information assurance section of Katakri. Further, it can be categorized under operations security. In operations security, installation infrastructure's role is the very beginnings of information lifecycle management. Secure installation infrastructure takes care of setting up appropriate and properly secured operating system installation to computer which then can be trusted.

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

1.2. Involved Protocols

Multiple network protocols have been developed and used to allow booting using IP network. Early published standards include RARP ("A Reverse Address Resolution Protocol", RFC903, published 1984 [4]) and BOOTP ("Bootstrap Protocol", RFC951, published 1985 [5]) which could be used to allow "a diskless client machine to discover its own IP address" [5], TFTP ("Trivial File Transfer Protocol", RFC783, published 1981 [6]) "may be used to move files between machines on different networks implementing UDP" [6].

Later developments include RARP and BOOTP to be superseded by DHCP ("Dynamic Host Configuration Protocol", RFC1531, published 1993 [7]) and TFTP superseded by NFS ("Network File System", RFC1094, published 1989 [8]) which "provides transparent remote access to shared files across networks." [8] PXE ("Preboot Execution Environment" [9]) is specification from Intel Corporation to standardize

preboot environment for network booting. In some cases TFTP or NFS or both can be replaced with HTTP (“Hypertext Transfer Protocol” [10][11]).

1.3. Risks

FIXME: Role of human?

CIA triad [12] divides network security into three elements: confidentiality, integrity and availability [13].

Confidentiality means that the sender of the message encrypts (FIXME: What vs. how?) the content so only a receiver with correct key can decrypt the content and see the message. Confidentiality can be achieved in network security for example by using TLS protocol [14] to encrypt network traffic.

Integrity control guarantees that the message cannot be modified during transfer. It consist 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 [15][16] can provide non-repudiation. Standards such as S/MIME [17] or OpenPGP [18] can be used for digital signature format.

Authenticity ensures receiving, transmitting or both parties determine they are communicating with intended party before exchanging any confidential messages. TLS protocol provides means to verify authenticity of communicating parties.

Availability means that the systems are up and operational. Perfect security could be achieved by turning everything off, but usability would be zero. It’s 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.

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 the issues with availability. For example, if the network or one or more components are not available, the whole stack of components is inoperative.

| 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 |

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

FIXME: how secudep handles the risks (in implementation chapter)

DHCP and DNS protocols could be used to redirect (“hijack”) future communications into malicious services [19][20]. DHCP is commonly used to assign IP address

to client and give various information (TFTP server's IP address, DNS servers' IP addresses). Malicious DHCP could take over following TFTP and DNS communications. DNS has many uses, but commonly it's used to translate host name into IP address. Malicious DNS server could redirect future communications into malicious services.

TFTP, NFS and HTTP protocols could be used to deliver malicious files which when executed in target system compromise the operating system installation or even infect the hardware the operation was performed in.

There has been development to secure DHCP and DNS. That however requires the network in question to be configured to take these security measurements in action. But the risks can be detected by other components (e.g. using TLS's server authentication, and digital signatures) so there's no need to changes to network configuration. Thus the installation can be done securely in any network and if something malicious is detected the installation process can halted.

Hardware (e.g. physical server or laptop) and peripherals (e.g. displays, keyboards, mice, removable medias) can have backdoored firmware [21]. The backdoors could have been installed already on factory or firmware was infected with some malware previously ran on the machine. Mitigations for risks against hardware is out of scope of this work.

1.4. Current state

FIXME: historical development (what was before, what's now?)

Software deployment technologies [22], securing virtual machines [23] as well as cloud computing security challenges [24][25] have been widely studied. However, network installation of operating system is still much the same as in the 1980s and it's the base for any software deployment or virtual machine installation to get operating system installed securely.

Alpine Linux's PXE Boot HOWTO [26] 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's "for power users who appreciate security, simplicity and resource efficiency" [27]. Similar setup is required for other Linux distributions like Red Hat [28], and for Microsoft Windows [29].

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. (FIXME: Historical development? Not much here, I'm afraid. Things have not changed since '80s)

TFTP, NFS and HTTP protocols could be replaced with HTTPS (HTTP over TLS) where TLS (Transport Layer Security Protocol [14]) 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's difficult to replace so its shortcomings need to be countered with other protocols.

Also, DNS (Domain Name System, RFC1035 [30]) is a vital protocol to the internet. It provides translation from name to IP addresses and back (and other name services). DNS is also de facto standard. Work to protect DNS traffic has been done (DNSSEC, RFC4035 [31]) and DNSSEC is slowly getting a foothold to protect DNS communications. The risk for DNS in installation systems is MitM attack. Without DNSSEC it's possible to continue using DNS and use other means outside of DNS protocol to verify DNS is working as it should. (NOTE: No relevance to DDoS, MitM is the problem)

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

1.5. Challenging environments

Computer networks are not safe nor secure [32]. Internet being the most unsafe of networks. It requires only one compromised device in a network to make the whole network unsafe. Connections in the internet do not see 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 someone might be listening or even altering the connection to ones own agendas. It might be governmental body (like NSA's PRISM program [33]), criminal organization who have gained foothold on point of network or simply curious individual just being able to do so.

Same problems can also be present in networks like corporate intranets where both government and criminal organizations might have gained foothold to operate. In USENIX Enigma 2016 conference Rob Joyce, Chief of Tailored Access Operations in National Security Agency [34] describes how his team infiltrates networks and moves there laterally to gain what they are after. Therefore intranets should be treated with same level of mistrust as internet.

1.6. Mitigation

FIXME: Role of humans?

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

Boot environment is 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 operating system kernel and other files safely over network.

Network communication is done using HTTPS with X.509 certificate pinning. This authenticates the remote server and makes it harder to MitM attack the connection. If secure channel can't be established, the boot process should be halted.

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

1.7. Comparison to virtualization and cloud

Installing operating system to virtual machine (in a cloud or other virtualization platform) enjoys many benefits compared to installation to a physical hardware. Virtualization gives easier “programmable” access to every state of virtual machine installation from setting up the machine itself and it’s 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 [35].

Infrastructure as Code can be achieved for example with tools like Packer [36] which can be used to build machine images and Terraform [37] to set up virtual machines and launch machine images to produce running virtual machines. Both tools use simple description language where operations can be specified and then ran using the tool itself.

Physical computer however usually require physical access to be able to turn it on, plug in devices and cables, and to control the first stages of startup before operating system is running. There are remote management solutions like Intelligent Platform Management Interface (IPMI) which make it possible to remotely control physical hardware. However, using IPMI still requires the physical connections to be made.

Intelligent Platform Management Interface (IPMI) is a specification of interfaces for monitoring and controlling physical computer hardware remotely via network. Using IPMI it’s possible to command computer to turn on or off, and to control BIOS settings [38].

FIXME: More ..

2. IMPLEMENTING SECUDEP

To see if it's possible to use HTTPS and digital signatures, a simple implementation of installation system called *secudep* was implemented. Source code can be found from *secudep*'s project site on GitHub [39].

Secudep's implementation has three main design principles: ease of use, ease of deploy and security. Deploying 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 developing new use cases and applications on top of already existing system. With the implemented solution there should be no need to have monolithic and centralized installation infrastructure, but designs can shift more towards personal or per application installation infrastructure.

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

Security is more difficult design principle to tackle. For the installation infrastructure the concentration should be on selecting safe defaults and guide user to make safe choices.

This implementation borrows lots of ideas and lessons learned from `boot.foo.sh`[40] and from installation infrastructure used by Faculty of Information Technology and Electrical Engineering in University of Oulu.

2.1. Tools

Secudep uses iPXE [41] as a network boot firmware. iPXE is PXE [9] implementation with additional features such as support for booting via HTTP [11] protocol. Support for HTTPS can also be compiled in. iPXE binary build is done inside container using Docker [42] software containerization platform to achieve repeatable builds with managed dependencies.

Docker is tool to easily build operating system level virtualization [43] containers. Instead of virtualizing the hardware like Xen or KVM, containers use operating system's namespaces to separate containerized applications from each other. Docker is not mandatory for achieving build however.

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

2.2. Setting up installation system

Setting up installation system using *secudep* has the following steps:

1. Generate digital signing keys
2. Collect HTTP servers' X.509 certificates for public key pinning (FIXME: explain why, how)

3. Build iPXE bootable media
4. Write configuration file
5. Generate contents for deployment

Future work on secudep should simplify these steps even further. Digital signing (NOTE CWi: this is correct wording) 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 installation system to operate (from server side) are generated under one directory. This directory can then be published on HTTPS server. The URL for the installation system files is configured in secudep's configuration file.

2.4. Security

Secudep make 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. File is trusted only after it's signature is verified successfully. This requirement can't be turned off once it's turned on.

3. CASE STUDIES

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

The first case study looks into an already existing installation infrastructure 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 already existing installation infrastructure differs from secudep.

Third and last case study looks into secudep's promise to make installation infrastructure more secure. This is done by simulating the 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 online installation infrastructure system. This study also verifies the involved protocols which were described in the introduction chapter.

For installation infrastructure a service called boot.foo.sh [40] is used. Boot.foo.sh was chosen because it's open service known to be used to automatic installations in enterprises and it has been an inspiration to this thesis to make installation infrastructure more safer to use.

Boot.foo.sh is used to install CentOS 7 Linux operating system. CentOS Linux is community driven effort to provide free alternative to Red Hat Enterprise Linux (RHEL). CentOS is built using RHEL source code. Red Hat has 67 % market share of Linux distribution market according to Gartner's analysis [44].

3.1.2. *How it was done*

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

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

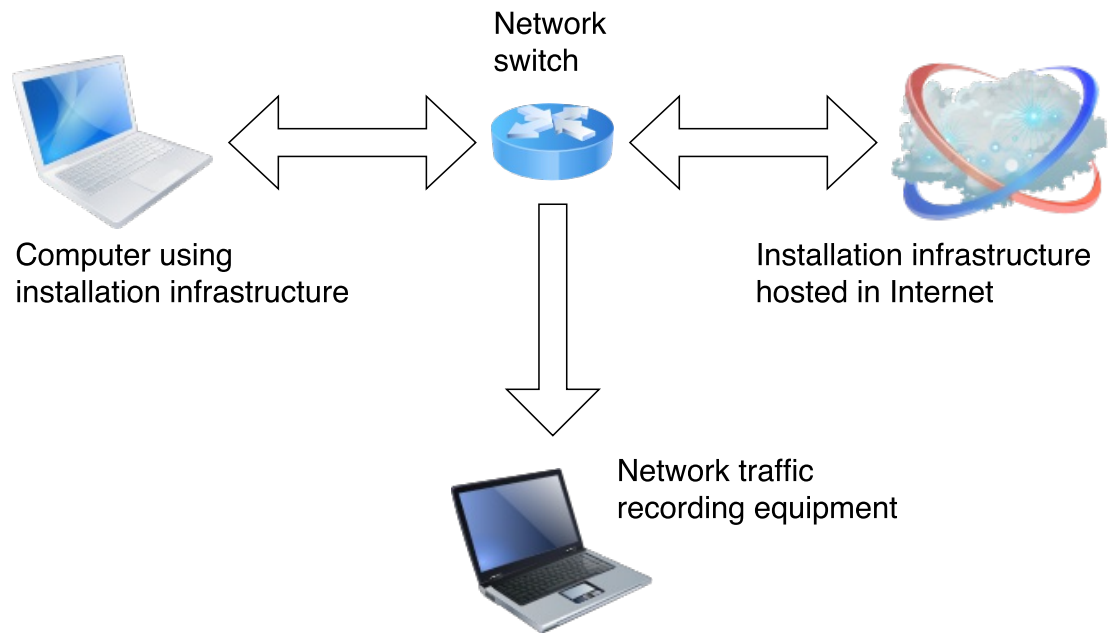


Figure 1. Network traffic recording setup.

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

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

3.1.3. Results found

Traffic recording was 788 megabytes of network traffic containing over 883 thousand network packets. Recording contains time span of a bit over nine minutes.

| Step | protocol |
|--------------------|-----------|
| Address resolution | DHCP |
| Name resolution | DNS |
| Boot menu | HTTPS |
| kernel and initrd | HTTP |
| Kickstart | HTTP |
| Installation files | HTTP (DS) |

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

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

Steps are identified and named for each system used to achieve the installation. 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 it’s purpose is to get IP address and DNS server addresses for system to be installed. DHCP is the standard protocol for this, and was also found to be used here.

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

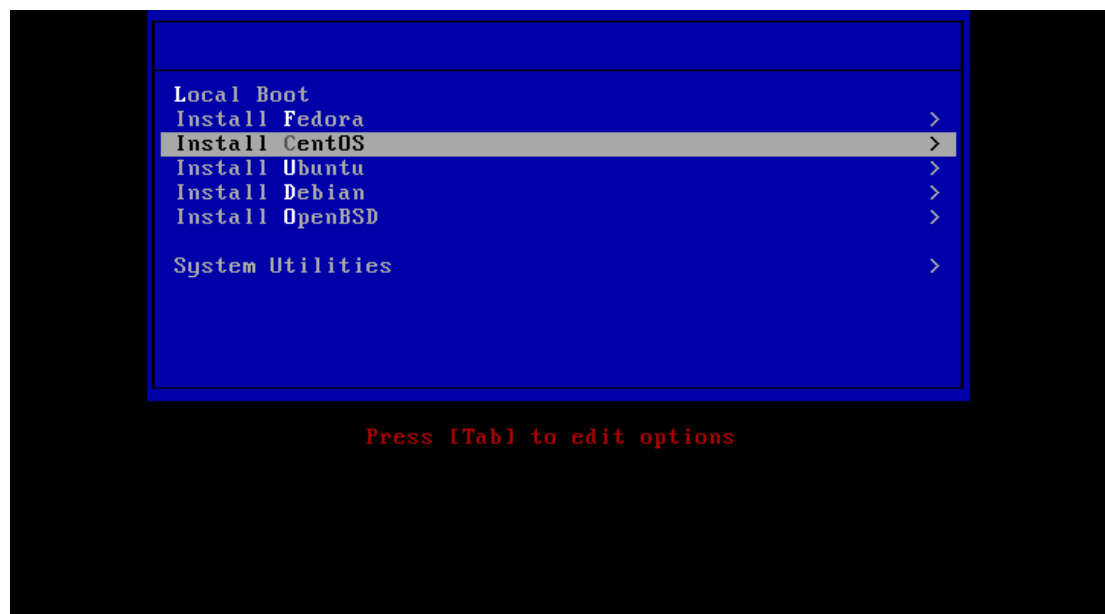


Figure 2. boot.foo.sh boot menu showing selection of operating systems.

“Boot menu” is used to display choices of operating systems to be installed. 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 and then kernel is executed and it continues the boot process. HTTP was used to communicate with CentOS 7 mirror to fetch the needed files.

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

“Installation files” are the contents of operating system to be installed. The files are downloaded and extracted to hard drive to achieve the installation. Operating system installer is usually trusted to verify digital signatures (e.g. CentOS uses OpenPGP [18]

(“GPG”) signatures) the downloaded content before extracting the files into hard drive. The CentOS documentation [45] 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, when initrd file is downloaded over insecure protocol or file content is not verified against signature it’s possible for malicious third party to inject it’s own OpenPGP keys into initrd and point installation system to malicious host serving the operating system installation files and thus gain full control of the installed system.

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 already existing installation infrastructure solution which was studied in case study 1. The purpose of this is to see the differences between used network protocols between these two systems.

3.2.2. How it was done

The results from case study 1 were used as a base and then implementation details about secudep were added and compared.

3.2.3. Results found

| 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) |

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

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 in Table 3 are discussed next.

3.2.4. Analysis of results

”Boot menu” is used to display choices of operating systems to be installed. HTTP is used in boot.foo.sh. HTTP protocol is susceptible to Man in the Middle attack. Secudep uses HTTPS (HTTP over TLS) with signed files to redemiate this issue.

Only secudep uses digital signatures and the signature files 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 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 handled to kernel. This means that secudep can’t provide digital signatures to any following files.

3.3. Case Study 3: Testing attacks against secudep

3.3.1. What was studied

This case study consist of attacks against implementation of secudep. Secudep’s main defence against attacks is the use of encryption (TLS) and digital signatures.

Table 3 on page 17 contains list of protocols involved in operating system installation process.

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

After boot menu step is done, secudep loads kernel and initrd over unsecured HTTP connection. For example Man-in-the-Middle attack could change kernel or initrd to another files, but digital signature verification should notice that.

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 HTTPS connection while boot progresses.

Testing that these verifications works can be done by either omitting the public key from Secudep boot media or serving wrong public key from Secudep’s HTTPS server.

Table 4 lists steps in boot process and what verification is used in each step.

| Attack against | Protocol | Defence |
|--------------------|------------|--|
| 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 |

Table 4. Attack and it's defence in various steps of installation. DS in table means Digital Signatures.

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.

| File | Used for | Where it's 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. Different files used for verification in various steps, where they are used and where they are located. DS in table means Digital Signatures.

Table 5 lists all files used for various steps in process. Any failure in verification should halt the installation process.

3.3.3. Results found

| 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 |

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

Five different tests were made and results can be found from 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 run the installation. The result was observed and material collected.

```
ISOLINUX 6.03 2014-10-06 ETCD Copyright (C) 1994-2014 H. Peter Anvin et al
iPXE ISO boot image
Loading ipxe.krn... ok
iPXE initialising devices...ok

iPXE 1.0.0+ (4775) -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS HTTP HTTPS iSCSI SRP AoE ELF MBOOT PXE bzImage Menu PXEXT
Configuring (net0 08:00:27:6f:16:b8)..... ok

https://raw.githubusercontent.com/ouspg/secudep/master/boot/start.ipxe... ok
https://raw.githubusercontent.com/ouspg/secudep/master/boot/start.ipxe.sig... ok

Could not verify: Permission denied (http://ipxe.org/0216eb3c)
FATAL: INT18: BOOT FAILURE
```

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

Two distinct iPXE error codes were found while conducting the test. One example of such error is shown in figure 3. As shown in the figure, the installation process was halted and couldn't proceed.

The first one, "0216eb3c" is documented¹ in 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 documented² in 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 wrong RSA signature was given in test or the file was changed so that the RSA signature verification should fail.

4. DISCUSSION AND CONCLUSION

- FIXME: remove this list
- CONCLUSIONS: reference to purpose of study
- CONCLUSIONS: value of / reasons for the study
- CONCLUSIONS: review of important findings / conclusions
- CONCLUSIONS: comments, explanations or speculations about findings
- CONCLUSIONS: limitations of study
- CONCLUSIONS: implications of study or generalizations
- CONCLUSIONS: recommendations for future or practical applications - USUALLY SKIPPED

This thesis took a look what network based risks could face installation infrastructure and then studied what kind of means could be used to protect the initial phases (before operating system kernel took the control of execution) of installation process using encryption and digital signatures.

Protecting every step of communications over networks is important and protecting installation infrastructure is no exception. This thesis has shown that it's possible to take a step further in a more secure installation infrastructure by using two technologies: encryption and digital signatures.

More secure systems can be build step by step by combining simple individual components without the need for designing a whole new systems and technologies. Replacing old components (like TFTP, NFS and HTTP protocols) with new ones (HTTPS protocol) and increasing the use of digital signatures is a small steps to take for big benefits in security. Also when using HTTPS it's possible to use HTTP's authentication schemes to hide installation scripts (kickstart files, etc.) which otherwise would be visible to 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 meta data with digital signatures, but some distributions have that functionality turned off by default. Maybe mirrors at some point could take step forward and enable HTTPS so files like kernel and initrd, and package database meta data could be securely downloaded?

The public key to verify digital signatures is also embedded into initrd file. 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 the iPXE and it's 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 gppe 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] Finlayson, Mann, Mogul & Theimer (accessed 26.5.2016) RFC903: A reverse address resolution protocol. Tech. rep. URL: <https://tools.ietf.org/html/rfc903>.
- [5] Croft B. & Gilmore J. (accessed 26.5.2016) RFC951: Bootstrap protocol (BOOTP). Tech. rep. URL: <https://tools.ietf.org/html/rfc951>.
- [6] Sollins K.R. (accessed 26.5.2016) RFC783: The TFTP protocol (revision 2). Tech. rep. URL: <https://tools.ietf.org/html/rfc783>.
- [7] Droms R. (accessed 26.5.2016) RFC1531: Dynamic host configuration protocol. Tech. rep. URL: <https://tools.ietf.org/html/rfc1531>.
- [8] Nowicki B. (accessed 26.5.2016) RFC1094: NFS: Network file system protocol specification. Tech. rep. URL: <https://tools.ietf.org/html/rfc1094>.
- [9] Berners-Lee T., Fielding R. & Frystyk H. (accessed 18.6.2016) Pre-boot execution environment (pxe) specification version 2.1. Tech. rep. URL: <ftp://download.intel.com/design/archives/wfm/downloads/pxespec.pdf>.
- [10] 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>.
- [11] 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>.
- [12] Perrin C. The CIA triad. Tech. rep. URL: <http://www.techrepublic.com/blog/it-security/the-cia-triad/>.
- [13] Anderson R.J. (2008) Security Engineering, 2nd edition. URL: <https://www.cl.cam.ac.uk/~rja14/book.html>.
- [14] 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>.

- [15] 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>.
- [16] 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>.
- [17] 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>.
- [18] 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>.
- [19] 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>.
- [20] 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>.
- [21] 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. 1–13 URL: https://www.hgi.rub.de/media/crypto/veroeffentlichungen/2016/05/12/hardware_trojan_fpga.pdf.
- [22] 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/.
- [23] 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>.
- [24] Owens D. (2010) Securing elasticity in the cloud. *Commun. ACM* 53, pp. 46–51. URL: <http://doi.acm.org/10.1145/1743546.1743565>.
- [25] 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>.
- [26] (accessed 26.5.2016), PXE boot howto - Alpine Linux. URL: https://wiki.alpinelinux.org/wiki/PXE_boot.

- [27] Team A.L.D. (accessed 26.5.2016), Alpine linux - about. URL: <https://alpinelinux.org/about>.
- [28] 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.
- [29] 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>.
- [30] Mockapetris P. (accessed 25.8.2016) RFC1035: Domain names - implementation and specification. Tech. rep. URL: <https://tools.ietf.org/html/rfc1094>.
- [31] 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>.
- [32] Ward R. & Beyer B. (2014) Beyondcorp: A new approach to enterprise security. ;login: Vol. 39, No. 6, pp. 6–11.
- [33] 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](http://www.europarl.europa.eu/thinktank/en/document.html?reference=IPOL-LIBE_NT(2013)474405).
- [34] Joyce R. (accessed 9.7.2016) Disrupting nation state hackers. Tech. rep. URL: <https://www.usenix.org/conference/enigma2016/conference-program/presentation/joyce>.
- [35] 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>.
- [36] HashiCorp (accessed 1.2.2017.), Hasicorp: Introduction to packer. URL: <https://www.packer.io/intro/>.
- [37] HashiCorp (accessed 1.2.2017.), Hasicorp: Introduction to terraform. URL: <https://www.terraform.io/intro/>.
- [38] (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>.

- [39] (accessed 26.5.2016), Secure deployment for challenging environments. URL: <https://github.com/ouspg/secudep>.
- [40] Mäkinen T. (accessed 4.6.2016), boot.foo.sh installation automation. URL: <http://boot.foo.sh/>.
- [41] (accessed 4.9.2016), iPXE - open source boot firmware. URL: <http://ipxe.org>.
- [42] (accessed 4.9.2016), Docker is the software containerization platform. URL: <https://www.docker.com/>.
- [43] 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>.
- [44] Gartner I. (accessed 11.2.2017), Vendor rating: Red hat. URL: <https://www.gartner.com/doc/reprints?ct=150106&id=1-26VHVSW>.
- [45] (accessed 24.8.2016), CentOS GPG keys - how CentOS uses GPG keys. URL: <https://www.centos.org/keys/>.