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
FIXME Month 2016

Herrala O. (2016) Secure Deployment Story for Challenging Environments. University of Oulu, Degree Programme in Electrical Engineering. Bachelor's thesis, 15 p.

ABSTRACT

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

Keywords: sample, keywords

Herrala O. (2016) FIXME: Turvallinen käyttöönotto haastavissa ympäristöissä.
Oulun yliopisto, sähkötekniikan tutkinto-ohjelma. Kandidaatintyö, 15 s.

TIIVISTELMÄ

Esimerkkitiivistelmä

Avainsanat: esimerkki, sanoja

TABLE OF CONTENTS

FOREWORD

FIXME FOREWORD

This LaTeX-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)
FTP	File Transfer Protocol (IETF)
HTTP	Hypertext Transfer Protocol (IETF)
IETF	Internet Engineering Task Force
IP	Internet protocol (IETF)
MitM	Man In The Middle
NFS	Network File System (IETF)
PXE	Preboot Execution Environment
RARP	A Reverse Address Resolution Protocol (IETF)
RFC	Request for Comments
TFTP	Trivial File Transfer Protocol (IETF)
TLS	Transport Layer Security (IETF)
UDP	User Datagram Protocol (IETF)

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.

Usually network installation systems are built to serve single organization (e.g. single business) inside their own networks (“intranets”) to achieve repeatable and homogeneous installations. Installation preferably should be as easy and as fast as possible.

Many Linux distributions offer “net install” where a small image is used to boot the computer into a state where rest of the installation software and packages can be downloaded directly from Internet.

This thesis looks at history of installation systems, then identifies what network based threats there are and studies how to protect the installation process using readily available tools and components.

After introduction, in next chapter discusses about a proof of concept implementation of network installation system and it’s design principles.

Then before discussion and conclusion, the proof of concept implementation is compared against to two other installation systems.

1.1. Protocols

Multiple protocols have been developed and used in combination to allow booting using IP network. Early published standards include RARP (“A Reverse Address Resolution Protocol”, RFC903, published 1984 [1]) and BOOTP (“Bootstrap Protocol”, RFC951, published 1985 [2]) could be used to allow “a diskless client machine to discover its own IP address” [2], TFTP (“Trivial File Transfer Protocol”, RFC783, published 1981 [3]) “may be used to move files between machines on different networks implementing UDP.” [3].

Later developments include RARP and BOOTP to be superseded by DHCP (“Dynamic Host Configuration Protocol”, RFC1531, published 1993 [4]) and TFTP superseded by NFS (“Network File System”, RFC1094, published 1989 [5]) which “provides transparent remote access to shared files across networks.” [5] PXE (“Preboot Execution Environment” [6]) is specification from Intel Corporation to standardize preboot environment for network booting.

1.2. Current state

Alpine Linux’s PXE Boot HOWTO [7] 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:
>
> - Set up a DHCP server and configure it to support PXE boot.
> - Set up a TFTP server to serve the PXE boot loader.

- Set up an HTTP server to serve the rest of the boot files.
- Set up an NFS server from which Alpine can load kernel modules.
- Configure mkinitfs to generate a PXE-bootable initrd.

As we can see, 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 can be replaced with HTTPS (HTTP over TLS) where TLS protocol provides communications security using cryptography and authentication of one or both communicating parties.

DHCP is de facto standard to achieve zero configuration and it's difficult to replace so it's shortcomings need to be countered with other protocols.

1.3. Challenging environments

Computer networks are not safe nor secure. Internet being the most unsafe of networks. 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 [8]), 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, etc. 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 [9] describes how his team infiltrates networks and moves there laterally to gain what they are after. Because of this intranets should be treated with same level of mistrust as internet.

1.4. Threats

CIA triad divides network security into three elements: confidentiality, integrity and availability.

Confidentiality means sender of the message encrypts the content so only receiver with correct key can decrypt the content and see the message.

Integrity control guarantees that 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.

Authenticity ensures receiving, transmitting or both parties determine they are communicating with intended party before exchanging any confidential messages.

Availability means that the systems are up and operational.

Threats 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.

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

Component	Role	Threat(s)
HTTP	File transfer	confidentiality, integrity
DNS	Name service	non-repudiation
NFS	File transfer	confidentiality, integrity
TFTP	File transfer	confidentiality, integrity
DHCP	Zero configuration	non-repudiation

DHCP and DNS protocols could be used to redirect ("hijack") future communications into malicious services. 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 future 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 threats can be detected by other components (e.g. using TLS's server authentication, and file 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. The backdoors could have been installed already on factory or firmware was infected with some malware previously ran on the machine. Discussing mitigations for threats against hardware is out of scope of this work.

1.5. Mitigation

Threats 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 opened, 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.

2. IMPLEMENTING SECUDEP

Implementation has three main design principles: ease of use, ease of deploy and security. Deploying new installation infrastructure 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[10] and from installation infrastructure used by Faculty of Information Technology and Electrical Engineering in University of Oulu. These two systems are also compared to this implementation in next chapter.

2.1. Ease of use

Setting up installation system using secudep has the following steps:

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

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

2.2. Ease of deploy

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.

2.3. Security

Secudep tries to make it as easy as possible to use public key pinning for HTTPS hosts and file signing 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. COMPARISON BETWEEN SYSTEMS

3.1. Method

I studied three different implementations of installation infrastructure (later called "systems") by installing operating system using it and studying what protocols are used. Protocols were identified from network traffic capture using Wireshark network protocol analyzer. The operating system used for testing installation was CentOS Linux Distribution because it was available on all three systems.

First system was what I want to call "traditional" and is designed to be used inside corporate network. Second system is a more modern (boot.foo.sh [10]) designed to be used over the internet and then last one (secudep) was the implementation discussed in this thesis. The comparison analyzes protocols used to achieve the installation and doesn't go deeply into contents of the protocol messages and only cursorily looks at how protocols are used.

All three installations were done using VirtualBox virtual machines. VirtualBox allows network traffic to be captured from the beginning of virtual machine's lifetime. This allowed me to capture and study what happens with earliest stages of boot process.

3.2. Result

Step	traditional	over internet	secudep
Zero configuration	DHCP	DHCP	DHCP
Name resolution	DNS	DNS	DNS
Boot menu	TFTP	HTTP	HTTPS (FS)
File signatures	N/A	N/A	HTTPS
kernel and initrd	TFTP	HTTP	HTTP (FS)
Kickstart	NFS	HTTP	HTTPS
Installation files	NFS	HTTP	HTTP

Table 2. Comparison between how three different installation infrastructures use protocols. FS in table means File Signing.

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

I identified and named common steps each system used to achieve the installation. The steps were "zero configuration", "name resolution", "boot menu", "kernel and initrd", "kickstart" and "installation files". Secudep also has additional step "file signatures".

"Zero configuration" is the first step and it's purpose was to get IP address and DNS server addresses for system to be installed.

"Name resolution" is used to translate host names into IP address to communicate with other servers.

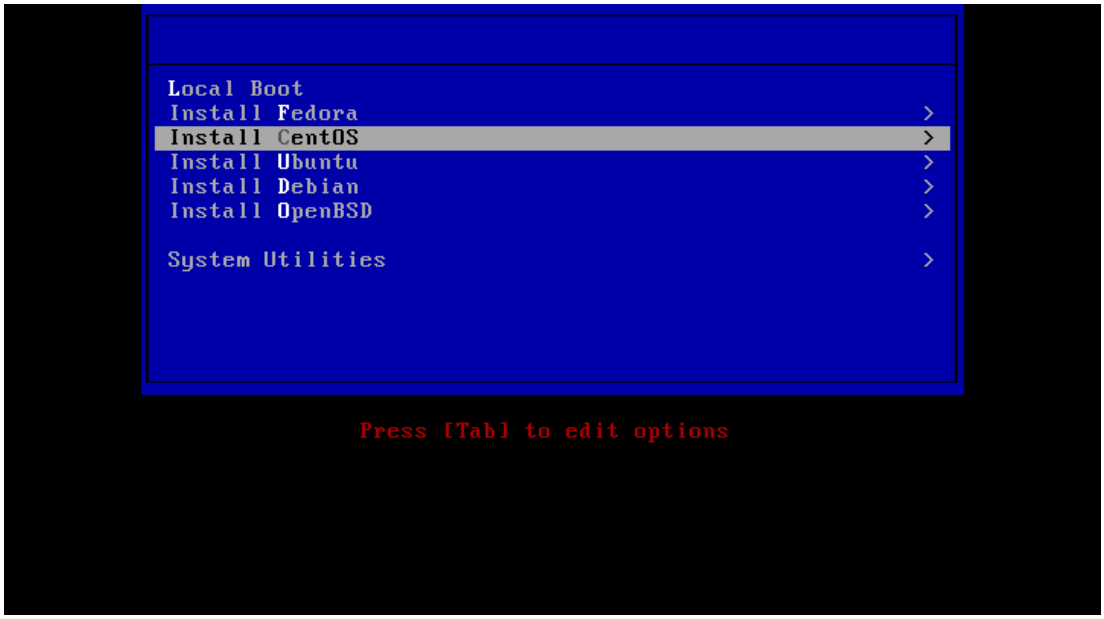

Figure 1. boot.foo.sh boot menu showing selection of operating systems.

"Boot menu" is used to display choices of operating systems to be installed. Example of boot menu can be seen in figure 1.

"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.

"Kickstart" is CentOS specific file for automating unattended installation. It's set of instructions downloaded and executed by the installation process.

"Installation files" are the contents of operating system to be installed. The files are downloaded and extracted to hard drive to achieve the installation.

"File signatures" are cryptographically calculated proofs to verify files. If the file is changed, the signature check fails and user can be alerted about the incident.

3.3. Analysis

In all three systems the same protocols are used for zero configuration (DHCP) and name resolution (DNS). DHCP and DNS are de facto standard protocols to achieve the task they solve so there's no surprise here.

Boot menu is used to display choices of operating systems to be installed. TFTP and HTTP are the protocols used in traditional and over the internet systems. Both TFTP and HTTP protocols are susceptible to Man in the Middle attack. Secudep uses HTTPS (HTTP over TLS) with signed files.

Only secudep uses file signatures and the signature files are fetched over HTTPS. This is the step missing from the traditional and over the internet installation systems.

Kernel and initrd are the files needed to launch Linux installation. Here traditional system uses TFTP to serve these files, but over internet and secudep systems use HTTP. Again TFTP and HTTP are susceptible to Man in the Middle attacks. TFTP is the de facto standard to deliver these files inside intranet systems. HTTP is used because

the files are fetched from CentOS's official mirror over the internet. Secudep uses file signing to verify downloaded content.

After kernel and initrd are downloaded and file signatures are verified the execution is handled to kernel. This means that secudep can't provide file signatures to any following files. However, there's still two important steps in installation process: kickstart file and installation files.

Kickstart is CentOS specific file for automating unattended installation. The kickstart file is downloaded by the initrd system so secudep can't do file signature verification. However, secudep uses HTTPS where traditional relies on NFS and over internet system uses HTTP. Both NFS and HTTP are suspecting to Man in the Middle attacks.

Installation files are downloaded and then installed into hard drive to achieve the operating system installation. Operating system installer is trusted to verify (e.g. CentOS uses GPG signatures) the downloaded content before extracting the files into hard drive. CentOS documentation [11] 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 website 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 GPG 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.

4. DISCUSSION AND CONCLUSION

- 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 threats could face installation infrastructure and then studied what kind of means could be used to protect the initial phases (before OS kernel took the control of excecution) of installation process using encryption and file signing.

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 file 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 techonologies. Replacing old components (like TFTP, NFS and HTTP protocols) with new ones (HTTPS protocol) and increasing the use of file signatures is a small steps to take for big benefits in security.

Linux distributions and other open source operating systems use GPG or other file signing 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 file 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 metadata could be securely downloaded?

The public key to verify file 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 iPXE and it's TLS implementation and file signing capabilities. This was intentionally left out from this thesis.

5. REFERENCES

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

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

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

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

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

[6] 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.

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

[8] 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.

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

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

[11] (accessed 24.8.2016.), Centos gpg keys - how centos uses gpg keys. URL: https://www.centos.org/keys/.