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, 14 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 auxliaries)**

Keywords: sample, keywords

Herrala O. (2016) FIXME: Turvallinen käyttöönotto haastavissa ympäristöissä. Oulun yliopisto, sähkötekniikan tutkinto-ohjelma. Kandidaatintyö, 14 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. university department or single business) to achieve repeatable and homogeneous installations. For example school's computer class wants to have identical installations on all machines and reinstalling the machines should be as simple and as fast as possible.

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

This thesis briefly identifies what network based threats there are and then studies how to protect the installation process using readily available tools to enable encryption and code signing. Proof of concept implementation of network installation system is tested to see how encryption and code signing can help secure the installation process.

1.1. Protocols

Multiple procotols 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] summarises 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 bootloader.
> - 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 are not secure and should not be used over Internet.

TFTP, NFS and HTTP protocols can be replaced with HTTPS (HTTP over TLS) where TLS protocol provides communications security using cryptocraphy and authentication of one or both communicating parties.

1.3. Threats

Tanenbaum's Computer Networks[8] divides network security threats into four categories: secrecy, authentication, nonrepudiation and integrity control. Secrecy (or data confidentiality) means sender of the message encrypts the content so only receiver with correct key can decrypt the content and see the message. Authentication ensures receiving, transimitting or both parties determine they are communicating with intented party before exchanging any confidential messages. Integrity control guarantees that message cannot be modified during transfer. Nonrepudiation ensures proof of integrity and the origin of data. This is usually achieved with using authentication and integrity control.

Threats can be indentified in all components from hardware to operating system vulnerabilities. Table 1.3 lists some common known attacks which could be targeted towards network booting or network installation infrastructure.

Component	Role	Threat(s)
HTTP	File transfer	secrecy, integrity control
DNS	Name service	nonrepudiation
NFS	File transfer	secrecy, integrity control
TFTP	File transfer	integrity control
DHCP	Zero configuration	nonrepudiation

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

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 code signing) 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.

1.4. 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 SOMETHING

Implementation has three main design principles: ease of use, ease of deploy and security. Deploying new installation infrastructure should be easy so that it encouraces building small and easy to update setups. Ease of deployment might also attract developing new uses and applications on top of already existing system. With the implemented solution there should be no need to have monolithic and centralized installation infrastucture, but things 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 automatised 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 lesson's learned from boot.foo.sh[9].

2.1. Ease of use

```
[DEFAULT]
webroot = https://raw.githubusercontent.com/ouspg/secudep/master/boot/
destdir = /Users/oherrala/ouspg/secudep/boot
signkey = /Users/oherrala/ouspg/secudep/src/codesign/codesign.key
signcert = /Users/oherrala/ouspg/secudep/src/codesign/codesign.pem

[centos7]
name = CentOS 7
kernel = http://mirror.centos.org/centos/7/os/x86_64/isolinux/vmlinuz
initrd = http://mirror.centos.org/centos/7/os/x86_64/isolinux/initrd.img
params = text utf8 inst.ks=https://raw.githubusercontent.com/ouspg/secudep/master/boot/centos7.ks
```

Table 2. Sample config file used to build installation infrastructure

2.2. Ease of deploy

2.3. Security

3. TESTING SOMETHING

3.1. Traffic analysis

TODO Capture traffic via VirtualBox PCAP dump and take a peek inside.

```
IP 0.0.0.0.bootpc > broadcasthost.bootps: BOOTP/DHCP, Request from 08:00:27:6f:16:b8 (oui Unknown), length 399
IP 10.0.2.2.bootps > 10.0.2.15.bootpc: BOOTP/DHCP, Reply, length 548
IP 0.0.0.0.bootpc > broadcasthost.bootps: BOOTP/DHCP, Request from 08:00:27:6f:16:b8 (oui Unknown), length 399
IP 10.0.2.2.bootps > 10.0.2.15.bootpc: BOOTP/DHCP, Reply, length 548
IP 0.0.0.0.bootpc > broadcasthost.bootps: BOOTP/DHCP, Request from 08:00:27:6f:16:b8 (oui Unknown), length 411
IP 10.0.2.2.bootps > 10.0.2.15.bootpc: BOOTP/DHCP, Reply, length 548
ARP, Request who-has 10.0.2.2 tell 10.0.2.15, length 28
ARP, Reply 10.0.2.2 is-at 52:54:00:12:35:02 (oui Unknown), length 28
```

Table 3. Traffic capture showing DHCP initialization

```
ARP, Request who-has 10.0.2.2 tell 10.0.2.15, length 28
ARP, Reply 10.0.2.2 is-at 52:54:00:12:35:02 (oui Unknown), length 28
```

Table 4. Client doing ARP request for learning gateway's MAC address

```
IP 10.0.2.15.56359 > 172.17.1.1.domain: 63232+ A? raw.githubusercontent.com. (43)
IP 172.17.1.1.domain > 10.0.2.15.56359: 27875 2/4/0 CNAME github.map.fastly.net., A 23.235.43.133 (177)
```

Table 5. Client doing DNS request to determine IPv4 address for raw.githubusercontent.com

```
IP 10.0.2.15.54313 > 172.17.1.1.domain: 15216+ A? mirror.centos.org. (35)
IP 172.17.1.1.domain > 10.0.2.15.54313: 15216 1/0/0 A 93.113.36.66 (51)
IP 10.0.2.15.50434 > mirrors.ch-center.com.http: Flags [S], seq 1040635034, win 65532,
    options [nop,nop,TS val 996012 ecr 0,nop,nop,sackOK,nop,wscale 9,mss 1460], length 0
IP mirrors.ch-center.com.http > 10.0.2.15.50434: Flags [S.], seq 448001, ack 1040635035,
    win 65535, options [mss 1460], length 0
IP 10.0.2.15.50434 > mirrors.ch-center.com.http: Flags [P.], seq 1:135, ack 1, win 65532,
    length 134: HTTP: GET /centos/7/os/x86_64/isolinux/vmlinuz HTTP/1.1
IP mirrors.ch-center.com.http > 10.0.2.15.50434: Flags [.], ack 135, win 65535, length 0
```

Table 6. Client doing DNS and HTTP request to receive CentOS 7 kernel

3.2. Man in the middle attacks

3.2.1. Attack against TLS

TODO Invalid certificate on remote server

3.2.2. Malicious binaries over HTTP

TODO Test what happens if Linux distribution mirror shares forged binary.

4. DISCUSSION

Your discussion.

5. 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 generalisations
- CONCLUSIONS: recommendations for future or practical applications - USUALLY SKIPPED

The best thesis evar.

6. 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] Tanenbaum A.S. (1996) Computer Networks. Prentice Hall Professional Technical Reference, 3th ed.

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