Network Boot in a Zero-Trust Environment

Stephano Cetola
Agenda

• Understanding PXE Boot
• Limitations with the Legacy BIOS Model
• Modernizing Network Boot with UEFI
• Use Cases for HTTP(S) Boot
• Frequently Asked Questions
Goals

- High Level Overview of PXE & HTTPS Boot
- Review Attack Vectors & Threat Models
- Justify Upgrades as a Business Requirement
- Room for Improvement: Community Goals
Understanding PXE Boot
Preboot eXecution Environment

- Network client is booted to a server on a local network
- Originally defined by Wired for Management (WfM, ca. 1997)
- Architecture based on 16-bit Basic Input/Output System (BIOS)
- Requires a network controller to provide a low-level driver
- Based on Universal Network Device Interface (UNDI, ca. 1988)
PXE Boot

PXE Client 1

DHCP Server

switch

PXE Client 2

PXE Boot Server
What’s wrong with this picture?
Legacy BIOS PXE Limitations

- Does Not Scale (IPv4 & UDP)
- Requires Modifications to the DHCP Server
- PXE server must be on same subnet
- Router/Switch “fast learning spanning tree” may drop UDP packets
- Cannot specify specific PXE server (first response wins)
- No Security or Authentication

<blink>No Security or Authentication</blink>
Legacy BIOS PXE Limitations

• Does Not Scale (IPv4 & UDP)
• Requires Modifications to the DHCP Server
• PXE server must be on same subnet
• Router/Switch “fast learning spanning tree” may drop UDP packets
• Cannot specify specific PXE server (first response wins)
• No Security or Authentication

“We live in an era of unprecedented cyber risk, made worse by archaic systems, dissonant silos, and a dizzying array of new security tech.”

[link]
go.forrester.com/zero-trust/zero-trust-model
How do we do better?
Modernizing Network Boot

UEFI improvements on Legacy BIOS PXE

• Networking Model
• Network Boot via PXE
• Secure Boot
• Network Boot via HTTP/HTTPS
UEFI 2.3.1+ Networking Model

- PXE BC / iSCSI (OS-boot-loader visible API’s)
- Network App
- MTFTP4
- DHCP4
- TCP4
- UDP4
- IPv4
- ICMP
- IGMP
- ARP
- MNP
- UNDI Driver (NIC UEFI Option ROM)
- NIC
- IPv6
- ND
- MLD
- ICMPv6
- IP6CONFIG
- MTFTP6
- DHCP6
- TCP6
- UDP6
- IPv6
- Network App
UEFI 2.3.1+ Network Stack

Application
Transport
Internet
Network Interface
UEFI Network Boot via PXE

• Based on old PXE Model
• Adds IPV6 & iSCSI/TCP features
• Adds Secure Boot
• Still relies on UDP / TFTP
• Not Really Secure
• <blink>Not Secure</blink>
UEFI 2.5 HTTP(S)
Boot from URL / filename

<table>
<thead>
<tr>
<th>Driver</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Boot Driver</td>
<td>HTTP Library</td>
</tr>
<tr>
<td>HTTP Driver</td>
<td>TlsLib Library</td>
</tr>
<tr>
<td>HTTP Utilities Driver</td>
<td>OpenssLib Library</td>
</tr>
<tr>
<td>TLS Driver</td>
<td></td>
</tr>
</tbody>
</table>

- Flexible Network Deployment
- Home Environment Support
- Corporate Environment Support
HTTP(S) Boot via UEFI

- Boot file URL can use DNS (i.e. www.xyz.com/bootme.efi)
- Works on any network that supports TCP & HTTP(S)
- Supports IPv4/IPv6, not limited to a single subnet
- Can take advantage of Secure Boot
- HTTPS is secure, verified via client certificates
Currently, UEFI HTTPS Boot only supports server authentication with an unauthenticated client. This requires enrolling a Server CA certificate (rootcert.pem) on the Client prior to boot. Reference: Getting Started with UEFI HTTPS Boot on EDK II
Use Cases for HTTP(S) Boot

• Install / deployment of OS
• Firmware update without an OS (see: UEFI Capsule Update)
• System recovery from LAN or cloud storage
• Diskless systems boot to OS via HTTPS (blade, thin client, ...)

[Image]
Use Cases for HTTP(S) Boot

• Install / deployment of OS
• Firmware update without an OS (see: UEFI Capsule Update)
• System recovery from LAN or cloud storage
• Diskless systems boot to OS via HTTPS (blade, thin client, ...)

Use cases are extended beyond traditional “trust boundaries”
FAQ
Does my distro support it?

- openSUSE 42.3
- SUSE SLES 15
- RHEL 7.6
  - (HTTPS in progress)
  - RAM disk for now
Does my BIOS support it?

- Please stop saying BIOS
- I’m sorry UEFI is complex
- IBVs - HP, AMI, Phoenix
Will it be hard to configure?

- No
- Instructions online
- Thank Gary Lin (SUSE)
- Also Joey Lee (SUSE)
- Ping the community: tianocore.org edk2.groups.io
- Think about where we should store our certs
✓ Upgrade from PXE
✓ Review Secure Boot
✓ Talk to your distros
✓ Think about certs
Questions?
Information on UEFI Systems with HTTPS

- Information on HTTP(S) Boot in EDK II
  - https://github.com/tianocore/tianocore.github.io/wiki/HTTP-Boot
- Virtual Environment using EDK II Open Virtual Machine Firmware (OVMF)
  - https://github.com/tianocore/tianocore.github.io/wiki/OVMF
- MinnowBoard Max Turbot (support included in firmware images)
  - https://firmware.intel.com/projects/minnowboard-max
- Hewlett Packard Enterprise* Proliant Gen10 servers
  - https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-a00016376en_us
UEFI HTTP(S) Installation Instructions

• Setup UEFI HTTPS boot in OVMF (virtual environment)
  https://en.opensuse.org/UEFI_HTTPPBoot_with_OVMF
• Setup UEFI HTTPS for Physical Host Server
  https://en.opensuse.org/UEFI_HTTPPBoot_Server_Setup
• HTTP/HTTPS Boot Getting Started Guide for EDK II