*presented by*


# Increasing risks to UEFI firmware due to growing attack surfaces

Fall 2018 UEFI Plugfest

October 15 – 19, 2018

Presented by Glenn Plant (Phoenix Technologies Ltd.)

# Legal Stuff

# Agenda

- Firmware as a target
- Spec extensions provide new attack surfaces
- OEM features add even more
- Examples of risky implementations
- Mitigation recommendations
- Suggestions for working groups
- Questions?

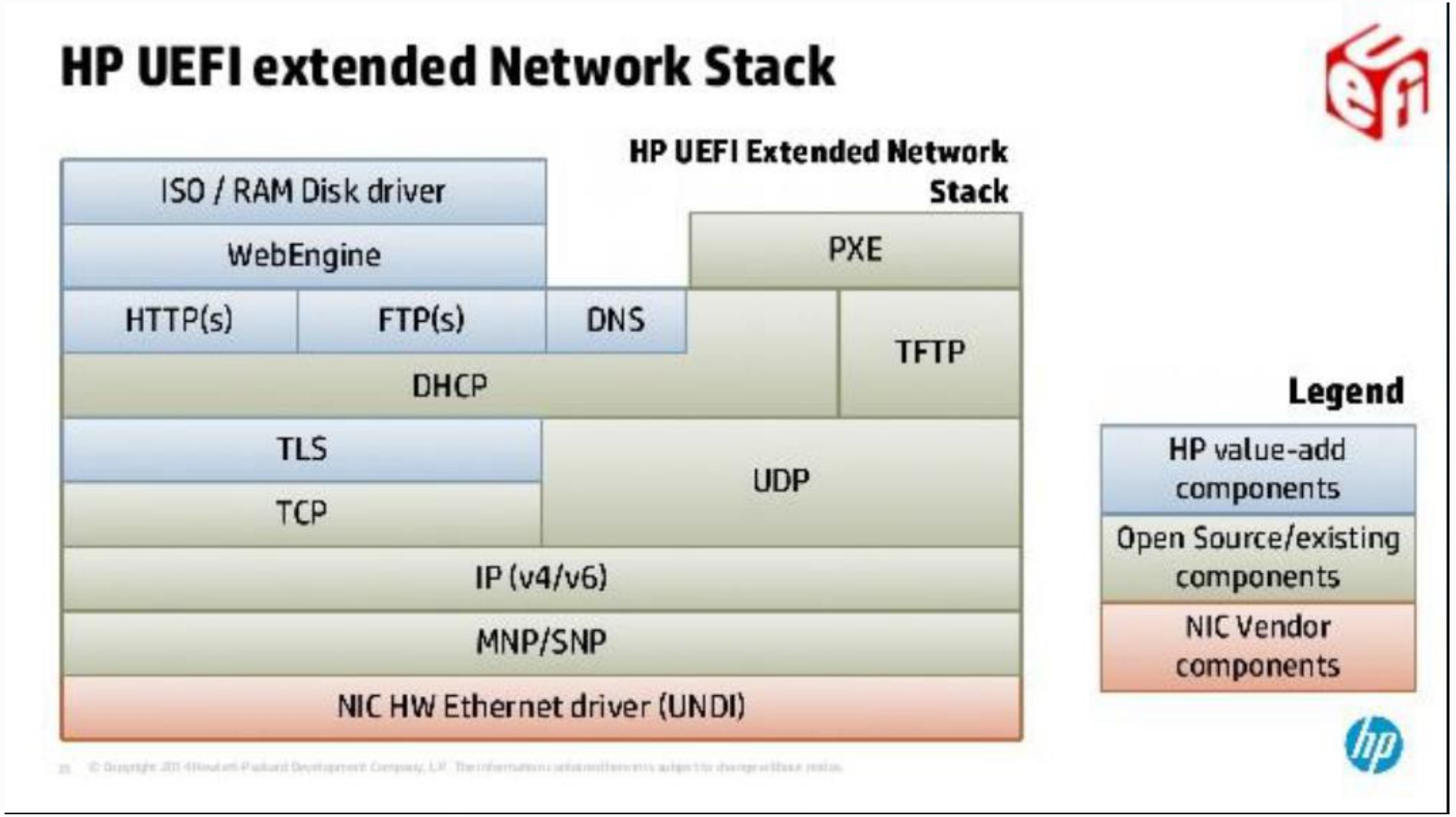
# Firmware as a target

- As OSes and apps are hardened, the bad actors move to platform firmware
- If firmware is compromised, nothing that runs later is safe
  - Malware can spoof an OS, Virtual Machines, Anti-virus, etc. Any code that runs later
  - It can be persistent, runs boot after boot
  - Wiping the system and reinstalling software may not clear it

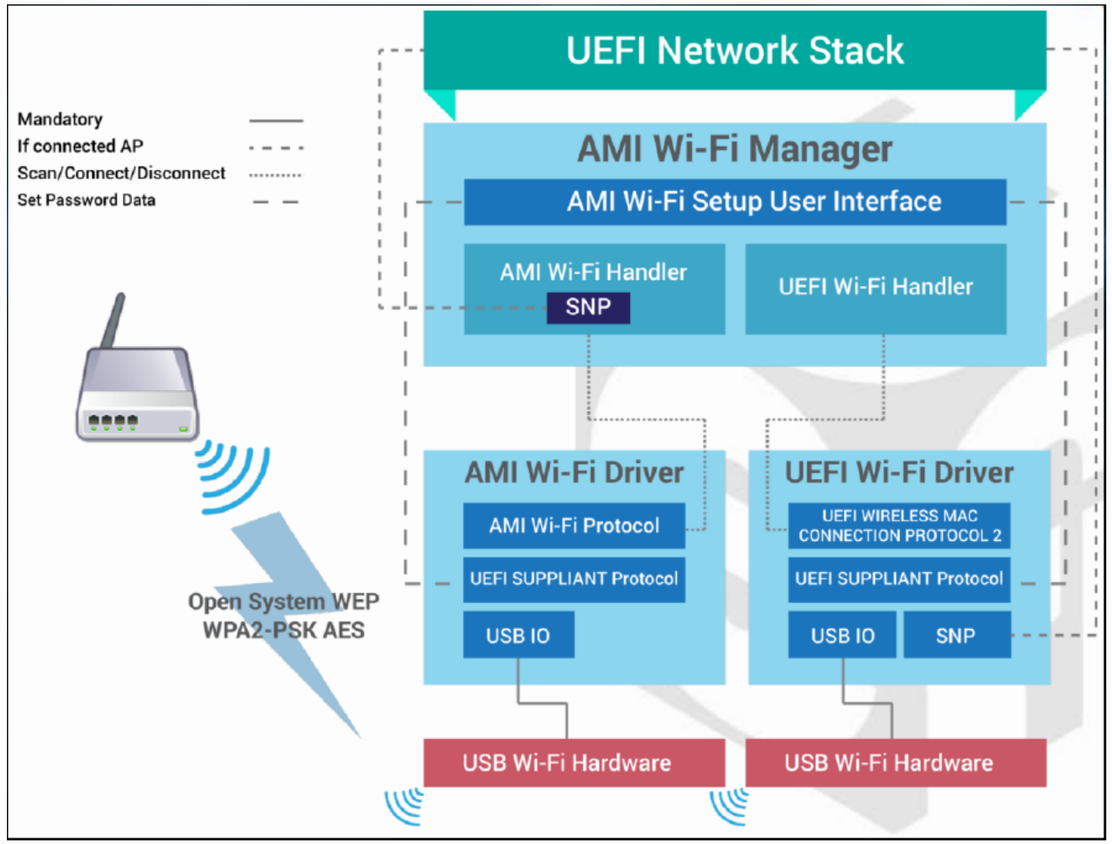
# UEFI features add attack surfaces

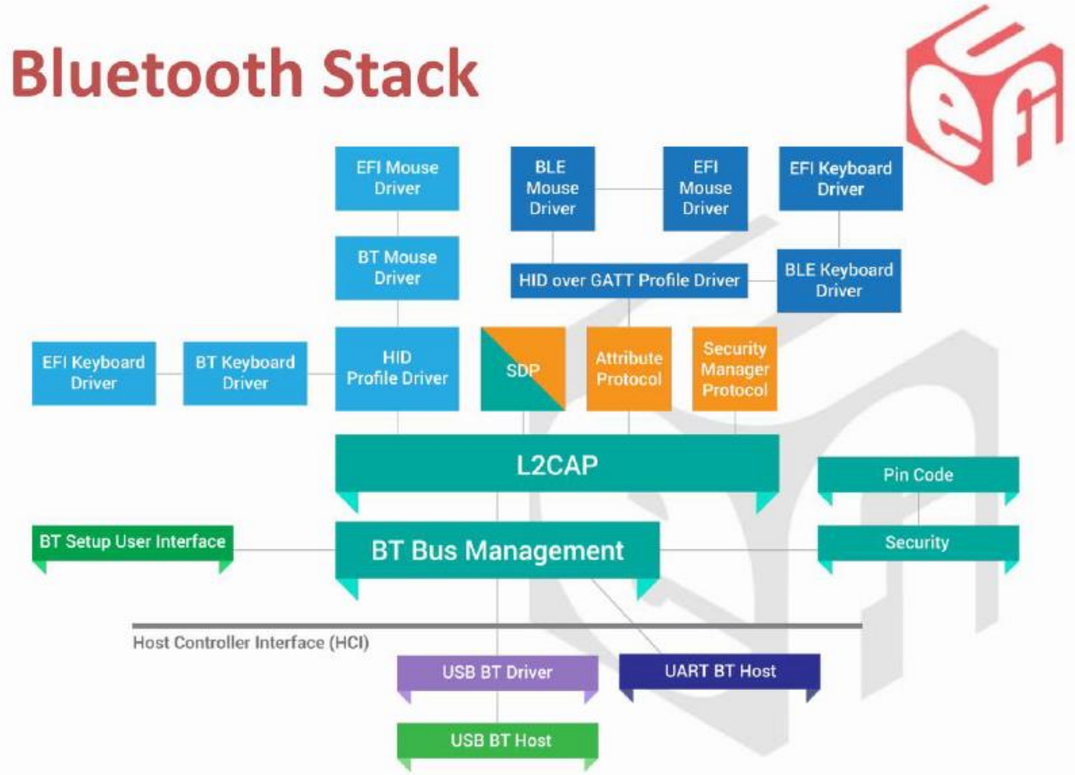
- In the past several years, UEFI Forum has added network support to the spec
  - SNP, PXE, BIS, HTTP(S) Boot, TCP/IP, UDP, IPSec, FTP, TLS, ARP, DHCP, MTFTP
  - Users have also added SNMP and others
  - Network connectivity allows for exploits that don’t require physical access to a system
- Some have added NTFS filesystem support to firmware

# An example


HP UEFI extended Network Stack
HP UEFI Extended Network Stack
ISO / RAM Disk driver
WebEngine
PXE
HTTP(s)
FTP(s)
DNS
TFTP
DHCP
TLS
UDP
TCP
IP (v4/v6)
MNP/SNP
NIC HW Ethernet driver (UNDI)
Legend
HP value-add components
Open Source/existing components
NIC Vendor components

# More examples



# Why are these features necessary?

- While not fundamentally needed to "boot the box", they enable:
  - Remote management
  - Network boot
  - Failure recovery
  - Other value-add features

# So, what are the risks?

- Eclypsium Inc. (https://www.eclypsium.com/) has delivered Blackhat/Defcon presentations on the dangers of these attack surfaces
- These examples have been presented in public so the “bad actors” out there are aware of them
  - Many implementations have lots of ports open
    - Are they really needed?
  - Some are known to be vulnerable to attack

# Remote management

- Many server type systems allow for a remote management interface in the pre-boot environment
- This may be via a BMC and private network or other mechanism
- This management interface is particularly dangerous as it allows low-level control of:
  - Loading firmware, UEFI drivers, OSes, device drivers, etc.
  - Software/firmware (mis)configuration for evil or denial of service
- Remote management may not be visible to end users and may happen when the system appears to be off
- Many ports are open for remote management
  - Are they really needed?

# BMCs are particularly vulnerable

- Some BMCs have been shown to be very insecure
  - Vulnerabilities may allow unauthorized and persistent remote access
- The IPMI protocol specification has known vulnerabilities
- They depend on a private, air-gapped, network for access security
- It is common for them to use older processor designs and ancient software
- They do not securely boot themselves
- Their firmware is rarely updated
- They can be used by a malevolent host OS/app to compromise the private management network
  - Ref: Blackhat presentation: The Incredible Lightness of BMCs
  - https://blog.rapid7.com/2013/07/02/a-penetration-testers-guide-to-ipmi/

# SMTP & NTFS at boot time

- The Eclypsium folks displayed a motherboard with a UI to send email in the pre-boot environment and support for NTFS
- This was in support of the OEM's customer services
- With this capability built in, malicious pre-boot software could attach any file to an email and send it during pre-boot without the OS knowing

# Another customer service example

- Another OEM provided an interface to download an EFI app over a network for “hardware diagnostics”
- That app could be run without signature checks, bypassing the secure-boot Chain of Trust
- The EFI app can upload results to a customer provided URL
- It can be set to run once or periodically
- Either the download or upload URL could be “spoofed” to transfer anything

# Firmware update

- We, in the UEFI forum, have been discussing the need to update platform firmware regularly
- It is important that OEMs have a path to get security fixes into platforms ASAP
- We cannot depend on end users to download updates
- So let’s do it automatically over a network. What could go wrong?

# Pull updates

- Multiple vendors have added pre-boot code to get updates
- They can go to default OEM URLs for updates or can be customized
- Many can be customized for check frequency
- They typically exchange XML (or similar) messages containing update availability data

# What's wrong with that?

- URLs can be spoofed or replaced
- Any issues with update signature checking can be exploited
- Insecure messages can be altered or replaced directing downloads from anywhere
- Actual testing has shown malformed messages cause firmware hangs (denial of service)
- OEMs have been forced to disable this functionality in hundreds of SKUs (thousands of systems)

# Debugging interfaces

- Traditionally, firmware debugging was done over proprietary hardware interfaces (JTAG, ITP, etc.) which could be fused or depopulated in production systems
- The cost of populating the ITP header is restrictive, and blowing JTAG fuses at EOM is standard
- Newer designs allow debugging over USB, which is convenient but USB ports are, by design, enabled and readily accessible, leaving the firmware configuration as the only gate

# Mitigations

- Make sure your company is following best practices in code development
  - Do targeted code reviews
  - Don’t “roll your own” when there is a quality and tested implementation available
  - See earlier Phoenix plugfest presentations for more examples of best practices

# Hardware/compiler assisted

- Enable
  - NX data execution protection
  - Stack cookies (stack overrun detection)
  - Heap corruption detection
  - Address space layout randomization
- Disable
  - USB debugging interfaces in production systems

# Solutions for firmware update

- The UEFI Forum needs to have some serious discussions around how firmware update gets done
- Leaving OEMs on their own with no direction has resulted in some poor and insecure implementations
- Insecure implementations are damaging to the community and UEFI reputation
- What do we do?
  - Does the forum specify (direct) an approach?
  - Do we provide example implementation(s) via Tianocore?
  - Do we provide whitepapers that provide clear guidance for secure implementations?
- Phoenix believes the forum should take the lead in helping the membership get this right

# Questions?

- Any questions?

Thanks for attending the Fall 2018
UEFI Plugfest

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

*presented by*