Integrity Enhancements for Embedded Linux Devices

David Safford
safford@us.ibm.com

This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency, Cyber Security Division (DHS S&T/HSARPA/CSD), BAA 11-02 and Air Force Research Laboratory, Information Directorate under agreement number FA8750-12-2-0243. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Department of Homeland Security, Air Force Research Laboratory or the U.S. Government.

Embedded Linux Integrity

• Server	$10K+	PB	4768 Crypto card Trusted and Secure Boot
• PC	$1K	TB	TPM Trusted and Secure Boot (Win8)
• mobile	$500	GB	Restricted Boot
• embedded	$50	MB	Nothing
• Sensor	$10	KB	Nothing

Example Embedded Linux Devices

- Pogoplug
- TP-Link MR3020
- D-Link DIR-505
- Linksys WRT54G

MR-3020 Main Components

Three main chips

- SoC (32 bit MIPS)
- RAM (32MB)
- SPI Flash (4MB)

Partition	Name	Size	Contents
mtd0	“boot”	64KB	u-boot
mtd1	“kernel”	1024KB	Linux Kernel
mtd2	“rootfs”	2816KB	Linux root filesystem
mtd3	“config”	64KB	config data
mtd4	“ART”	64KB	radio config data

Recent Embedded Linux Vulnerabilities

- **2012: 4.5 Million home routers compromised in Brazil**
  - https://www.securelist.com/en/blog/208193852/The_tale_of_one_thousand_and_one_DSL_modems
  - Many different Broadcom based devices, across 4 ISPs
  - Redirected all clients to malicious DNS servers
  - Router Exploit: “**get.pl http://192.168.1.1/password.cgi**”
  - Only on internal interface, so **CSRF** required
- **2013: D-Link DIR-645 home router gives away password**
  - http://archives.neohapsis.com/archives/bugtraq/2013-02/0151.html
  - Unauthenticated request from **WAN**/WLAN
  - Exploit: “**curl -d SERVICES=DEVICE.ACCOUNT http://<device ip>/getcfg.php**”

Recent Embedded Linux Vulnerabilities

- **2013: Five new vulnerabilities in Linksys routers**
- https://superevr.com/blog/2013/dont-use-linksys-routers/
  - WRT54GL Firmware Upload CSRF Vulnerability
  - EA2700 XSS Vulnerability
  - EA2700 File Path Traversal Vulnerability (CSRF required)

    POST /apply.cgi
    submit_button=Wireless_Basic&change_action=gozila_cgi&next_page=/etc/passwd
  - EA2700 Password Change Insufficient Authentication and CSRF Vulnerability
  - EA2700 Source Code Disclosure Vulnerability (CSRF)
    - Adding trailing / returns source code of any page
    - **http://192.168.1.1/Management.asp/**

Recent Embedded Linux Vulnerabilities

- **2013: Researchers Hack over a Dozen Home Routers**
- http://securityevaluators.com//content/case-studies/routers/soho_router_hacks.jsp
- 2 remote (CSRF) root (Belkin N300, Belkin N900)
- N300/N900:
  - **<form name="belkinN300" action="http://192.168.2.1/apply.cgi" method="post"/>**
  - **<form name="belkinN900" action="http://192.168.2.1/util_system.html"**
- 4 local (WLAN) root (same, plus Netgear WNDR4700)

Embedded Integrity Issues

- No way to “measure” firmware
  - Supply chain can load malicious firmware
- No way to “lock” firmware
  - Remote malware can root-kit the firmware
- No way to sign-authenticate updates
  - Easy to push malicious firmware updates
- No secure or trusted boot
  - Malware is neither blocked or attested

Current Embedded BIOS Integrity

Device	Measure BIOS?	Lock BIOS?	Signed/local updates?	Secure Boot?
Pogoplug	Yes - SATA	No	No	No
D-Link DIR-505	No	No	No	No
TP-Link MR3020	No	No	No	No
Linksys WRT54G	Yes - JTAG	No	No	No

Traditional Hardware Root of Trust Approaches for Integrity

- “Trusted Boot”
  - **Measurement/Attestation chain** rooted in hardware TPM (trusted platform module)
  - List of SHA-1 hashes of everything read or executed on system
  - If Malicious code or data is accessed, its measurement will be in list, and the measurement cannot be removed from the TPM.
  - Assumes trusted remote management system to verify
- “Secure Boot”
  - **Appraisal chain** rooted in UEFI 3.2.1 BIOS
  - No code is executed unless it is signed
  - Assumes secure boot code AND PUBLIC KEY are immutable
  - Assumes all sensitive files signed

NIST - Integrity Protection Guidelines

- BIOS Integrity Measurement (Is my BIOS correct?)
  - Draft NIST-SP800-155-December-2011
    - BIOS Integrity Measurement (BIOS/UEFI “Trusted Boot”)
- BIOS Integrity Protection (Can I keep it that way?)
  - NIST-SP800-147-April-2011
    - Non-bypassable
    - Authenticated update
    - Secure local update
- Does Not Cover:
  - Signature checking of OS being booted (“Secure Boot”)

Constraints

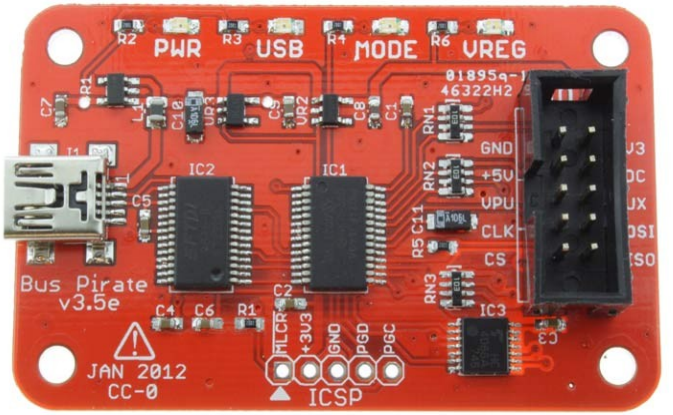
- Vendors unwilling to spend any money on security
  - Retail price of $35 means hardware cost ~ $10
- **What can be done for NO additional cost?**

How can we “Measure” firmware?

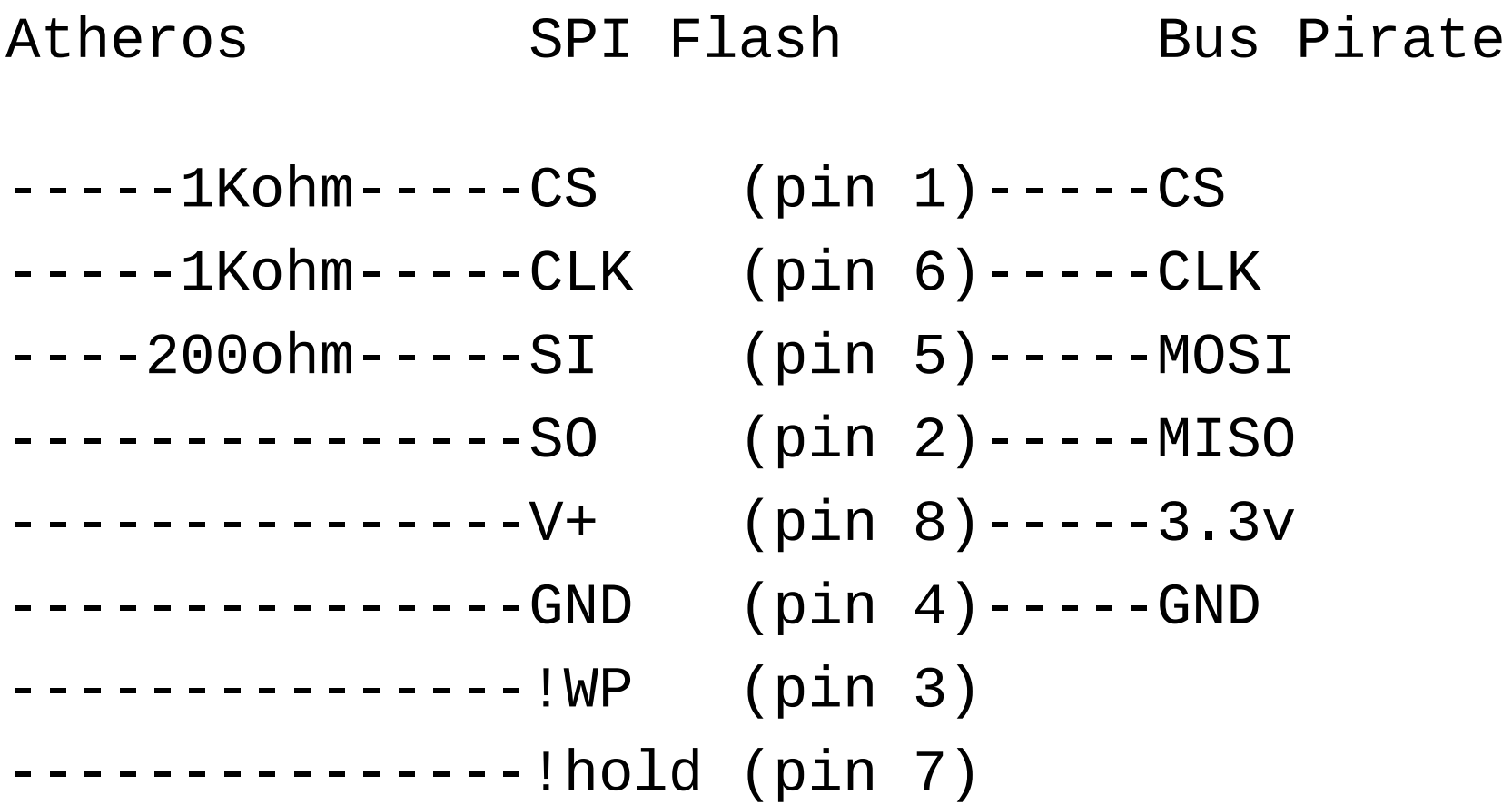
- “Trusted boot” normally requires TPM chip
  - Expensive ($.75)
- Without a measurement root of trust, how do we verify BIOS? We can't just ask it, because it may lie.
- Other methods:
  - SPI Header (many PC BIOS's have this)
  - JTAG header (Linksys WRT54 had this)
  - Immutable (ROM) bootstrap in CPU chip (Broadcom)
  - Unsolder flash chip, and read it in flash programmer (UGH!)

Reading SPI Flash on MR-3020/DIR-505

- DIR-505 and MR3020 are both Atheros SOC.
- Can add SPI in-circuit read support with just 3 additional isolation resistors
  - $30 buspirate (USB – SPI) interface
    - http://dangerousprototypes.com/docs/Bus_Pirate
  - Free flashrom software
    - http://flashrom.org



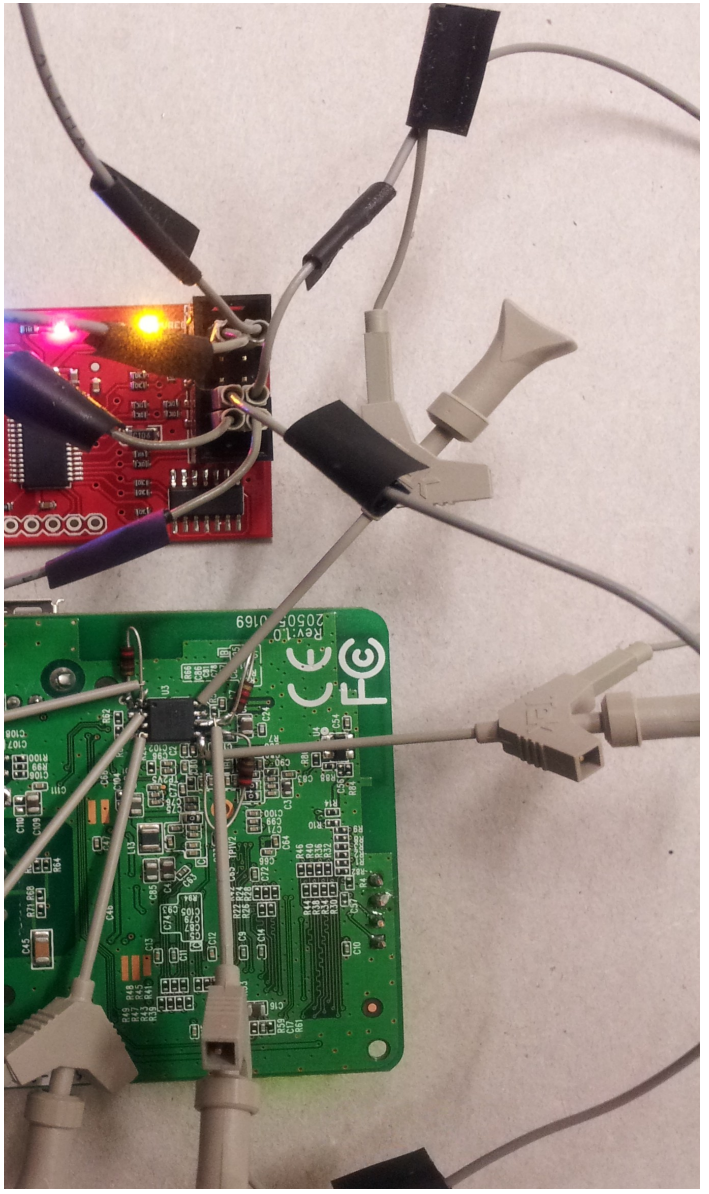
In-Circuit-Programming Mod
(works for both MR3020 and DIR-505)



In-Circuit SPI Flash Reading

Bus Pirate

MR-3020 with SPI isolation resistors



How can we “lock” firmware?

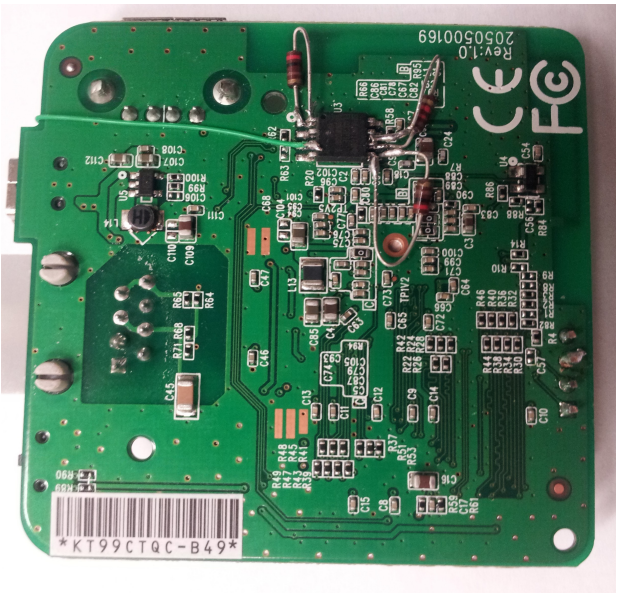
- SPI flash chips have hardware and software write protection
  - Status Register non-volatile protection bits:
    - Write disable bit – when 1 enables write protection
    - BP0 – BP4 – selects which addresses protected
  - !WP pin – when low, enables Hardware Protection Mode which locks status register from software changes.
- How do you allow local updates?
  - Physical presence proven by holding button at boot
  - !WP latch reset by power cycle
  - Immutable boot can update stage2, but verifies signature
  - Disassemble and remove shorting washer (chromebook)

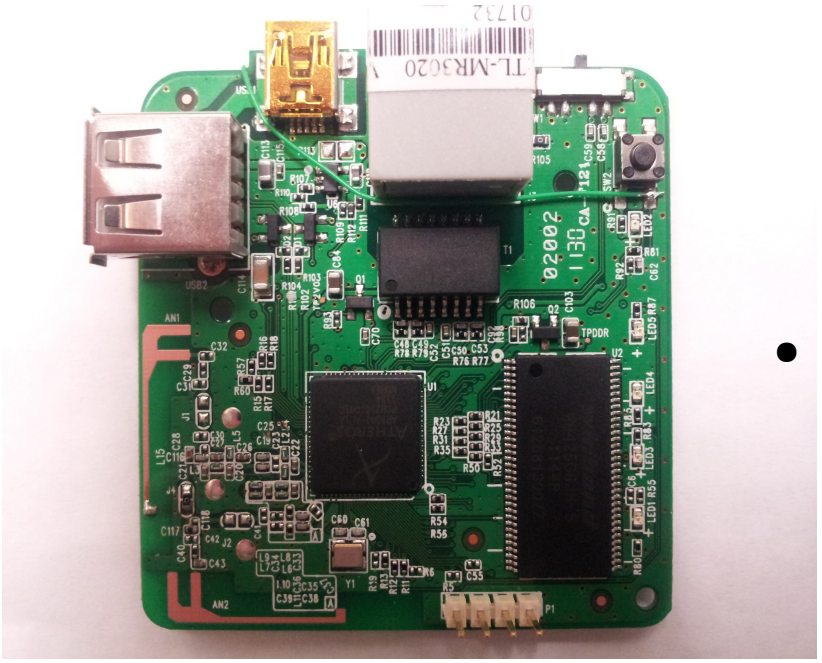
Integrity Protection on MR3020/DIR-505

- MR3020
  - WPS button pressed at boot to unlock flash
  - Reverts to HPM on reset after upgrade
- DIR-505
  - Control !WP directly from unused fourth position of existing mode switch
- U-boot modified
  - Locks flash on every boot
  - If !WP held high at boot, will unlock flash

MR3020 Integrity Protection Mods

- Bottom side - SPI flash with isolation resistors and !WP line from WPS button

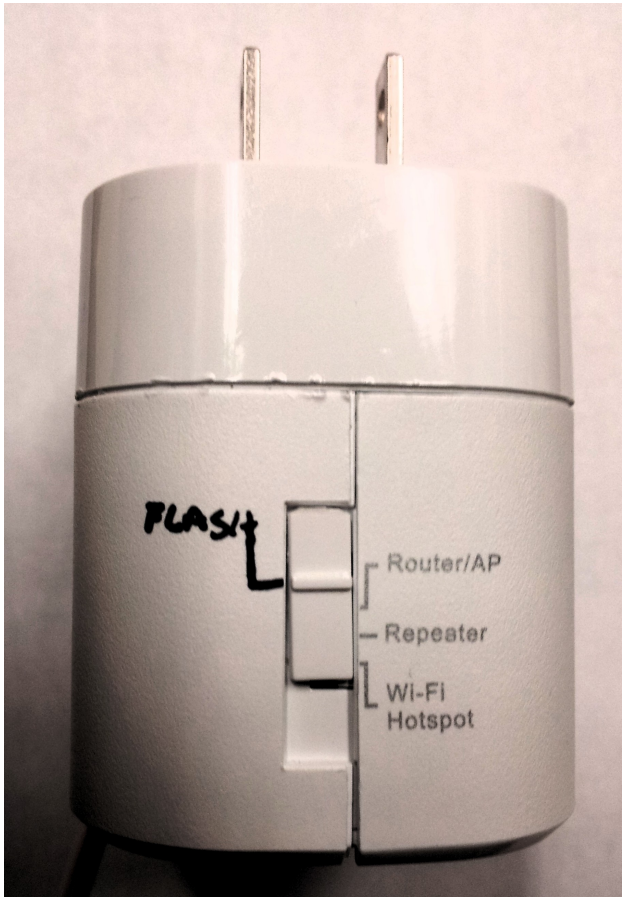




- Top side – WPS button and serial console header

DIR-505 Integrity Protection Mods

- Using the mode switch's extra fourth position for !WP control
- Or can use the WPS button as done with MR3020



Secure Boot on MR3020/DIR-505

- Modify u-boot to verify signature on kernel before boot
  - Public key stored at end of u-boot partition
    - Public key protected with !WP mods
  - Kernel signature stored at end of kernel partition
  - Code derived from PolarSSL code (GPL)
    - Raw binary public key and PKCS1.5 signature
    - Adds 8kb to compressed u-boot (total 62K)

Modified u-boot

- Write protection

```
Write_protect: starting SR = 2
Write_protect: ending SR = 9c
Write_unprotect: starting SR = 9e
Write_unprotect: ending SR = 9e
Write_unprotect failed.

…

Write_protect: starting SR = 2
Write_protect: ending SR = 9c
Write_unprotect: starting SR = 9e
Write_unprotect: ending SR = 2
Write_unprotect succeeded.
```

U-boot modifications

- Secure boot

```
## Booting image at 9f020000 ...
kernel sha1 E9321D87C091F971C8D955C399EBA53807429A61
modulus:
929275845306 3D803DD603D5E777D7888ED1D5BF35786190FA2F23EBC0848AEA
DDA92CA6C3D80B32C4D109BE0F36D6AE7130B9CED7ACDF54CFC7555AC14EEBAB
93A89813FBF3C4F8066D2D800F7C38A81AE31942917403FF4946B0A83D3D3E05
EE57C6F5F5606FB5D4BC6CD34EE0801A5E94BB77B07507233A0BC7BAC8F90F79
signature:
2CB0F653FF3BBCFF2E31ACC0840F02A84975B7167291BB36EEE3F74D02EB3B6A
ACADE02CBCF6E2326230C296E4D8A8D70F309479B388A99591AD5C41938280E3
F51EA9865ED8A0360A0F5BD6A6C676C363B43E5461D9CCF00D46E1B5449CB262
BDE36CAD4AFBEE51ED731BBF48340F290DF8DD844791D81259CEDF99CD1CA2E6
rsa verify kernel succeeded
   Uncompressing Kernel Image ... OK
```

BIOS Integrity with Zero Cost Mods

Device	Measure BIOS?	Lock BIOS?	Signed/local updates?	Secure Boot?
Pogoplug	Yes - SATA	Yes	Yes	Yes
D-Link DIR-505	Yes - buspirate	Yes	Yes	Yes
TP-Link MR3020	Yes - busoirate	Yes	Yes	Yes
Linksys WRT54G	Yes - JTAG	Yes	Yes	Yes

Summary

- At zero cost, we can:
  - Measure firmware
    - Bus pirate
  - Lock firmware
    - HPM mode of flash chips, physical presence for write
  - Authenticate firmware updates
    - RSA signature on updates
  - Add secure boot
    - U-boot verifies signature on linux kernel