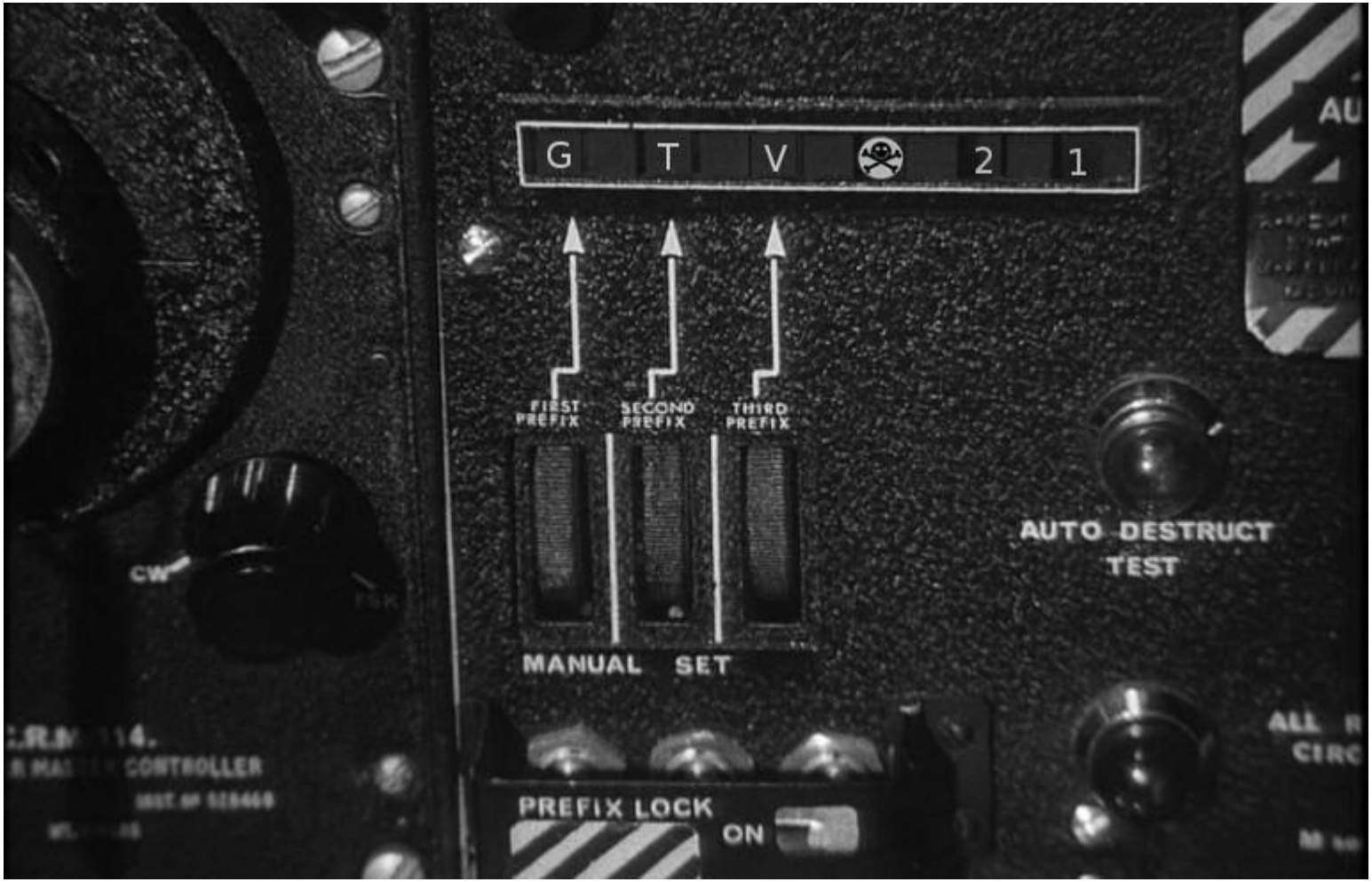
Google TV Or: How I Learned to Stop Worrying and Exploit Secure Boot



GTVHacker: The Team

- GTVHacker is a group of 6 hackers with individual skill sets who work together to unlock Google TV devices.
- Our primary goal is to bypass hardware and software restrictions to allow for unsigned kernels to be loaded and used.
- To date the team has released multiple methods for unlocking Google TV devices.
- The GTVHacker team won a $500 bounty for being the first to root the Google TV.
- We hack things because we believe in open and free hardware. Our current target just happens to be the Google TV.

Members

Mike Baker ([mbm])- Firmware developer and co-founder of OpenWRT

Hans Nielsen (AgentHH)- Senior Security Consultant at Matasano

CJ Heres (cj_000) – IT Systems Manager

gynophage – He's running that big ole DEFCON CTF right now

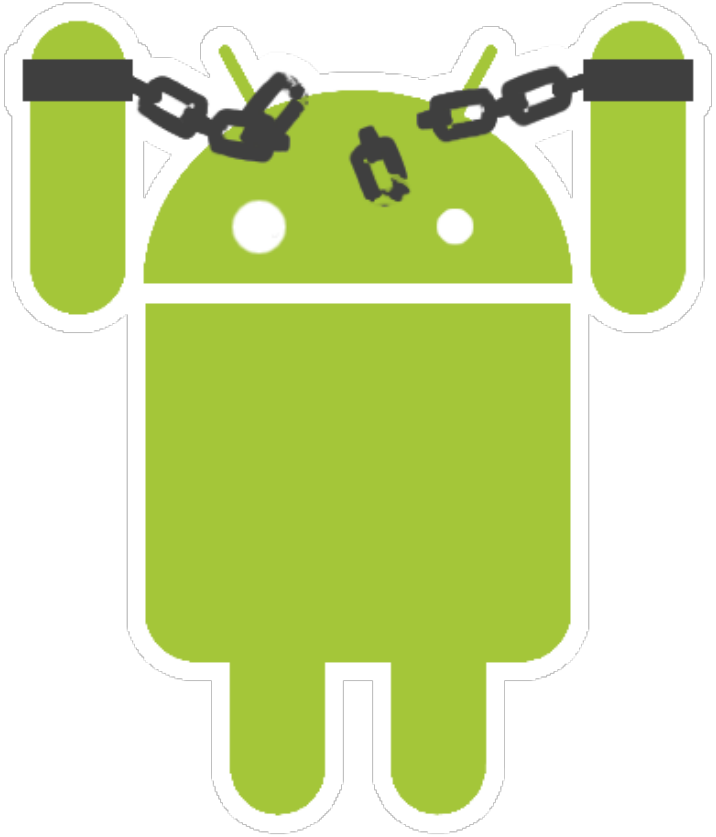
Tom Dwenger (tdweng)- Excellent with APK reversing and anything Java

Amir Etemadieh (Zenofex) – Research Scientist at Accuvant LABS, founded GTVHacker

What's the Google TV?

- Google TV is a platform that:
  - Bridges the gap between your TV and an Android device.
  - Creates an overlay on television stream and also contains an IR transmitter to transmit to media center devices (cable box, TV, sound system).
  - Receives over-the-air updates automatically from OEM manufacturers.
  - Contains a forked version of Chrome with all plugins and extensions disabled.
  - Was originally released without the Android Market available but was eventually updated to include it.
  - Provides a built-in Flash Player, however most content providers block the Google TV.

Why We Hack It



Just a few reasons why we targeted the platform:

- Locked bootloader
- Heavily restricted kernel preventing user modifications
- Generation 1 EOL
- Crippled Flash Player

In short, the Google TV devices are locked down and crippled by their limitations. Our goal is to change that.

Last Year

They released devices... We hacked them all.

Let's make this quick so we can get to the exploits!

Generation 1 Hardware

Logitech Revue

NSZ-GT1

NSZ-[24-46]GT1

Extremely limited number of devices compared to second generation.

First generation has been discontinued.

Recap of Generation 1 Exploits

- Logitech Revue
  - Root UART
  - /dev/devmem (Dan Rosenberg)

- Sony NS[X|Z]-[24-46]GT1
  - Downgrade nodev bug
  - Recovery LCE
  - kexec as module
  - Unsigned Kernels

Along the way: Chrome Flash Player Modification

Hulu and other sites check the Flash Player version string on the box, preventing access.

From:

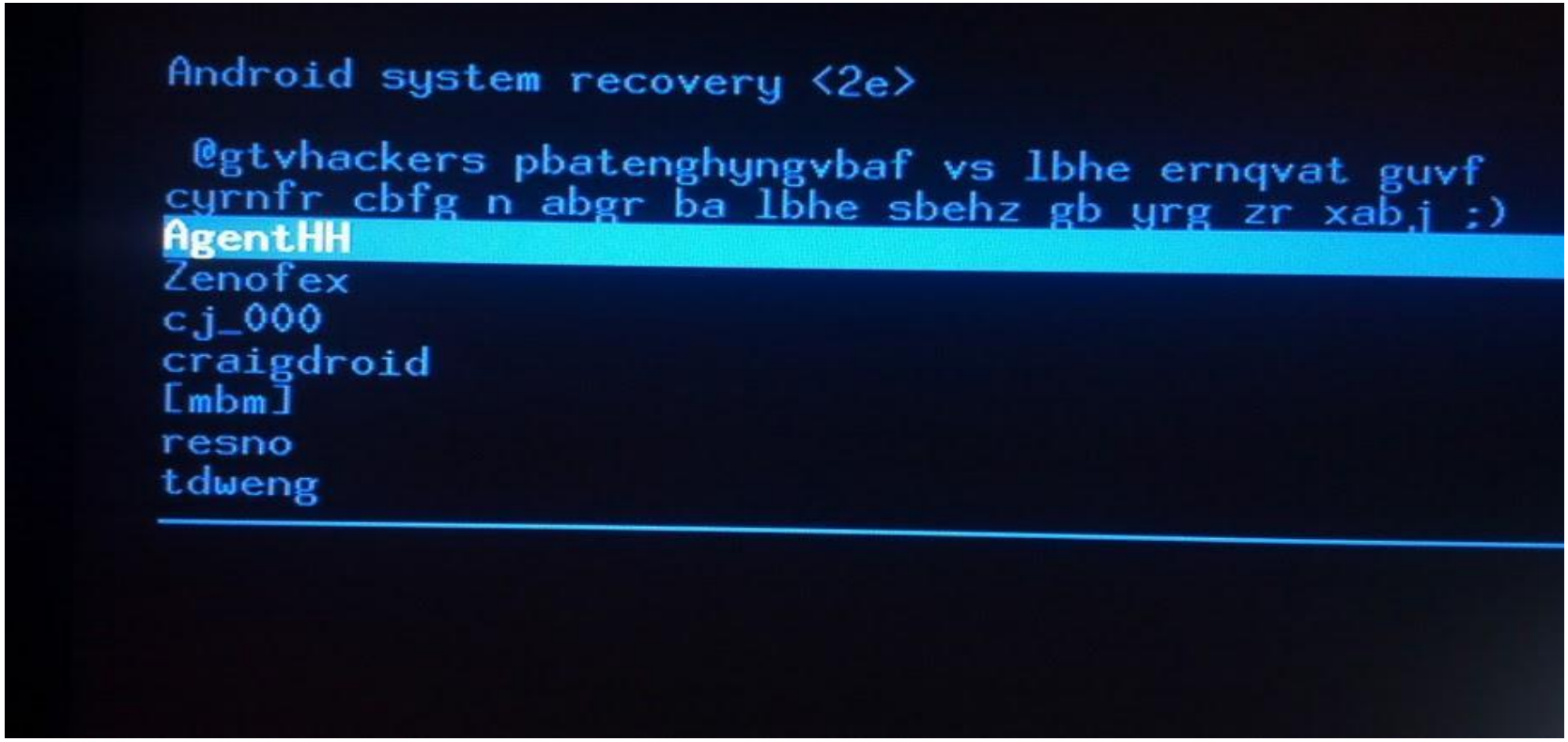
```
00969F52 69 6E 3A 00 47 54 56 20 31 30 2C 31 2C 31 30 37 2C    in:.GTV 10,1,107,
00969F63 31 39 31 00 50 6C 75 67 49 6E 00 35 2E 31 00 25 32    191.PlugIn.5.1.%2
```

To:

```
00969F52 69 6E 3A 00 41 54 56 20 31 30 2C 32 2C 31 30 38 2C    in:.ATV 10,2,108,
00969F63 31 35 31 00 50 6C 75 67 49 6E 00 35 2E 31 00 25 32    151.PlugIn.5.1.%2
```

This simple change teamed with modifying the browser user-agent results in a Content Block Bypass on all blocked sites.

Logitech's Secret Message to Us



"@gtvhackers congratulations if your [*sic*] reading this please post a note on your forum to let me know ;)"

Boxee Box

We disclosed an exploit for Boxee at last year's DefCon

- Software LCE
- Hardware Root UART (under some VIA's)
- Spawned Boxee+ Community
- Modifications based off our root that extend the life and functionality of the Boxee Box
- 308,128 Views since December, 2012
- STILL VULNERABLE :)

TL;DR We dropped an exploit at DEFCON 20, the community responded. Keep up the awesome work Boxee community.

The Next Generation...

Generation 2 Hardware

LG U+

Asus Cube

LG 47/55G2 & G3

Netgear Prime

Sony NSZ-GS7/GS8

Hisense Pulse

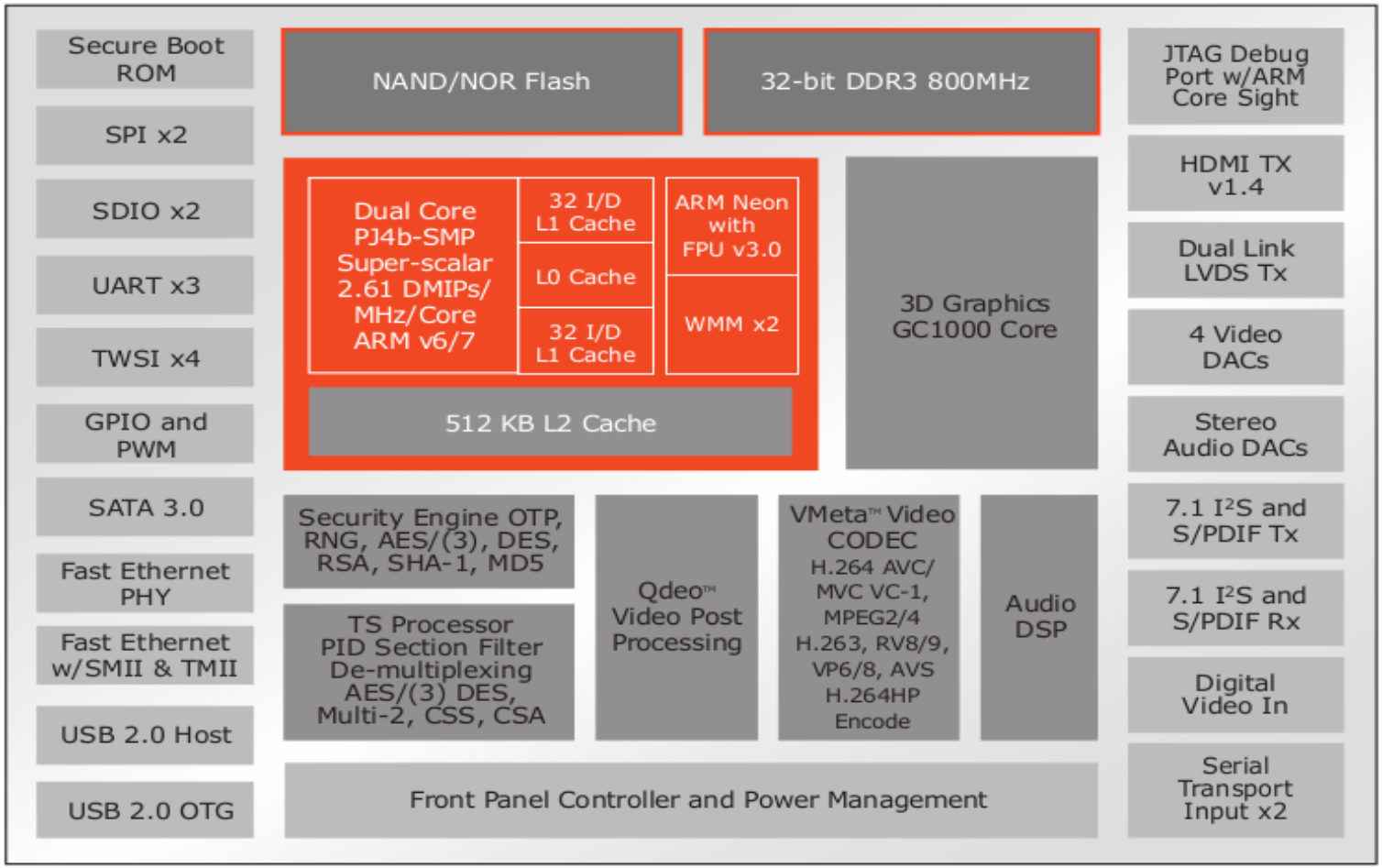
Vizio Co-Star

Similar hardware design throughout most of the generation

Generation 2

- Marvell 88DE3100 based
- ARM – Dual 1.2GHz processors
- Dubbed the “Armada 1500”
- On-die Crypto processor, separate memory
- Secure Boot from ROM via RSA and AES

Marvell Armada 1500 (88DE3100)



Chain of Trust

Chain of Trust Placeholder

Platform Information

- Android 3.2
  - No public vulnerabilities work
- Not a Bionic libc
  - No Android native libraries supported*
- Gen 1: Intel CE4150
  - Single Core Atom ~1.2GHz
- Gen 2: Marvell Armada 1500
  - Dual Core ARM ~1.2GHz each
- Android 4.2.2 incoming for Gen 2
  - Adds Native Libraries, Bionic libc

Sony NSZ-GS7/GS8

- 8GB EMMC Flash
- Best remote
- Larger form factor
  - Internal PSU
  - Built in IR blasters
- $199

Same box as the GS7, but with a voice search remote

Vizio Co-star

- Small form factor
- No Voice Search
- Custom Launcher
- $99 MSRP
- Updates are encrypted via Update Logic
  - Common in all Vizio devices

Hisense Pulse

- 2nd Best Remote
- Launched with ADB running as root
  - Patched shortly after
- $99 MSRP

Hisense Pulse Root

- Teardown showed a root shell over UART
- ro.debuggable=1
- adb root was all it needed!
- Released a script that disabled updates and installed our Chrome Flash Modification

We'll have a select number of USB to TTL adapters available at the Q&A

Netgear NeoTV Prime

- Horrible Remote
- $129 MSRP
- Two exploits
  - One real
  - One oversight

Netgear NeoTV Prime Root

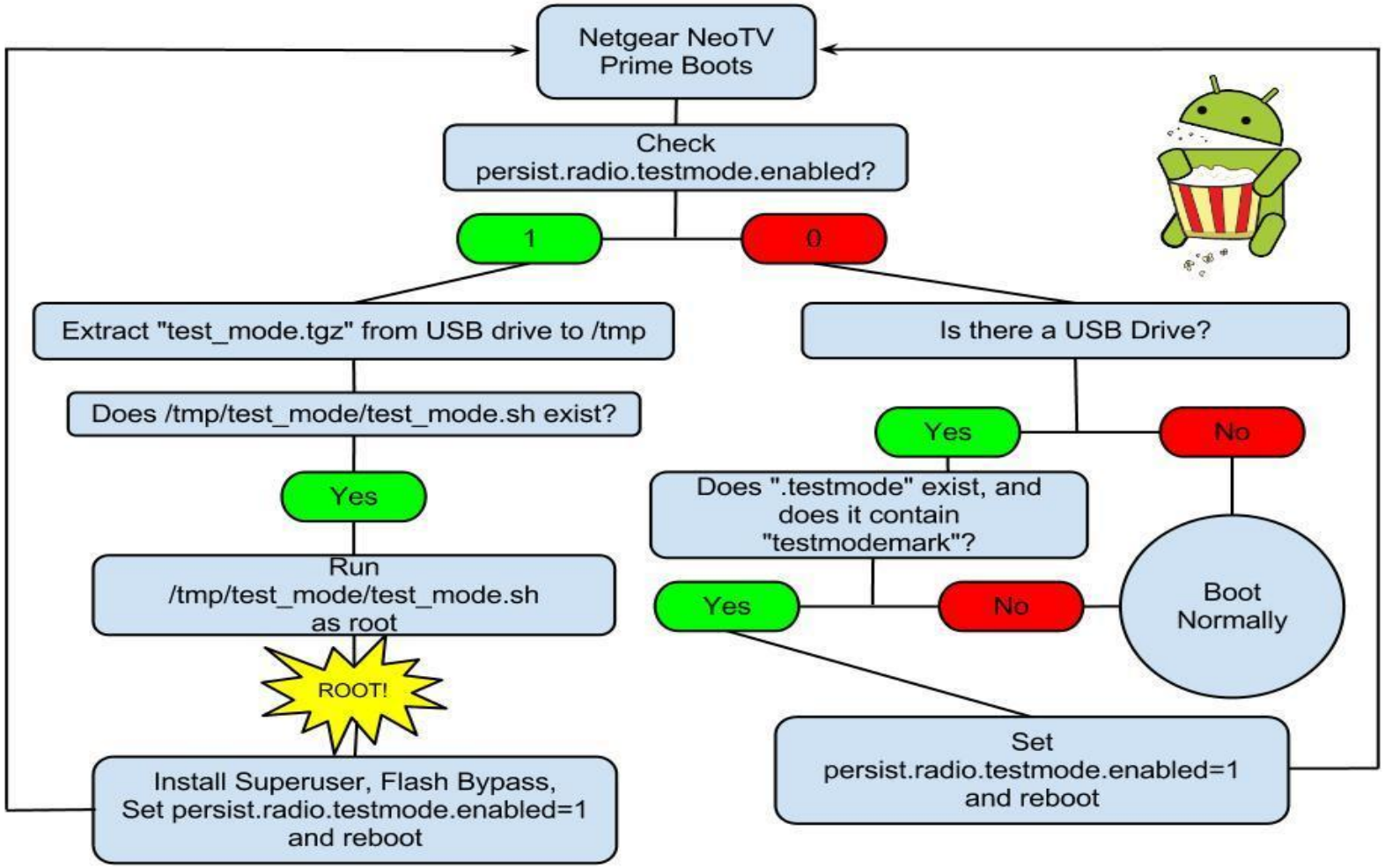
Prime auto-spawned a console as the root user over UART regardless of the security setting.

```
### force to create a console no matter what ###
on property:ro.secure=0
    start console
on property:ro.secure=1
    start console
```

Factory backdoor in the “testmode” service.

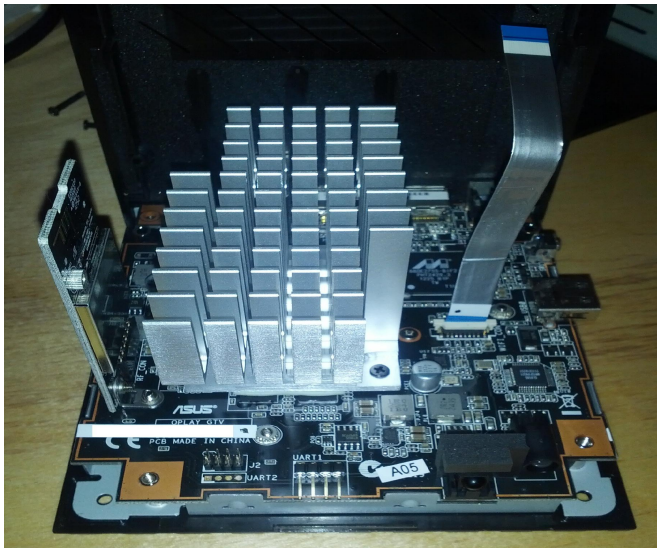
Allowed for execution of code from USB as root.

GTVHacker Netgear NeoTV Prime Root Exploit

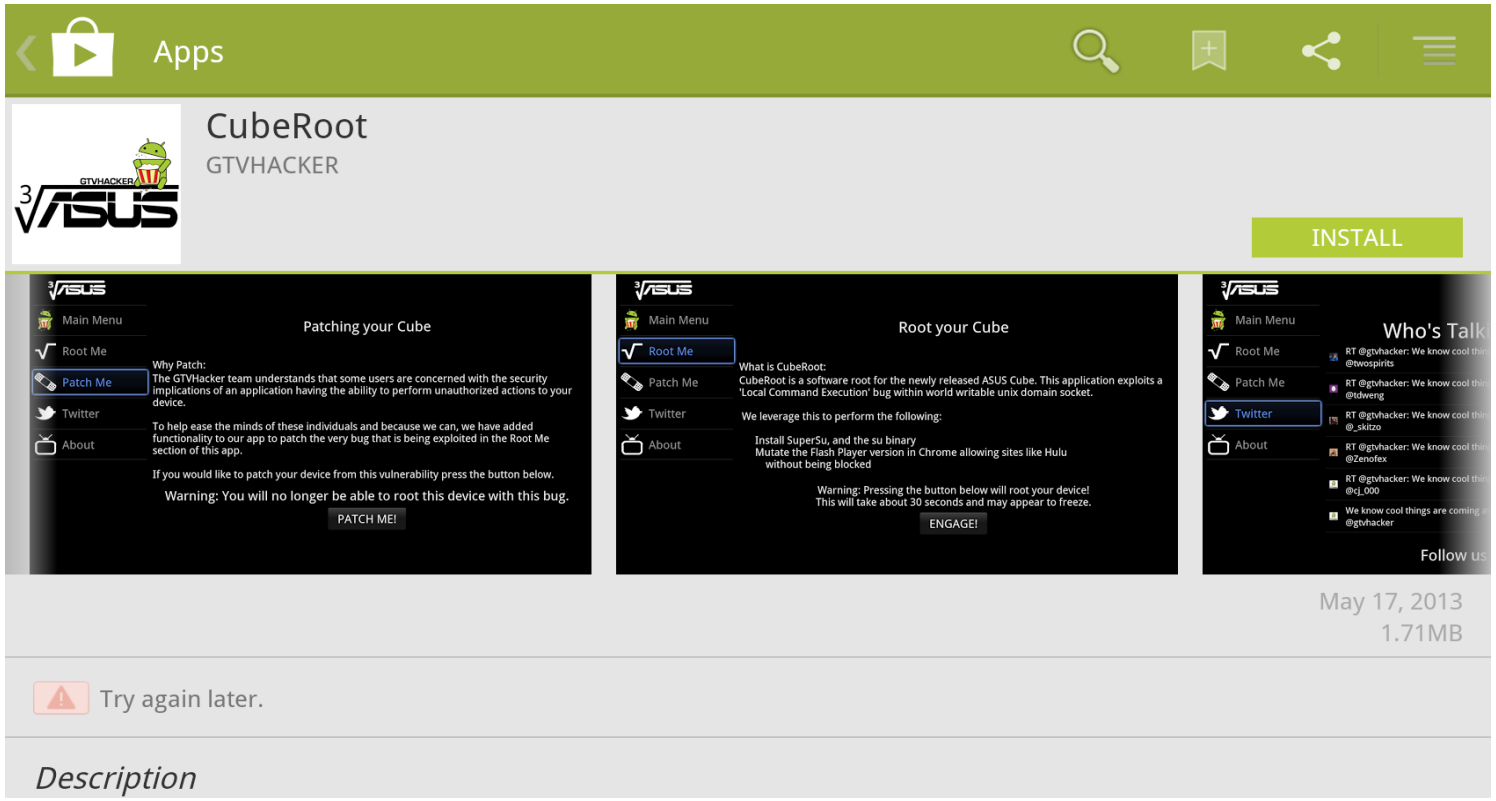


Asus CUBE

- Same generation 2 hardware
- Bad Remote
- $139 MSRP



CubeRoot

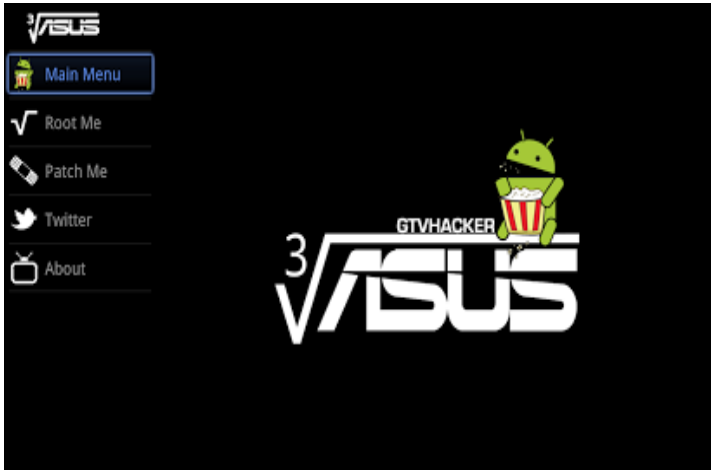


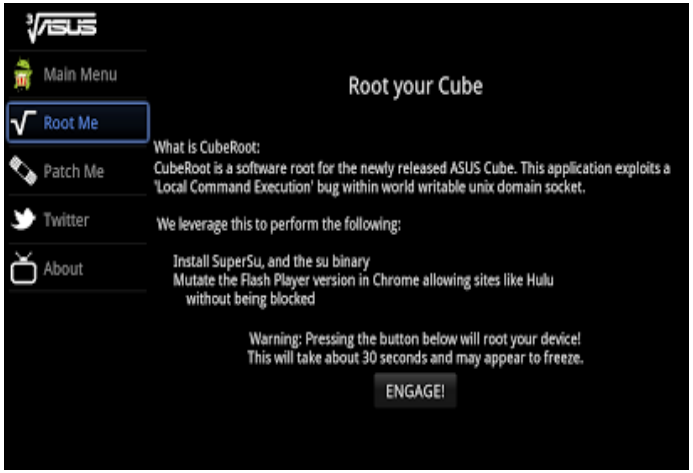
Auto exploits and patches your Asus Cube from an App!

CubeRoot

- Exploited a helper app (oplayhelper) via a world writable socket
- Helper application passed un-sanitized input to the mount command resulting in LCE
  - We triggered the vulnerability from within an Android APK
  - Point, click, pwn
  - Added in Google Play Store

CubeRoot





- Also patches the exploit, to prevent evil apps
- Pulled by Google – get it at GTVHacker.com
  - Downloaded ## boxes
  - Rooted ## boxes (Included 1 eng build)
  - Listed in Google Play store for 6 days
- Patched at the beginning of July
  - Took roughly 2 months

One Root to Rule Them All

Magic USB

- Recall our past exploits with file system nodes and block devices?
  - In the first generation of GoogleTV devices, our original "4 usb recovery exploit" leveraged a USB device improperly not mounted "nodev"
  - That was only two very similar devices.

What about something a bit bigger?

Magic USB

- All Google TV's and some other Android devices are vulnerable.
  - Certain specific Linux boxes too!
- vold mounts NTFS partitions without “nodev”
- A little known “feature” of NTFS is that it supp

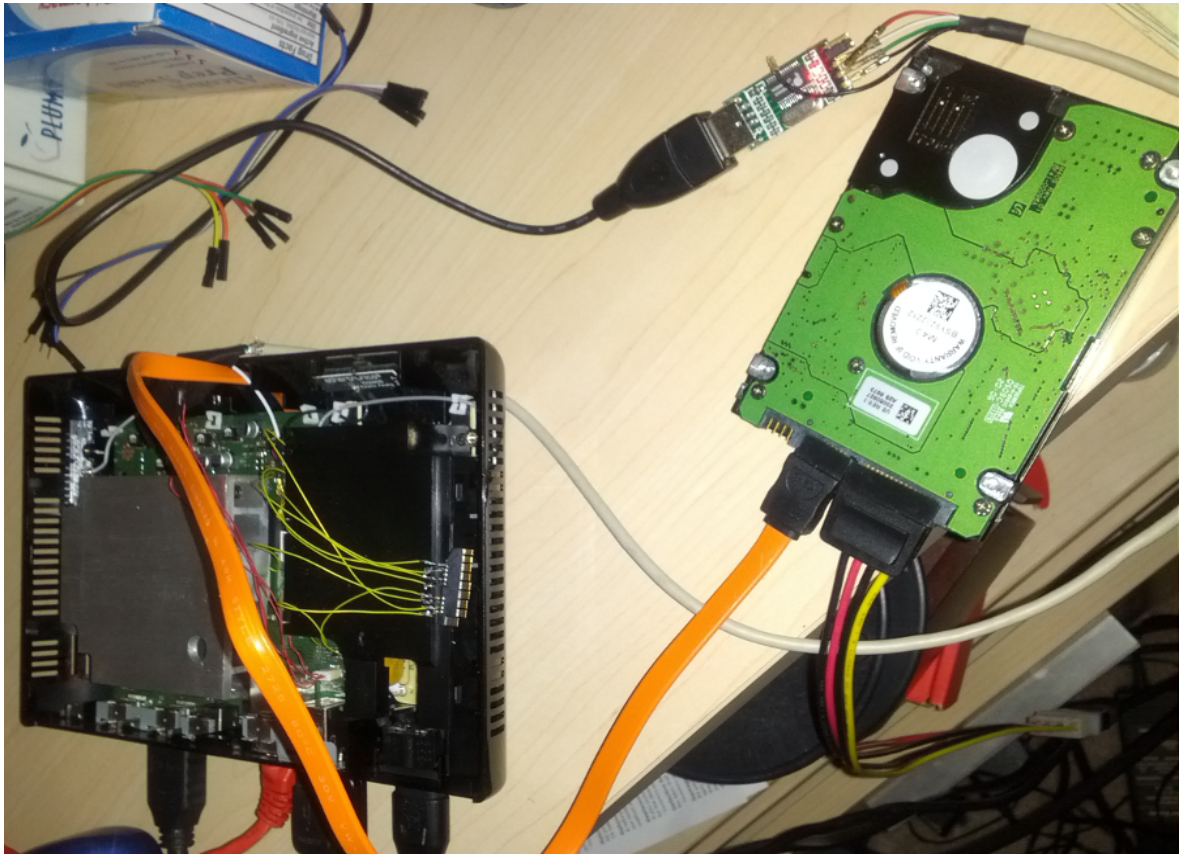
  Linux block / character devices

Magic USB

- NTFS Drive + Block Device
  - Read / Write on any box, any partition.
- Easy root, on every single box!
  - Dump boot.img
  - Patch init.rc or default.prop to ro.secure=0
  - Write it back (as a user, no root needed)
  - Reboot, you are rooted – win!
  - Sony boxes require an additional step

OOOHHHH YEAH

Hardware Mods / Exploits



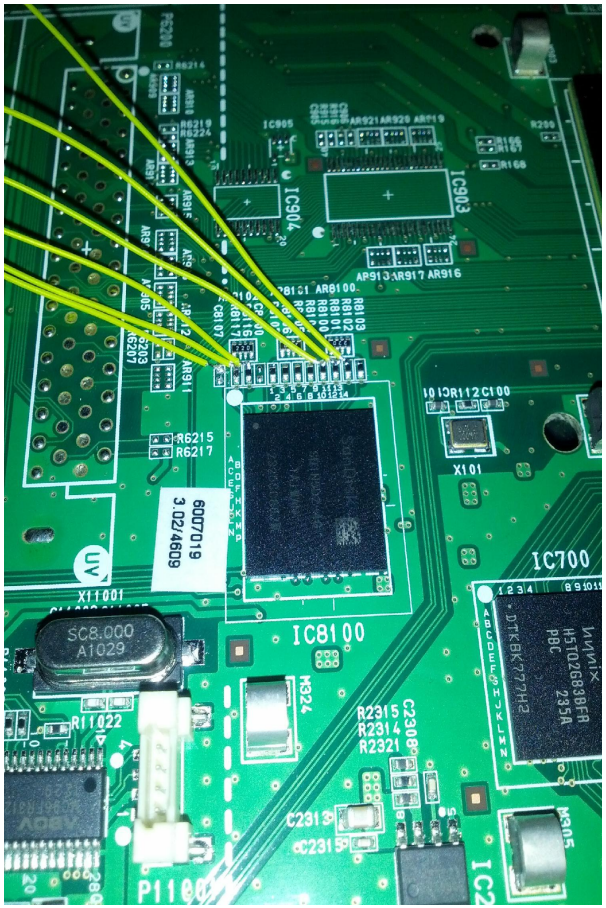
Sony NSZ-GS7 with EMMC->SD and SATA Mods

LG 47/55G2

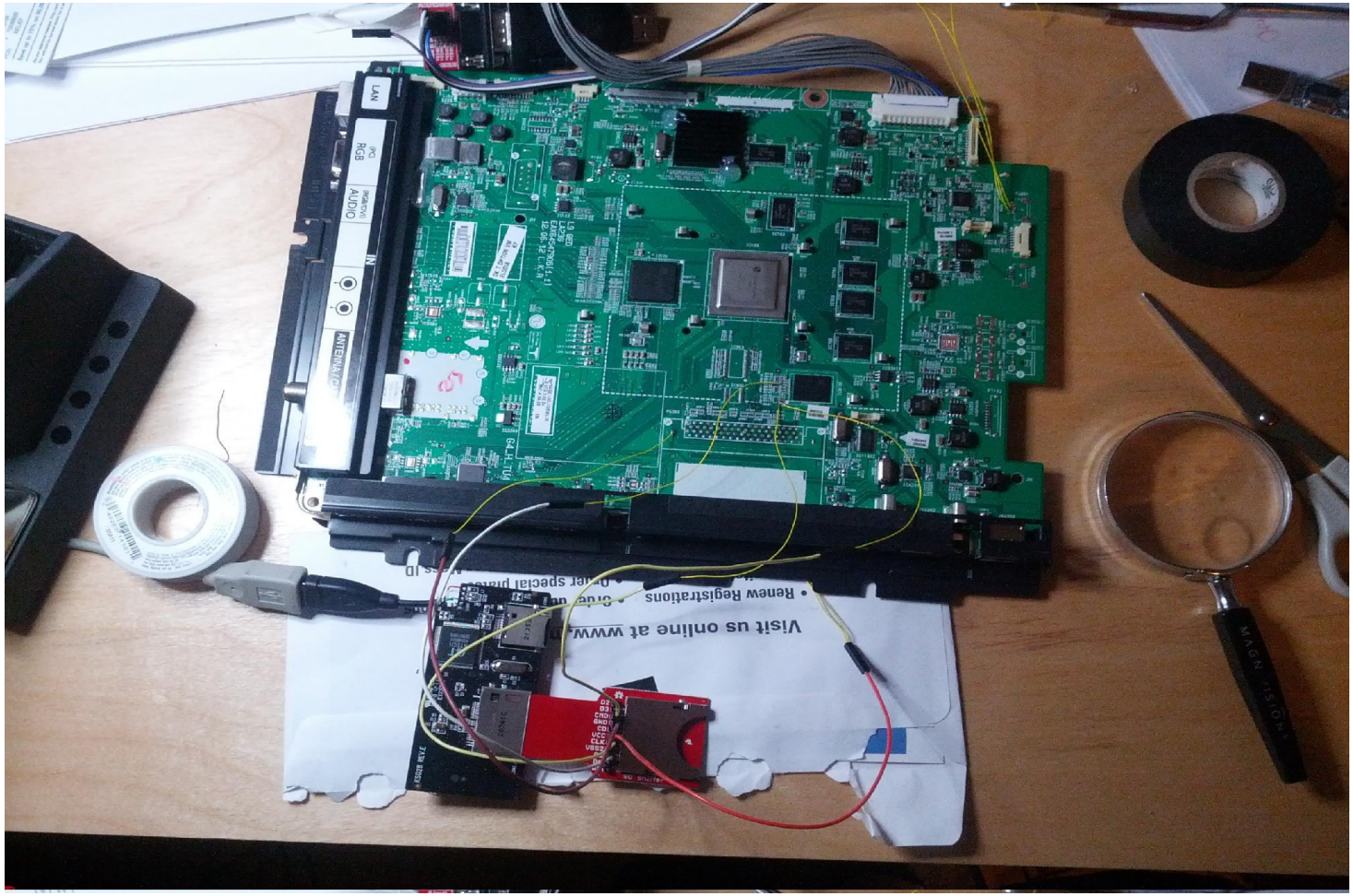
- Dual Core ARM L9
  - aka LG1152
- Signed Everything
  - Even the splash!
- Our "White Whale"
  - Why spend $1K?
  - Next best thing
    - Power supply and Mobo

LG 47/55G2 Root

- Hardware Root!
  - EMMC Flash
    - EMMC
    - MMC
    - SD
- All fall back to SPI mode



LG 47/55G2 Root



LG 47/55G2 Root

```
00100000  09 06 11 20 01 24 02 03  00 00 00 00 1e 00 00 00  |... .$..........|
00100010  6c 39 5f 65 6d 6d 63 00  00 00 00 00 00 00 00 00  |l9_emmc.........|
00100020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00100030  00 00 00 ec 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00100040  00 00 00 00 01 00 00 00  73 65 63 75 72 65 62 6f  |........securebo|
00100050  6f 74 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |ot..............|
00100060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 04 00  |................|
00100070  73 65 63 75 72 65 62 6f  6f 74 2e 62 69 6e 00 00  |secureboot.bin..|
00100080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00100090  00 20 01 00 52 01 00 00  01 01 00 00 03 00 00 00  |. ..R...........|
001000a0  62 6f 6f 74 00 00 00 00  00 00 00 00 00 00 00 00  |boot............|
001000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
001000c0  00 00 04 00 00 00 0c 00  62 6f 6f 74 2e 62 69 6e  |........boot.bin|
001000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
001000e0  00 00 00 00 00 00 00 00  f0 29 04 00 52 01 00 00  |.........)..R...|
001000f0  01 01 00 00 03 00 00 00  70 61 72 74 69 6e 66 6f  |........partinfo|
00100100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00100110  00 00 00 00 00 00 00 00  00 00 10 00 00 00 04 00  |................|
00100120  50 41 52 54 2e 49 4e 46  4f 00 00 00 00 00 00 00  |PART.INFO.......|
00100130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00100140  48 16 00 00 13 00 00 00  01 01 00 00 03 00 00 00  |H...............|
00100150  72 65 63 6f 76 65 72 79  00 00 00 00 00 00 00 00  |recovery........|
00100160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00100170  00 00 14 00 00 00 00 04  72 65 63 6f 76 65 72 79  |........recovery|
00100180  2e 69 6d 67 00 00 00 00  00 00 00 00 00 00 00 00  |.img............|
00100190  00 00 00 00 00 00 00 00  90 70 ad 00 00 00 00 00  |.........p......|
```

partinfo at 0x100000.

Take the filename, count back 6 bytes and byteswap – your location.

/system is at 122,159,104

mount -text4 -o,skip=122159104 /dev/sdXX /mnt/system

LG 47/55G2 Root

- Root FS is a signed squashfs image
- Init script calls: /system/vendor/bin/init_forcestripped.sh
- Mount, edit to spawn telnet, root shell over uart, or over PL2303 USB serial adapter.
- Debug agent (dongle needed) runs over UART

```
on boot
    # Fix system date if necessary.
    exec /system/bin/fixdate

    # TODO: remove an unnecessary comment.
    # Init osd0 for fast-boot
    setprop debug.sf.nobootanimation 1
    write /proc/lg/fbdev/scrclear "0"
    start init_osd0

    # TODO: remove an unnecessary comment.
    # Init volume
    #exec /system/vendor/bin/init_volume.sh
    start init_volume

    # Sets ro.forcestripped based on build.prop or persist.gtv.forcestripped
    exec /system/vendor/bin/init_forcestripped.sh
```

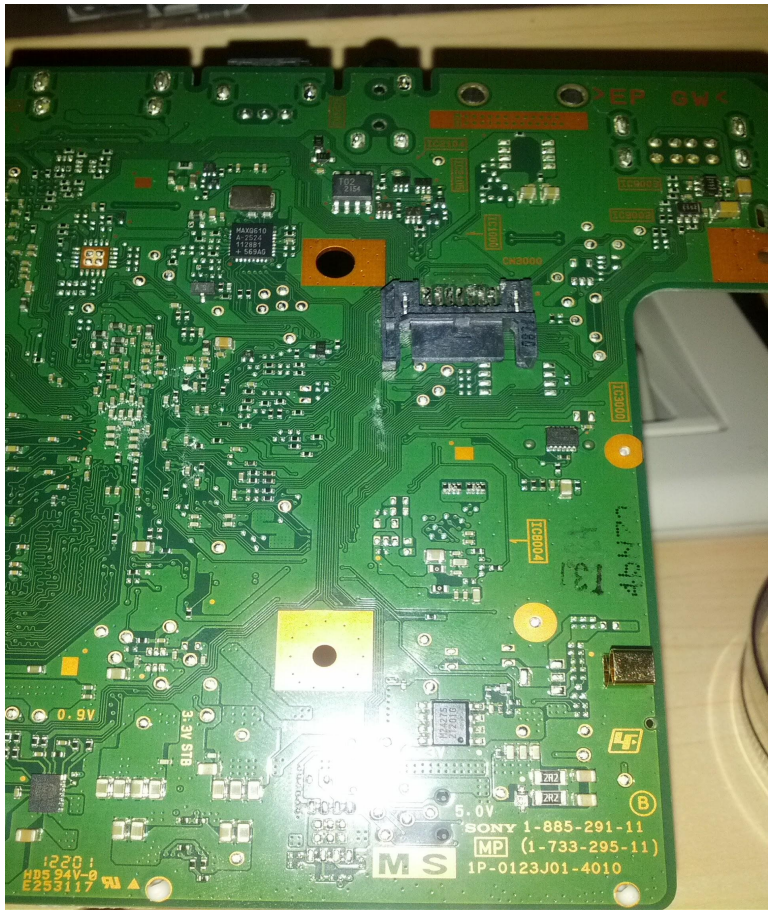
Sony NSZ-GS7/GS8

- Sony also uses an EMMC Flash making interfacing easier
  - Boot & system are not signed
- To gain root we rewrite /boot or /system
  - However, the RSA signed init scripts check for certain props
    - EX: Check for ro.secure=0, if so, reboot
      - Since we can modify /boot we can remove the check
  - Sony also disabled dd, insmod, and some other bits via kernel calls
  - Being able to write /system and /boot you can change most restrictions at will!

Sony NSZ-GS7/GS8

- SATA HDD
  - Jumpers / caps over front points
  - Add SATA connector on the back
  - Connect a HDD. Ach, it's not being detected!

But no kernel support for SATA



Now What?

We've got root but we want more.

Marvell Armada 1500 Secure Boot Exploit

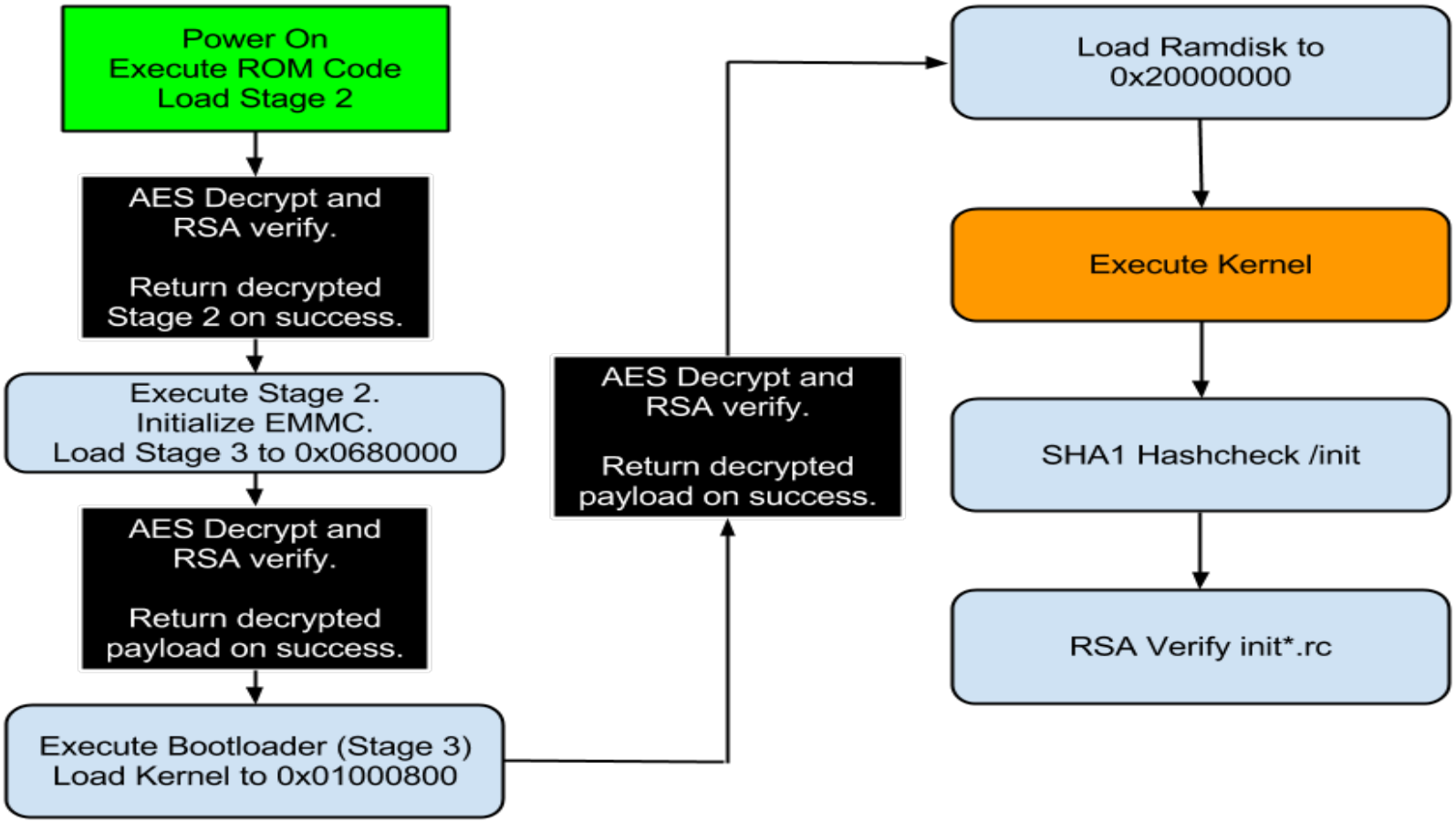
- Armada 1000 = 88de3010
- Armada 1500 = 88de3100

May also work on the Armada 1000

Marvell Armada 1500 Secure Boot Exploit

- Sony NSZ-GS7
- Netgear NeoTV Prime
- Vizio Co-Star
- Hisense Pulse
- Asus CUBE
- Sony NSZ-GS8
- LG U+ IPTV
- Google “Berlin”
- ZeroDesktop MiiPC
- Hisense XT780 TV
- Lenovo S31/S61 TV
- TCL MoVo
- And Others!

Detailed Security Overview



Bootloader Messages

```
cj@cj-desktop: ~
File  Edit  View  Search  Terminal  Help
WOL MAC address: 08:60:6e:e1:72:f8
Image3 bootargs: androidboot.console=ttyS0 console=ttyS0,115200 init=/init pxa3xx_nand.use_cache_progr2
mkbootimg bootargs: androidboot.hardware=asusberlin root=/dev/mtdblock:boot
Generated bootargs: androidboot.hardware=asusberlin root=/dev/mtdblock:boot androidboot.console=ttyS0 2
Send bootmode=0 to SM.
Re[s1p00on7d]iIn(gS Mt)o:  SsMy.ss..t
tWea r=m  0uxp0.,                    a
tByopoet  =n o0rxm8a,l  cGoTnVt einmta g=e
x0                                        0
[Flash Write]page=0x00005788, buf=0x0069fb2c, size=8192
[mv_nand_write_large_page,788] addr=0x0af10000, buf=0x006b0048, oob=0
[Flash Write]page=0x00005789, buf=0x006a1b2c, size=8192
[mv_nand_write_large_page,788] addr=0x0af
fts: record v197 commited @ 0x00310000
Start kernel at 0x01008000
Uncompressing Linux... done, booting the kernel.
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 2.6.35.14 (jason@jason-P43SJ) (gcc version 4.4.5 20100614 (prerelease) (F3
[    0.000000] CPU: ARMv7 Processor [562f5841] revision 1 (ARMv7), cr=10c53c7d
[    0.000000] CPU: VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
[    0.000000] Machine: MV88DE3100
[    0.000000] Memory policy: ECC disabled, Data cache writealloc
[    0.000000] PERCPU: Embedded 7 pages/cpu @80b38000 s6624 r8192 d13856 u65536
[    0.000000] pcpu-alloc: s6624 r8192 d13856 u65536 alloc=16*4096
[    0.000000] pcpu-alloc: [0] 0 [0] 1
[    0.000000] Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 162560
[    0.000000] Kernel command line: androidboot.hardware=asusberlin root=/dev/mtdblock:boot androidboo2
```

Start kernel at 0x01008000

Android Kernel + Marvell Secure Image

```
00000000|414E 4452 4F49 4421 20BF 2800 0080 0001|ANDROID! .(.....
00000010|78E0 0100 0080 0001 0000 0000 0000 F001|x...............
00000020|0001 0001 0008 0000 0000 0000 0000 0000|................
00000030|0000 0000 0000 0000 0000 0000 0000 0000|................
00000040|2063 6F6E 736F 6C65 3D20 696E 6974 3D2F| console= init=/
00000050|696E 6974 2072 6F6F 743D 2F64 6576 2F72|init root=/dev/r
00000060|616D 3020 726F 6F74 6673 7479 7065 3D65|am0 rootfstype=e
00000070|7874 3420 7261 6D64 6973 6B5F 7369 7A65|xt4 ramdisk_size
00000080|3D31 3034 3835 3736 3030 0000 0000 0000|=104857600......
00000090|0000 0000 0000 0000 0000 0000 0000 0000|................
000000A0|0000 0000 0000 0000 0000 0000 0000 0000|................
        ********************************************
00000220|0000 0000 0000 0000 0000 0000 0000 0000|................
00000230|0000 0000 0000 0000 0000 0000 0000 0000|................
00000240|0CBD 754A C641 116E 183E 3F95 AFDC 8C62|..uJ.A.n.>?....b
00000250|04CA 734E 0000 0000 0000 0000 0000 0000|..sN............
00000260|0000 0000 0000 0000 0000 0000 0000 0000|................
00000270|0000 0000 0000 0000 0000 0000 0000 0000|................
        ********************************************
000007E0|0000 0000 0000 0000 0000 0000 0000 0000|................
000007F0|0000 0000 0000 0000 0000 0000 0000 0000|................
00000800|8000 0000 0000 4248 1003 C002 4069 ....|......BH....@i
00000810|BC18 7E62 2DE0 F662 15D4 D23C 9D48 1618|..~b-..b...<.H..
00000820|0000 0000 0000 0000 0000 0000 0000 0000|................
00000830|0000 0000 0000 0000 0000 0000 0000 0000|................
00000840|0200 0000 0280 0000 B670 BEFF 7233 1F87|.........p..r3..
00000850|31CD EA9C 9B22 601D 1127 4684 0000 0000|1...."`..'F.....
00000860|0000 0000 0000 0000 0000 0000 0000 0000|................
00000870|0000 0000 0000 0000 0000 9CBE 2800 ....|............(.
00000880|7A06 3B30 E9F1 064F D7F6 46FA BB40 1E5E|z.;0...O..F..@.^
00000890|827A 49C5 30BA B16C DE03 6FB5 4462 CED5|.zI.0..l..o.Db..
000008A0|AC78 1064 E25F 6165 E7F5 ACD2 C02A 973E|.x.d._ae....*.>
000008B0|6327 0207 9019 B472 06F2 56C9 B5C3 FFAB|c'.....r..V.....
000008C0|61B4 A6FD EEA4 28B1 EAA7 9364 C012 B1BD|a.....(....d....
000008D0|0F06 6937 BE81 5BDA 6442 29D0 CCE0 C01D|..i7..[.dB).....
000008E0|E232 8070 2706 3868 8ADA 57D5 44D2 E76C|.2.p'.8h..W.D..l
000008F0|809E B8DB 81BB 2F73 D630 E607 EF9B 0DF0|...../s.0......
00000900|9787 E489 505A 25A7 CC29 1D3E 890A FA08|....PZ%..).>....
00000910|6999 F461 E719 1DE9 C41D DD9E F263 2025|i..a........c %
```

Android Kernel Header

Mkbootimg/
bootimg.h

```
#define BOOT_MAGIC "ANDROID!"
#define BOOT_MAGIC_SIZE 8
#define BOOT_NAME_SIZE 16
#define BOOT_ARGS_SIZE 512

struct boot_img_hdr
{
    unsigned char magic[BOOT_MAGIC_SIZE];

    unsigned kernel_size;  /* size in bytes */
    unsigned kernel_addr;  /* physical load addr */

    unsigned ramdisk_size; /* size in bytes */
    unsigned ramdisk_addr; /* physical load addr */

    unsigned second_size;  /* size in bytes */
    unsigned second_addr;  /* physical load addr */

    unsigned tags_addr;    /* physical addr for kernel tags */
    unsigned page_size;    /* flash page size we assume */
    unsigned unused[2];    /* future expansion: should be 0 */

    unsigned char name[BOOT_NAME_SIZE]; /* asciiz product name */

    unsigned char cmdline[BOOT_ARGS_SIZE];

    unsigned id[8]; /* timestamp / checksum / sha1 / etc */
};

/*
** +-----------------+
** | boot header     | 1 page
** +-----------------+
** | kernel          | n pages
** +-----------------+
** | ramdisk         | m pages
** +-----------------+
** | second stage    | o pages
** +-----------------+
**
```

Android Kernel + MV Secure Image

Android Magic | Kernel Size | Kernel Load Address

```
00000000 414E 4452 4F49 4421 20BF 2800 0080 0001 ANDROID! .(.....
00000010 78E0 0100 0080 0001 0000 0000 0000 F001 x...............
```

Ramdisk Size | Ramdisk Load Address

```
00000040 2063 6F6E 736F 6C65 3D20 696E 6974 3D2F  console= init=/
00000050 696E 6974 2072 6F6F 743D 2F64 6576 2F72 init root=/dev/r
00000060 616D 3020 726F 6F74 6673 7479 7065 3D65 am0 rootfstype=e
00000070 7874 3420 7261 6D64 6973 6B5F 7369 7A65 xt4 ramdisk_size
00000080 3D31 3034 3835 3736 3030 0000 0000 0000 =104857600......
00000090 0000 0000 0000 0000 0000 0000 0000 0000 ................
000000A0 0000 0000 0000 0000 0000 0000 0000 0000 ................
         ************************************
00000220 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000230 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000240 0CBD 754A C641 116E 183E 3F95 AFDC 8C62 ..uJ.A.n.>?....b
00000250 04CA 734E 0000 0000 0000 0000 0000 0000 ..sN............
```

SHA1 Hash

```
         ************************************
000007E0 8000 0000 0000 0000 4248 1003 C002 4069 ........BH....@i
000007F0 BC18 7E62 2DE0 F662 15D4 D23C 9D48 1618 ..~b-..b...<.H..
00000800 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000810 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000820 0200 0000 0280 0000 B670 BEFF 7233 1F87 .........p..r3..
00000830 31CD EA9C 9B22 601D 1127 4684 0000 0000 1....."`..'F.....
00000840 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000850 0000 0000 0000 0000 0000 0000 9CBE 2800 ..............(.
00000860 7A06 3B30 E9F1 064F D7F6 46FA BB40 1E5E z.;0...O..F..@.^
00000870 827A 49C5 30BA B16C DE03 6FB5 4462 CED5 .zI.0..l..o.Db..
00000880 AC78 1064 E25F 6165 E7F5 ACD2 C02A 973E .x.d._ae.....*.>
00000890 6327 0207 9019 B472 06F2 56C9 B5C3 FFAB c'.....r..V.....
000008A0 61B4 A6FD EEA4 28B1 EAA7 9364 C012 B1BD a.....(....d....
000008B0 0F06 6937 BE81 5BDA 6442 29D0 CCE0 C01D ..i7..[.dB).....
000008C0 E232 8070 2706 3868 8ADA 57D5 44D2 E76C .2.p'.8h..W.D..l
000008D0 809E B8DB 81BB 2F73 D630 E607 EF9B 0DF0 ....../s.0......
000008E0 9787 E489 505A 25A7 CC29 1D3E 890A FA08 ....PZ%..).>....
000008F0 6999 F461 E719 1DE9 C41D DD9E F263 2025 i..a.........c %
00000900 65F1 514E 4784 8F23 8672 77AD FFF1 F445 e.QNG..#.rw....E
00000910 EAB2 6FD9 4E56 89EC 7B47 7F9A 5B6D F8CB ..o.NV..{G..[m..
```

Kernel Arguments (only for show!)

Key Index

Signature

Encrypted Data Size

RSA 1024 Bit Signature

AES-128-CBC Encrypted

A Second Look

```
00000000|414E 4452 4F49 4421 20BF 2800 0080 0001|ANDROID! .(.....
00000010|78E0 0100 0080 0001 0000 0000 0000 F001|x...............
00000020|0001 0001 0008 0000 0000 0000 0000 0000|................
00000030|0000 0000 0000 0000 0000 0000 0000 0000|................
00000040|2063 6F6E 736F 6C65 3D20 696E 6974 3D2F| console= init=/
00000050|696E 6974 2072 6F6F 743D 2F64 6576 2F72|init root=/dev/r
00000060|616D 3020 726F 6F74 6673 7479 7065 3D65|am0 rootfstype=e
00000070|7874 3420 7261 6D64 6973 6B5F 7369 7A65|xt4 ramdisk_size
00000080|3D31 3034 3835 3736 3030 0000 0000 0000|=104857600......
00000090|0000 0000 0000 0000 0000 0000 0000 0000|................
000000A0|0000 0000 0000 0000 0000 0000 0000 0000|................
        ********************************************
```

Red = Ramdisk Size

Black = Ramdisk Load Address

You got your Ramdisk in my Kernel!

Secure Boot Exploit

- Note the ramdisk load address
  - Can be modified without breaking kernel signature
- Allows us to load a "ramdisk" anywhere in memory
  - Ramdisk in this case is a chunk of our own unsigned code
- Copies in our "ramdisk" at the address specified, and without any additional checks, we can run our own unsigned code

Area Of Attack (Pseudocode)

```
//hard coded load address
kernel_load_addr = 0x01000800;

//read kernel from emmc / nand flash to memory
do_emmcread(kernel_buf,kernel_load_addr);

//some stuff to parse the header into nice names

printf("Kernel image decrypt start now");

//start to decrypt and verify, send the image to the security processor
res = LoadImage(kernel_buf, header_kernel_size);

printf("Kernel image decrypt finished");

        if(res){
                printf("Verify Kernel image failed!");
                return 1;
        }else{
                //copy kernelbuf
                memcpy(kernel_buf, kernel_load_addr, header_kernel_size);
        }

        if(ramdisk_header_size){
                do_emmcread(ramdisk_buf, header_ramdisk_size);
                memcpy(ramdisk_buf, header_ramdisk_load_addr, header_ramdisk_size);
        }

printf("verify Kernel image passed.");
```

New Boot Flow / Memory Map

New Boot Flow / Memory Map

Exploit Process

- GTVHacker Custom Recovery on Sony NSZ-GS7
  - Sony box has additional security
  - Append a tiny secure image that will validate
  - Normal signed kernel will do
- Add on our custom Recovery + Kernel (w/ ramdisk)
  - Change Ramdisk size to match our new "ramdisk"
- Set Ramdisk Load Address:
    - 0x1008000 – Size of Signed Kernel
- Our custom Recovery ends up at 0x1008000, and boots!

Exploit Process

Exploit Image Placeholder

U-Boot

- We can also trigger the exploit and run uboot
- ASUS was kind enough to GPL parts, and with some patches, it runs
- Load a kernel via TFTP, Flash, or USB for development

```
MBOoDoEt= 0nxo0r T
al GTV image    m
Start kernel at 0x01008000
raise: Signal # 8 caught


U-Boot 2010.09-rc1 (Jun 03 2013 - 13:49:58)

Marvell U-boot Version 2.4 for MV88DE3100(B1) ASIC
 _   _     ____            _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/     |____/ \___/ \___/ \__|
DRAM:  1 GiB
[mv_nand_chip_init,634] NFC dump register:
        NDTR0CS0  = 0x84840a12
        NDTR1CS0  = 0x00208662
        NDREDEL   = 0x00000000
        NDSR      = 0x00000000
        NDCR      = 0x0186dfff
        NDECCCTRL = 0x10000000
wait bit 00000800 time out!
Nand flash init error!
NAND init error, restore back to SPI flash.
Flash: 16 MiB
Detecting eMMC ...
CMD1 Card OCR=0xc0ff8080 SDHC=1
hardware reset is permanently enabled
eMMC Indentify done.
EMMC: CID_SerialNum=11b7001b CardCapacity = 0x40000000  RCA=11b7001b
env_relocate[514] malloced ENV at 0cb00008
environment in SPI flash is invalid.
fail to load enviroment, use default.
Fail to load environment from eMMC flash.
macaddr: 00:80:11:11:00:41
In:    serial
Out:   serial
Err:   serial
MMC:   MV_SDIO: 0
Net:   Galois eth0, ethaddr=00:80:11:11:00:41
eth0
Hit any key to stop autoboot:  0
MV88DE3100|>
```

Future Research

Areas that need more work:

- Unsigned kernels on Gen 1 (Revue) w/ NTFS exploit

Demo

Thank You!

Slide Resources can be found at:

http://DC21.GTVHacker.com

WIKI: http://www.GTVHacker.com

FORUM: http://forum.GTVHacker.com

BLOG: http://blog.GTVHacker.com

IRC: irc.freenode.net #GTVHacker

Follow us on Twitter:

@GTVHacker