# Hacking The Google TV

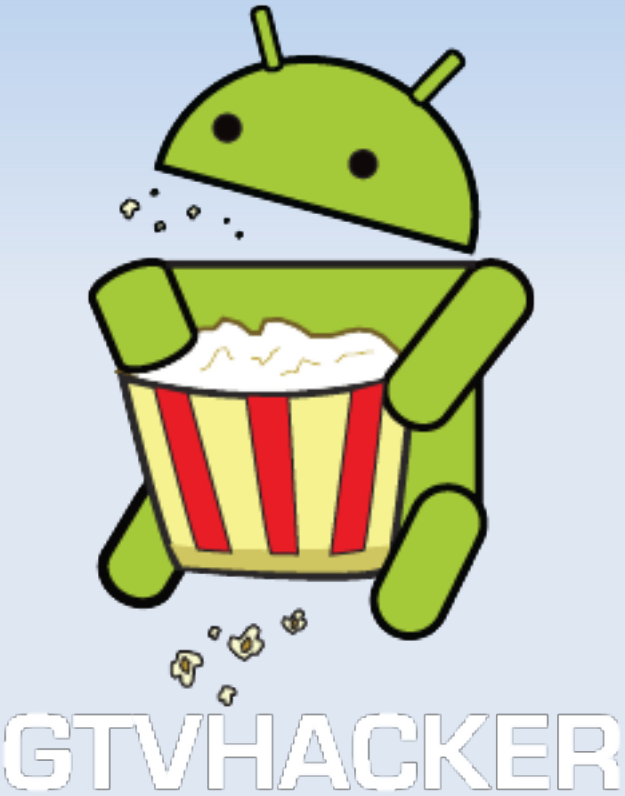

Presented by: Amir “zenofex” Etemadieh, CJ “cj_000” Heres,
Tom “tdweng” Dwenger, and Dan “bliss” Rosenberg

GTVHacker

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# 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.
- GTVHacker team won $500 bounty for being the first to root the Google TV.

# Team Members

The GTVHacker team officially consists of 6 members:

- AgentHH – First human outfitted with metal legs.
- cj_000 - Destroyer of words
- Gynophage – German rockstar reverse engineer
- [mbm] - known for founding the Open-WRT project and tossing 251 children down a well
- Tdweng – software developer turned super hero.
- Zenofex – [redacted]

With special guest:

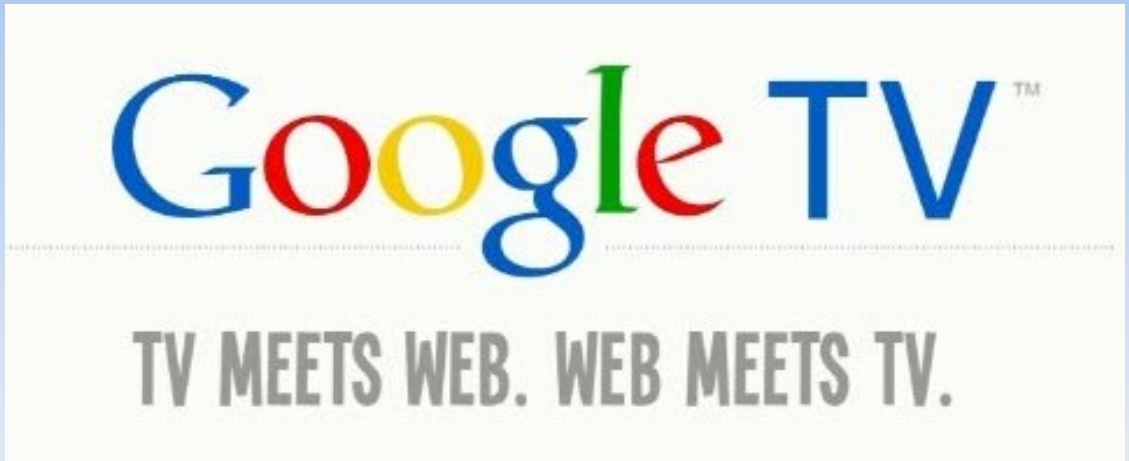
- Bliss - a vulnerability researcher who takes sick pleasure in exploiting anything with a CPU. He once punched an Android in the face.

GTVHacker

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# Google TV: What is it?




- Google TV is a platform that bridges the gap between your TV and an Android device.
- Platform creates an overlay on television stream and also contains an IR transmitter to transmit to media center devices (cable box, TV, sound system).
- Device was originally released without the Android Market available but was eventually updated to include it.
- Platform receives Over-the-Air updates automatically from OEM manufacturer.
- Platform contains forked version of Chrome with all plug-ins and extensions other than Flash disabled.

# Android vs. Google TV

Although Google TV runs Android there are differences:

- The device has a Chrome browser out of the box which provides a fairly reliable and safe browsing experience
- The Gen 1 Google TV platform is currently the only x86 set of Android devices.
- Although the platform does have the Android market, the amount of actual applications available is far below that of the actual market.
- Due to the fact that some Android applications include native code, some applications are not able to run on the x86 chip-set.
- Unlike most Android devices, GTV devices are USB hosts requiring ADB to be used over the network and ADB is restricted to one white-listed IP.

# x86 vs ARM?



- Most commonly deployed boxes are x86
- Newest Google TV Devices are ARM based
- Devices by Sony, LG and Vizio (Availability is still limited)
- More on the ARM devices a bit later!

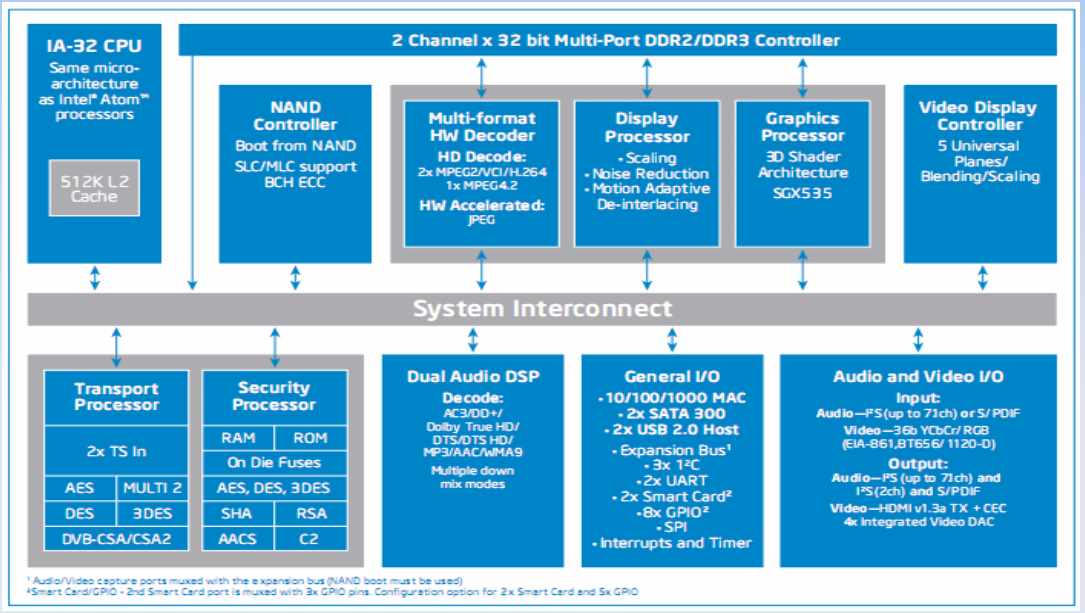
# GTV vs Content Providers


- From the initial release of the platform, the Google TV has been in a constant battle with the content providers.
- Content providers believed giving Google access to television programming advertising streams would strengthen Google's position in web advertising, as well as convince users to drop services like cable.
- Websites enforced checks by verifying the browser User-Agent as well as the Flash version string.

# Platform: x86 Android


- There are no other mainstream Android x86 devices.
- Architecture differences makes for a crippled marketplace.
- Code compiled for device can usually be compiled without the need for compiler toolchain.

# x86 / Gen 1 CPU


- Current generation of Google TV devices use an Intel CE41xx CPU.
- 45nm Atom core 1.2 Ghz with System-on-Chip (SoC).
- “On-die” security processor to handle DRM.

- Revue – CE4100
- Sony TV / Blu-Ray – CE4150

# Bootloader (Gen1)

- The bootloader for the CE41xx devices is known as the “Intel CEFDK” (Consumer Electronics Firmware Development kit).
- Bootloader is signed and signature is verified by security processor, beginning “chain of trust”.
- Intel supplies a stage 1 and stage 2 boot-loader in the SDK.
- Logitech uses both stages of CEFDK in its device.
- Sony uses Intel's stage 1 and it's own proprietary “NBL” for stage 2.

# “Chain Of Trust”

GTV platform utilizes a “Chain of Trust” boot


1) SoC decrypts and verifies signature of stage 1 CEFDK
2) Stage 1 CEFDK boots, checks signature, and decrypts Stage 2
3) Stage 2 boots and checks signature on Kernel
4) Kernel takes over
5) (Sony) Kernel SHA1 hashchecks init
6) (Sony) Init RSA verifies init.rc / init.(eagle/asura).rc

# Kernel Security


- Kernel requires modules to be properly signed before being inserted.
- All partitions except /data & /cache are marked as RO by the kernel.
- ADB shell only allows RW access to folders with “shell” permissions.
- Functions like ptrace are left out of the kernel.
- Access /dev/mem is restricted.
- Kernel is patched from all known public Android vulnerabilities.

# Current Devices

# Logitech Revue


- Released October 2010
- Full sized keyboard with built in touchpad
- Originally priced at $249 later reduced to $199 and finally $99
- Discontinued but still favoured by a majority of GTV users

# Logitech Revue Motherboard

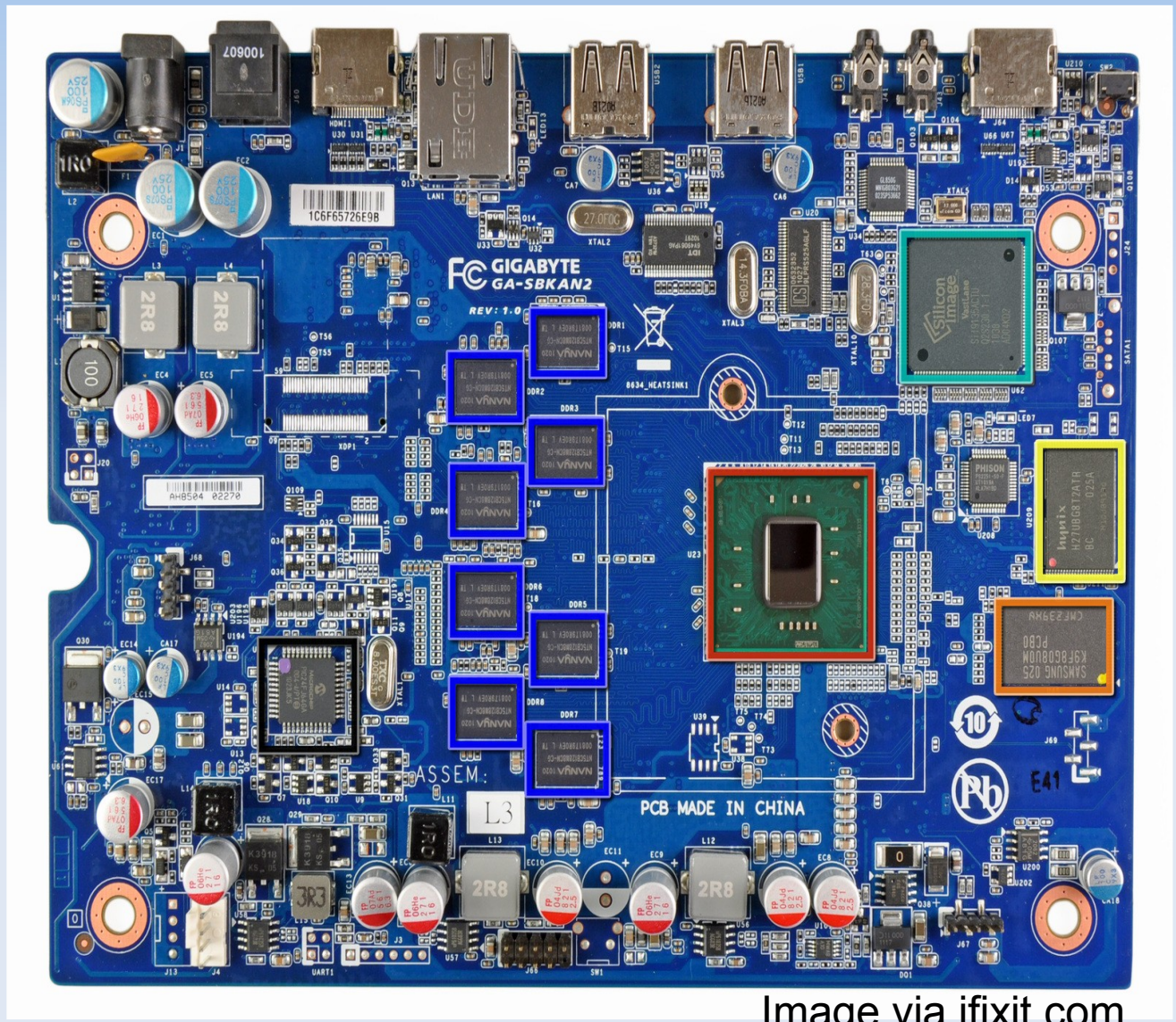



Image via ifixit.com

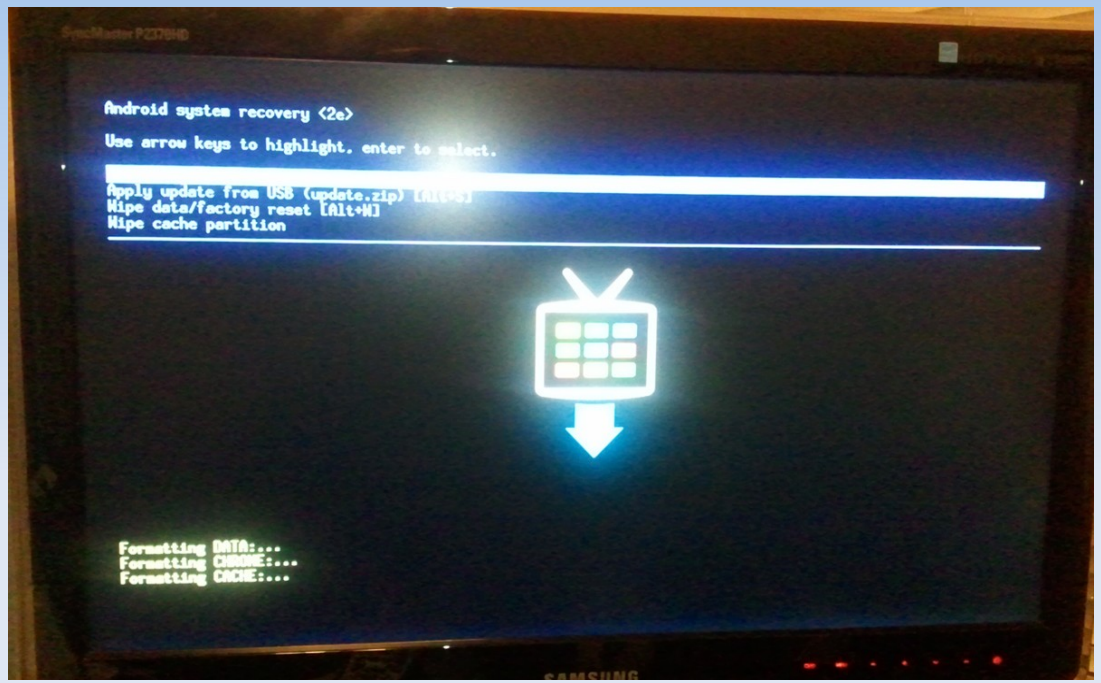
- UART1 - Console
- J3 - PICKIT2
- SW1 – Unused switch
- J20 – I2C
- J69 – USB
- SATA1 – SATA Header
- J24 – Unknown
- J13 – Power for SATA
- XDP1 – Intel XDP Debug Header

http://gtvhacker.com/pres/dc20.ppt

GTVHacker

DEFCON

# Revue: Recovery

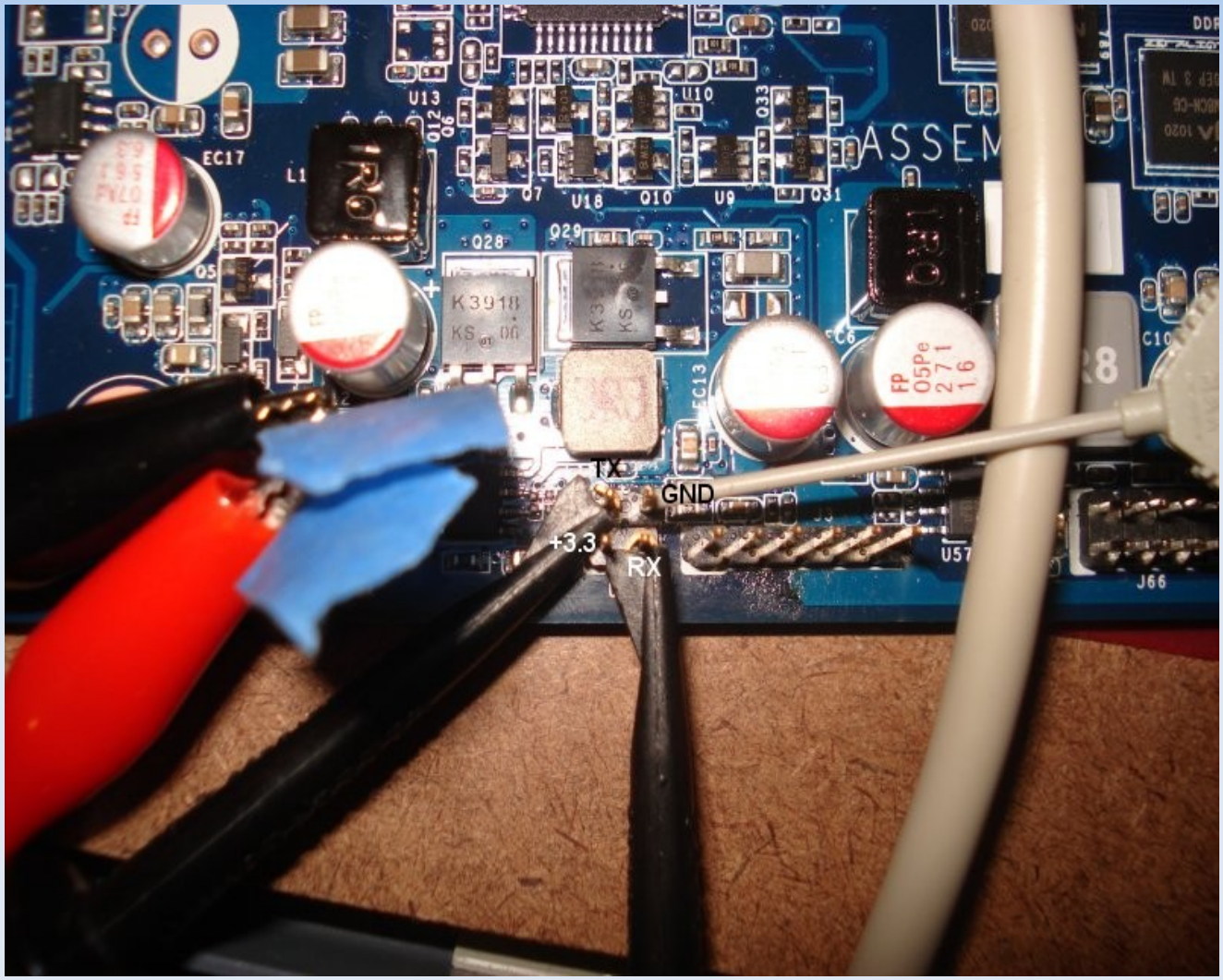



Recovery mode is an “Android 2e recovery” which is standard on many Android devices.

- Reboot
- Apply Update from USB (update.zip)
- Wipe data/factory reset
- Wipe cache partition

All update files provided are RSA verified before the box attempts installation.

# UART On The Revue




- First root on the Google TV Platform.
- Required a virgin Revue.
- Still works on newly purchased Revues.
- Soldering to four pads on the Revue and booting into recovery mode.
- Method allowed for Read/Write access to File System.

GTVHacker

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# UART On The Revue


- Created a manual update process that mirrored Google's but did not perform any of the signature checks.
- Continued to release modified updates which included an ADB running as root as well as our first attempt at a content provider bypass.

# First “Content Provider Bypass”

Bypassing Hulu/CBS/NBC/ABC's browser/flash checks was relatively easy and could be done quickly with a hex editor and RW /system access. All that was required was a simple change 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
```

Changing one letter in the flash version string as well as changing the browser user agent (which can be done directly from the box in Chrome's settings) will allow a user to watch normally restricted content.

# Honeycomb Suprises: Message from Logitech?

```
00 00 00 00 00 00 e0 41  20 40 67 74 76 68 61 63  |.......A @gtvhac|
6b 65 72 73 20 70 62 61  74 65 6e 67 68 79 6e 67  |kers pbatenghyng|
76 62 61 66 20 76 73 20  6c 62 68 65 20 65 72 6e  |vbaf vs lbhe ern|
71 76 61 74 20 67 75 76  66 20 00 00 63 79 72 6e  |qvat guvf ..cyrn|
66 72 20 63 62 66 67 20  6e 20 61 62 67 72 20 62  |fr cbfg n abgr b|
61 20 6c 62 68 65 20 73  62 65 68 7a 20 67 62 20  |a lbhe sbehz gb |
79 72 67 20 7a 72 20 78  61 62 6a 20 3b 29 00 41  |yrg zr xabj ;).A|
67 65 6e 74 48 48 00 5a  65 6e 6f 66 65 78 00 63  |gentHH.Zenofex.c|
6a 5f 30 30 30 00 63 72  61 69 67 64 72 6f 69 64  |j_000.craigdroid|
00 5b 6d 62 6d 5d 00 72  65 73 6e 6f 00 74 64 77  |.[mbm].resno.tdw|
65 6e 67 00 74 61 74 75  6e 67 34 00 63 63 64 74  |eng.tatung4.ccdt|
```

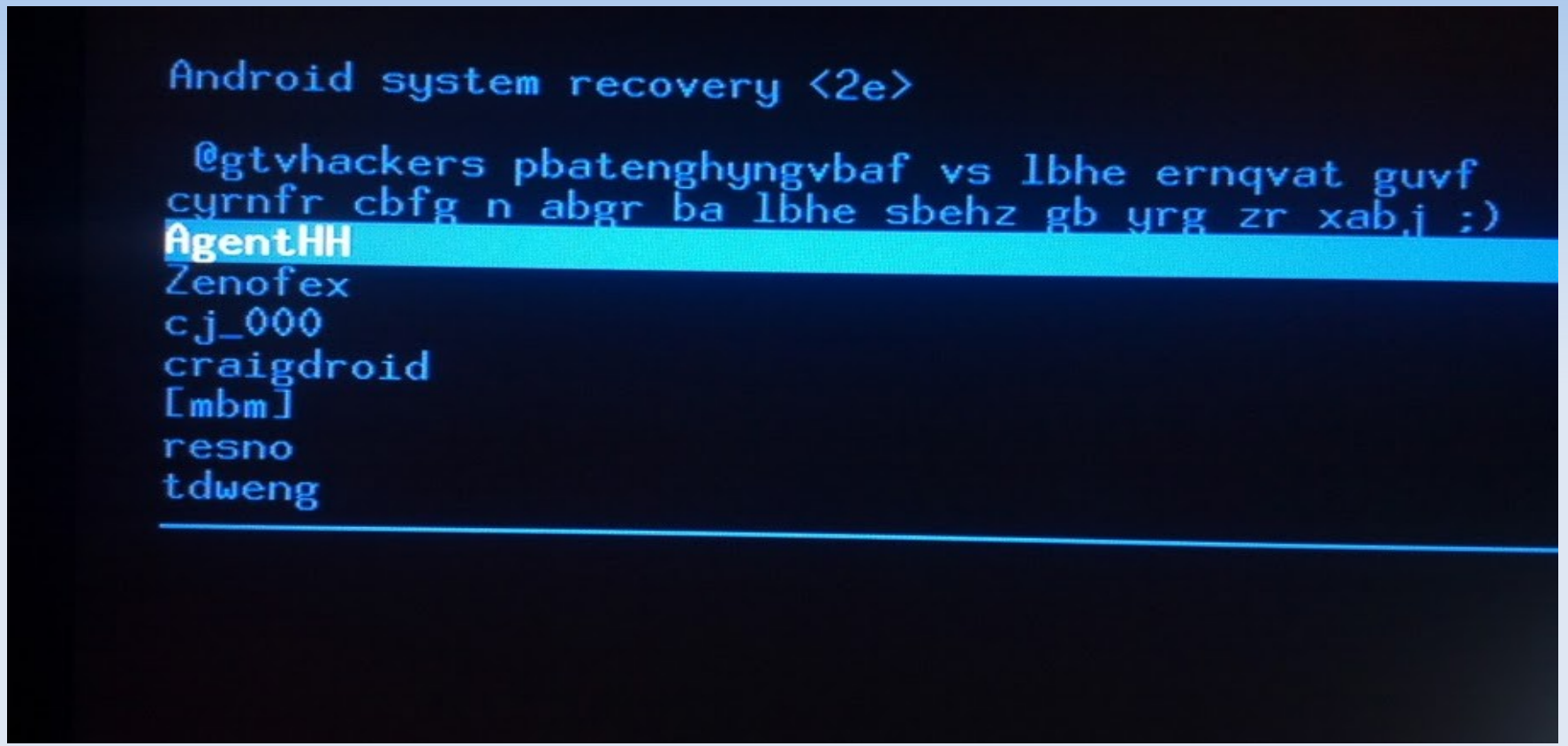
- Logitech removed the recovery menu and replaced it with a message to the GTVHacker team.
- Removed functionality to install manual updates therefore removing a user's ability to recover other than via the automatic process of erasing /cache and /data.
- The message was encoded in a ROT13 cypher.
- Each of the current GTVHacker team members' names were listed as no longer functioning recovery menu items.

**GTVHacker**

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# Honeycomb Suprises: Message from Logitech?




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

# Flash Sabotage: Revue

Getting a secret message from Logitech was awesome.
Having them remove the recovery menu functionality was not.

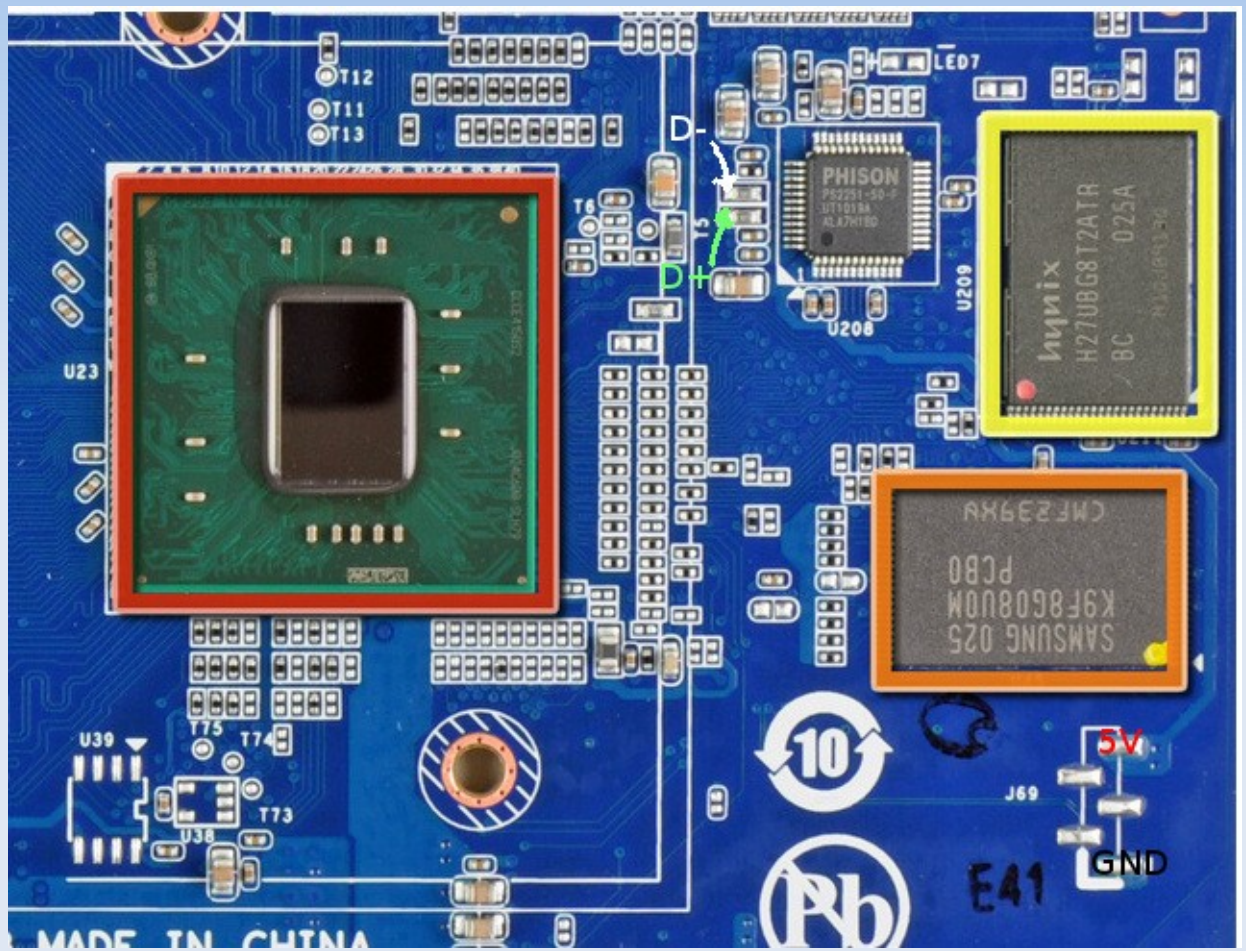
So we needed a way to play with the update functionality of the box...

The OTA updater writes to /cache/recovery/command, which uses the following syntax:

--update_package=CACHE:/somefile.zip

Now if only we had a way to write to cache...

# Flash Sabotage: Revue




- /cache and /chrome are EXT3 partitions stored on NAND flash.
- Luckily, that flash is connected to the Revue via a USB Controller.
- It's a flash drive!
- We can tap the data lines and stick our own flash drive in line.

# Revue Kernel Exploit

Revue root kernel exploit

To be added

# Revue Module Signing Exploit

Revue RSA kernel module signing bypass

To be added

# Sony Devices (Gen1)


Blu-Ray Player (NSZ-GT1)

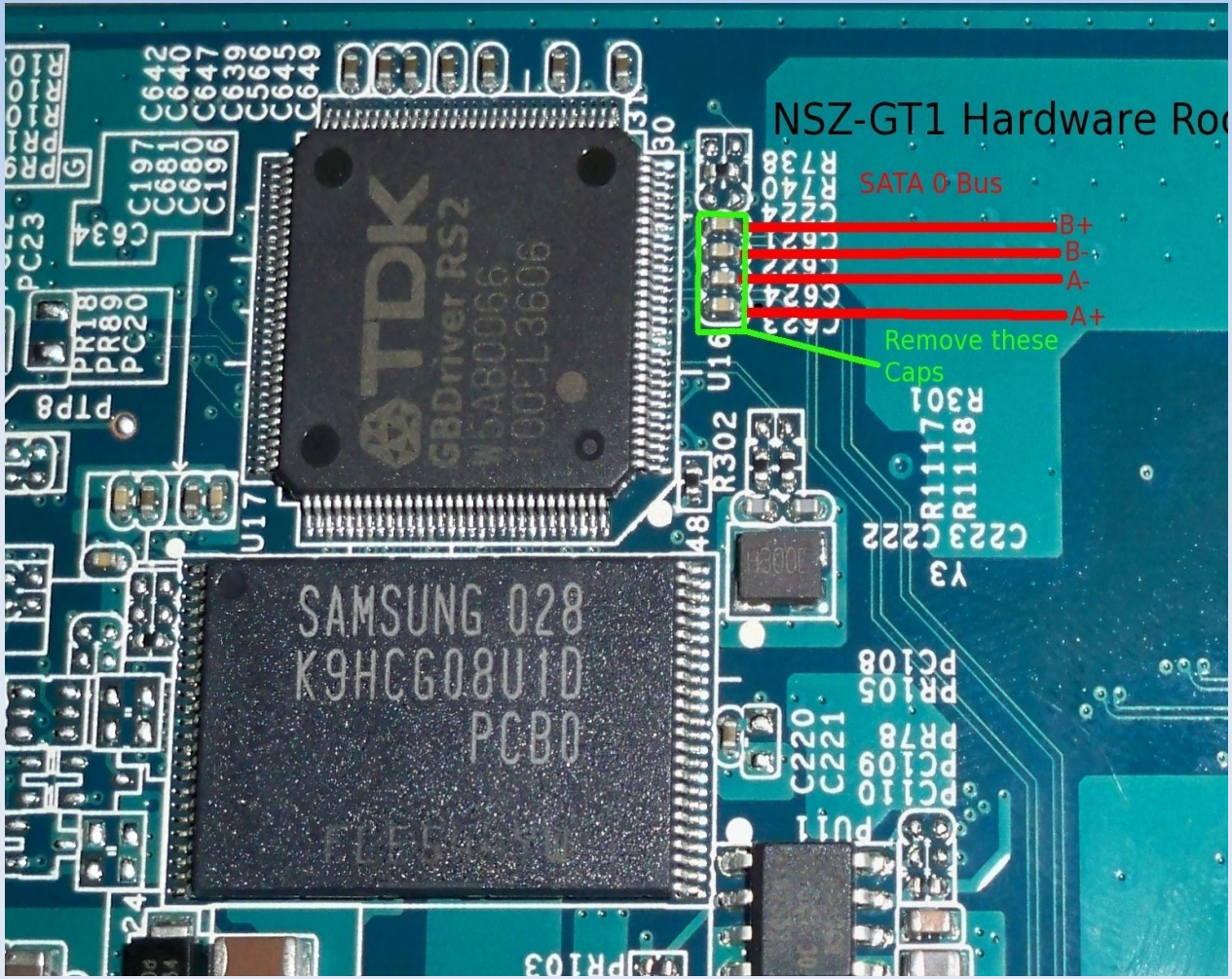

Television 24” - 46” (NSX-#GT1)

GTV hardware is nearly identical, other than the obvious differences.

# NSZ-GT1: Motherboard

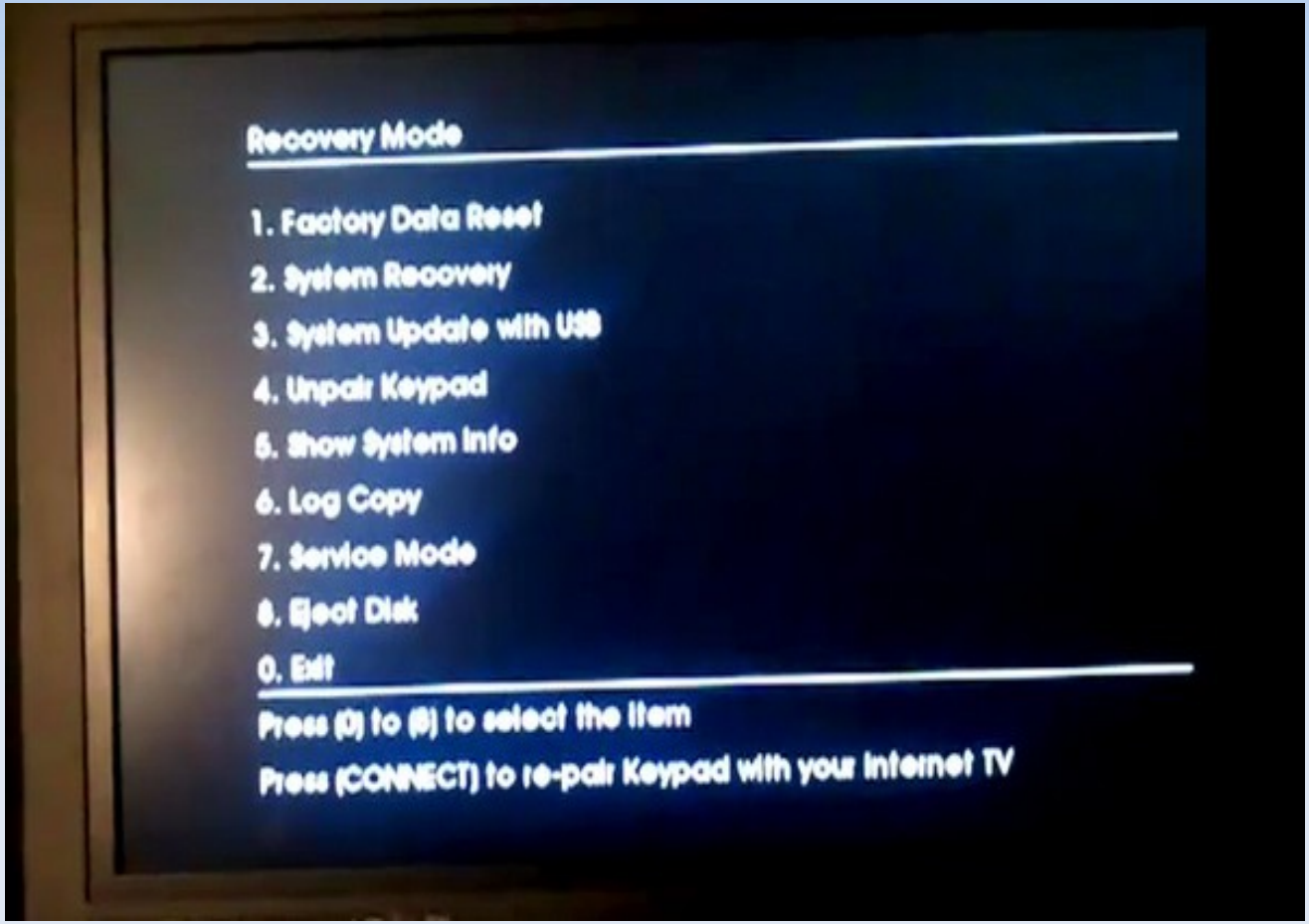
- Bulkier than the Revue.
- Built like a Sony.
- Populated debug pads!
- Contains a faster processor – CE4150 @ ~1.7GHz.

# SATA Sabotage: Sony




- Internal SSD via SATA
- GBDriver RS2 AES encrypts all data on NAND flash.
- ATA Password
- Sony stored all data on SSD, except bootloader and kernel.
- Risky procedure. Small points.
- Able to “redirect” SATA bus to our own device, which we had RW access to externally.
- Used this to downgrade to old SW versions, to look for flaws.

# Sony GTV: Recovery




- Far more interesting than that of the Revue.
- Like the Revue, has a similar “Update from USB” feature.
- Nearly entire backend is done through a series of scripts. Not standard Android, so no debug log is left behind. Though not impossible thanks to the UART.
- Sony updates are RC4 encrypted.

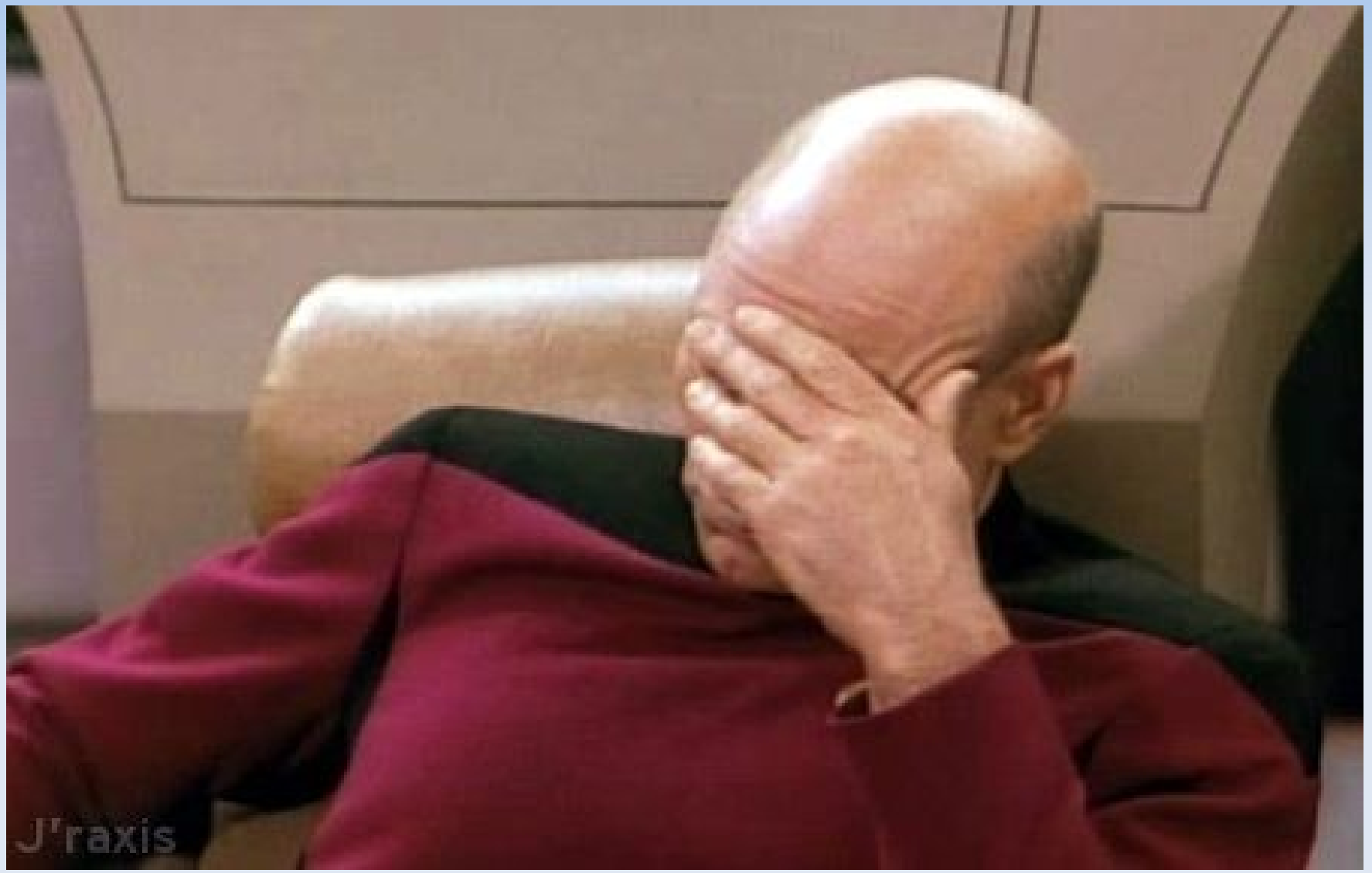
# Sony Google TV
# Command Execution Through Recovery

Can you spot the problem here?

**ls /tmp/mnt/diskb1/package_list_*.zip | head -1 | grep "package_list_"**

**/bin/sony/check_version.sh $1**

# Sony Google TV
# Command Execution Through Recovery




GTVHacker

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# Sony Google TV
# Command Execution Through Recovery

The exploit was simple, a package with a command:

package_list_;cd tmp; cd mnt; cd diskb1; sh t.sh; .zip

/package-updater.sh -l 0 -p /tmp/mnt/diskb1/package_list_;cd /tmp;cd /mnt;cd /diskb1;sh t.sh;.zip

The command above involved a t.sh bash script (to meet filename size limitations) which spawned a shell over UART and telnetd.

From there we proceeded to dump the recovery file system.

# Sony Google TV
# Command Execution Through Recovery

Unfortunately this exploit was patched in the 7/2011 update.

*“It's not exactly what we'd call a easy jailbreak, seeing as how it requires a soldering iron, a NAND format procedure, and a Logitech Revue that's never even been powered on, but it looks like it is possible to root a Google TV box after all.” - engadget.com*

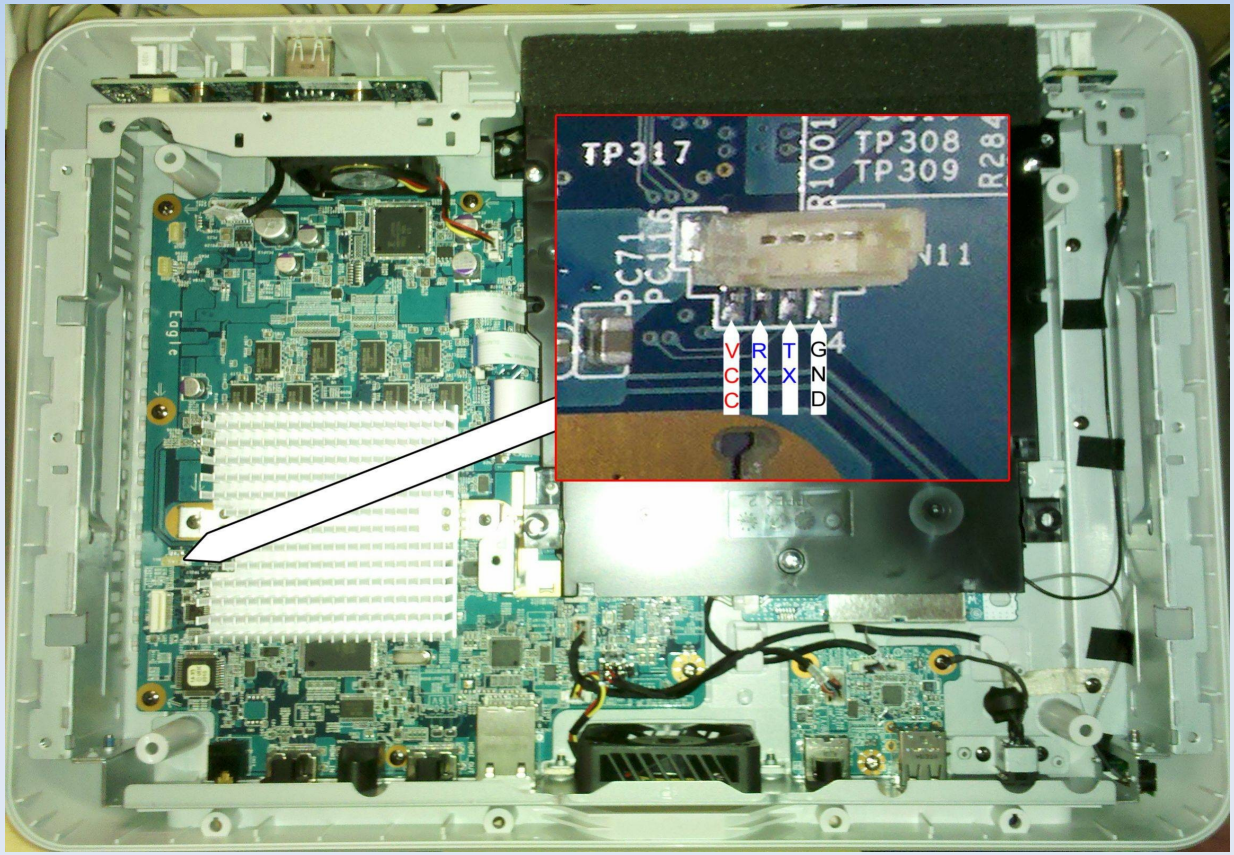
That was said about 4 large pads for the Revue.

Needless to say, this was not a viable option for the public.

# Sony Google TV UART

- Active UART line (output only)
- After initial hack - achieved root console in Linux.
- Memory dump shows existence of “NBL” - an extra step after Intel's initial bootloader.
- Mashing escape over UART at start-up brings us to a “Password:”
- Password found after reversing NBL areas of memory:

console_ON




GTVHacker

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# Sony Google TV
# UART / Bootloader

- NBL options included loading files into memory, and executing from internal flash or network via TFTP.

- Insecure booting features were disabled on production units.

- NBL Utilized signature and hash checks similar to the normal start-up mode.

Remember that exploitable recovery version?

# Sony Google TV
# UART / Bootloader

```
Successful patch applied!
SEC FW: SEC ready
SEC FW: firmware module sent to SEC for authentication and load
Successful firmware download!
SEC FW: firmware version valid: 2.1.0.6.

Verify stage2 PASS
Intel(R) Consumer Electronics Firmware Development Kit (Intel(R) CEFDK)
Copyright (C) 1999-2010 Intel Corporation. All rights reserved.
Build Time (06/11/10 16:00:37).
Loading 8051 MicroCode at 0x80000
SATA 0: BTVSSD01 - 8G
SATA 1: SONYBDP-410 - 0G

Password:
NBL BTV-EC 5.7.1 (base: 5.0-BTV_20100707) <built 11:14:57, 07/30/10 JST>
Machine: EAGLE (i386/sodaville/btv)
RAM: 0x00100000-0x02000000 available
hda: BTVSSD01
    (C/H/S = 30720/16/32, Total 15728640 sectors)
NBL> boot -f net:tftp:/vmlinux_recovery.trf -c "root=/dev/ram0 console=ttyS0,115200 mem=exactmap memmap=1M$0 memmap=199M@1M"  -initrd net:tftp:/initrd.trf
i8254x: Ethernet address: 54:42:49:d4:66:d2
i8254x: Link is up, 100Mbps Full Duplex
dev_net.c:net_getparams: enable bootp because IP==0
bootp: 'myip' is 192.168.1.121
'serverip' is 192.168.1.148
bootp: mask: 255.255.255.0
net_open: client addr: 192.168.1.121
net_open: subnet mask: 255.255.255.0
net_open: net gateway: 192.168.1.2
net_open: root addr: 192.168.1.148
net_open: server addr: 192.168.1.148
net_open: server path: /
net_open: file name: vmlinux_recovery.trf
TRF file is loaded : start = 0x00100000, length = 0x00396086
i8254x: Ethernet address: 54:42:49:d4:66:d2
i8254x: Link is up, 100Mbps Full Duplex
```

GTVHacker

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# Sony Google TV
# UART / Bootloader

Booting via TFTP allowed us to set kernel args.

boot -f net:tftp:/vmlinux_recovery.trf -c "root=/dev/ram0 console=ttyS0,115200"
-initrd net:tftp:/initrd.trf

Booting via TFTP however kept the internal SSD ATA locked.

The good news was that when that recovery booted to a locked ATA, the box dropped us into a console!

# Sony Google TV
# UART / Bootloader

Exploitable Recovery:

- System boot binaries stored on flash at /dev/Glob_Spectraa2
- ATA was locked, flash was not! Drivers just needed to be loaded.
- Replaced new recovery on flash with the old, exploitable version.
- Now we had an exploitable recovery!
- Wait for the rumored 3.2 release in late September to release exploit
- Google and Sony were slow – it was December.

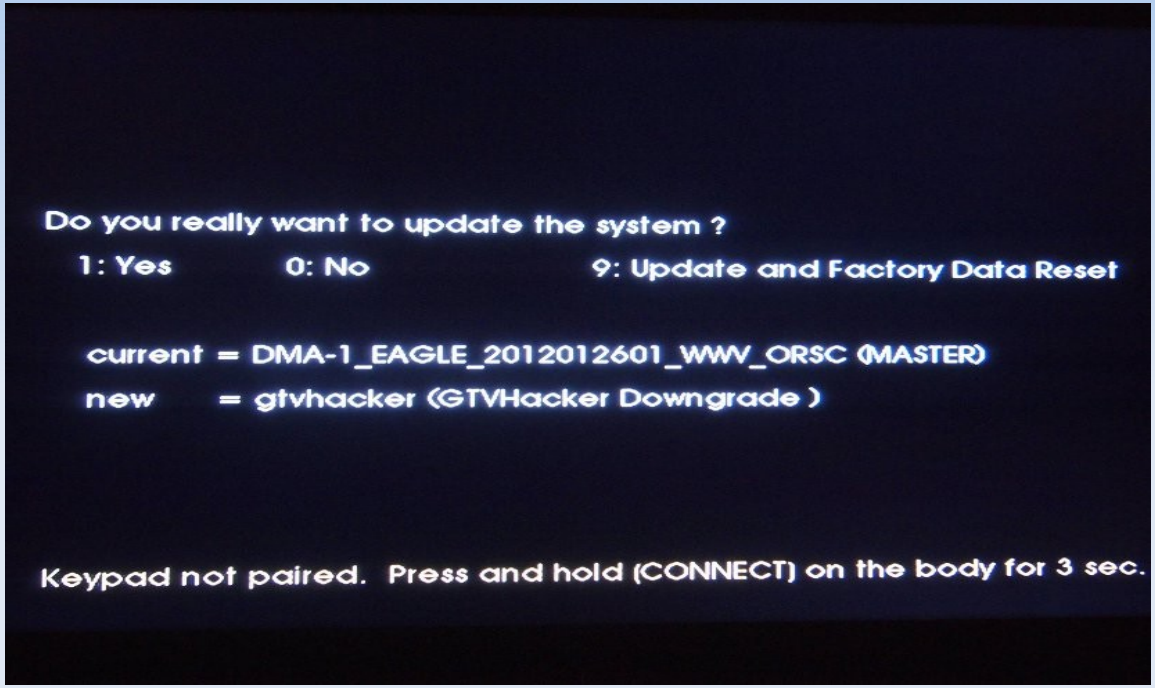
# Sony Google TV
# Downgrade via USB (nodev)

- Come the 3.2 release in December, we did not want to let on about the bootloader password being found. So, two weeks of intense bug finding was started.

- We found a few bugs, but not what we needed for privileged code execution

- However, we got to really, really know the update process...

# Sony Google TV
# Downgrade via USB (nodev)



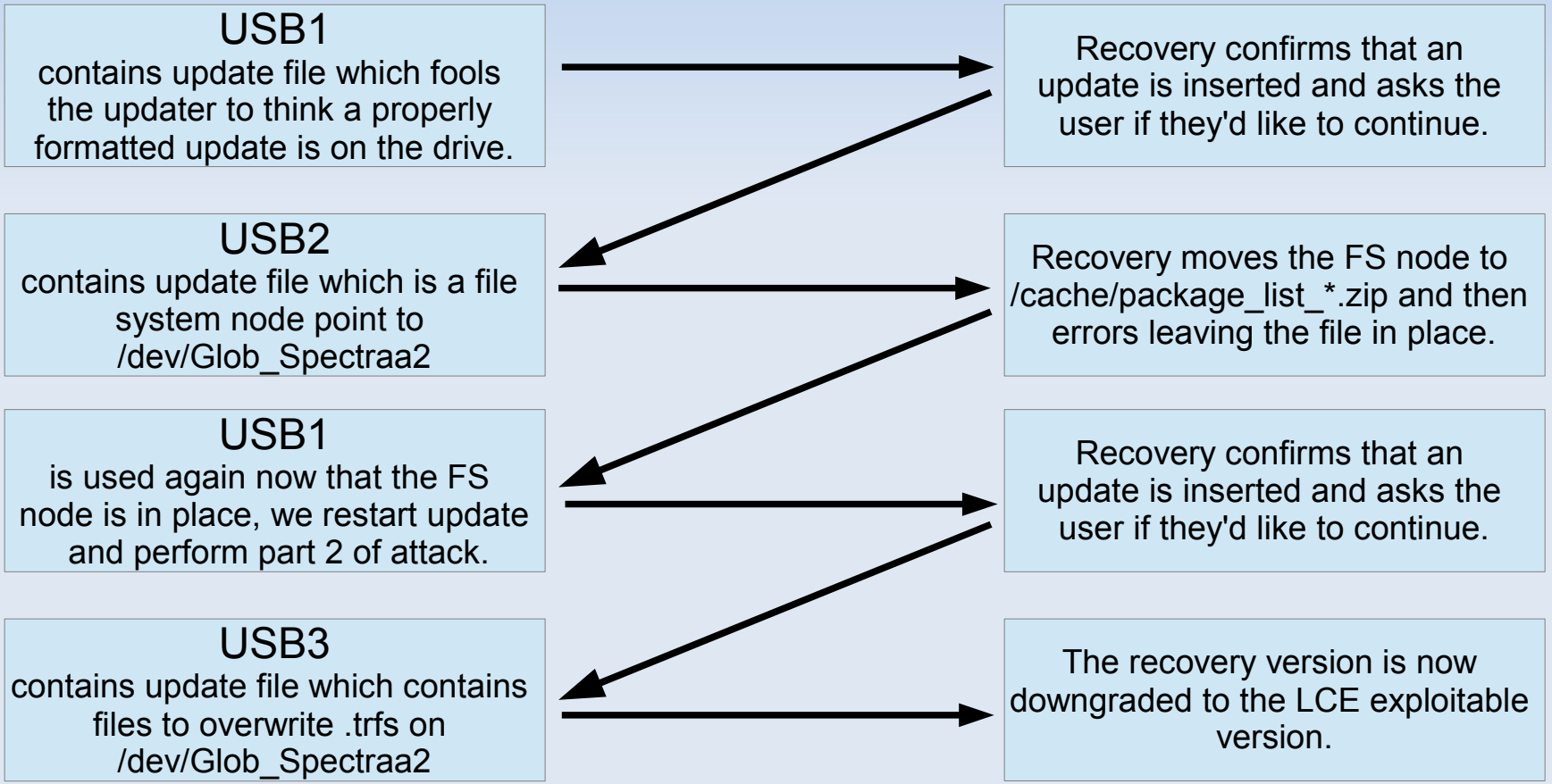

- Recovery mounts USB to /tmp/mnt/diskb1
- Looks for package_list_*.zip
- Passes this to package_updater.sh
- package_updater.sh then copies the file to /cache
- package_updater then unzips build.prop, and displays to the user

If the update is accepted, it's copied again to /cache

I'm sure they checked to see if there was a destination file...

# Sony Google TV
# Downgrade via USB (nodev)

The Sony recovery mounted ext2/3 partitions with no mount parameters meaning a block device on the USB could allow us to write to a device node as root.



# Sony Google TV
# Downgrade via USB (nodev)

Assuming the downgrade went correctly, use LCE recovery exploit.

Exploit:

- Re-partitions internal SSD
- Copies /boot to a new partition.
- Edits initial /boot to include kexec files.
- Hijacks initial boot process to call kexec.

```
mkfs.ext3 /dev/sda6 > /dev/null
mkfs.ext3 /dev/sda7 > /dev/null
mkfs.ext3 /dev/sda8 > /dev/null
sleep 5
echo "mkfs done"

/tmp/busybox mkdir /tmp/mnt/sda1
/tmp/busybox mkdir /tmp/mnt/sda5
/tmp/busybox mkdir /tmp/mnt/sda6
/tmp/busybox mkdir /tmp/mnt/sda7
/tmp/busybox mkdir /tmp/mnt/sda8
/tmp/busybox mkdir /tmp/mnt/spectra2

/tmp/busybox mount -text3 /dev/sda1 /tmp/mnt/sda1
/tmp/busybox mount -text3 /dev/sda5 /tmp/mnt/sda5
/tmp/busybox mount -text3 /dev/sda6 /tmp/mnt/sda6
/tmp/busybox mount -text3 /dev/sda7 /tmp/mnt/sda7
/tmp/busybox mount -text3 /dev/sda8 /tmp/mnt/sda8
/tmp/busybox mount -tvfat /dev/Glob_Spectraa2 /tmp/mnt/spectra2

echo "Devices Mounted: "
cat /proc/mounts

#copy initial system to new area, we don't need a reboot loop
/tmp/busybox cp /tmp/mnt/sda1/sbin/e2fsck-bak /tmp/mnt/sda1/sbin/e2fsck
/tmp/busybox cp -R /tmp/mnt/sda1/* /tmp/mnt/sda8/

#copy our rebooter stuff (kexec, new kernel etc) to sda8
/tmp/busybox cp -R /tmp/mnt/diskb1/copy/reboot/ /tmp/mnt/sda8/

#Yeah, we need a lot of busybox would help to install it, but oh well - next version!
/tmp/busybox cp /tmp/mnt/diskb1/copy/busybox /tmp/mnt/sda8/bin/busybox

/tmp/busybox cp /tmp/mnt/diskb1/copy/busybox /tmp/mnt/sda1/bin/busybox

#/tmp/busybox cp /tmp/mnt/diskb1/copy/su /tmp/mnt/sda8/bin/su
#/tmp/busybox chmod 4755 /tmp/mnt/sda8/bin/su

# Allthepermissions.png
/tmp/busybox chmod 777 /tmp/mnt/sda8/bin/busybox

/tmp/busybox cp /tmp/mnt/sda1/sbin/e2fsck /tmp/mnt/sda1/sbin/e2fsck-bak
/tmp/busybox rm -rf /tmp/mnt/sda1/sbin/e2fsck
/tmp/busybox cp /tmp/mnt/diskb1/copy/e2fsck /tmp/mnt/sda1/sbin/e2fsck

/tmp/busybox rm -rf /tmp/mnt/sda8/default.prop
/tmp/busybox cp /tmp/mnt/diskb1/copy/default.prop /tmp/mnt/sda8/

/tmp/busybox rm -rf /tmp/mnt/sda8/init.asura.rc
/tmp/busybox cp /tmp/mnt/diskb1/copy/init.asura.rc /tmp/mnt/sda8/

/tmp/busybox rm -rf /tmp/mnt/sda8/init.rc
/tmp/busybox cp /tmp/mnt/diskb1/copy/init.rc /tmp/mnt/sda8/

/tmp/busybox rm -rf /tmp/mnt/sda8/init
/tmp/busybox cp /tmp/mnt/diskb1/copy/init /tmp/mnt/sda8/

/tmp/busybox rm -rf /tmp/mnt/sda5/build.prop
/tmp/busybox cp /tmp/mnt/diskb1/copy/build.prop /tmp/mnt/sda5/

/tmp/busybox mv /tmp/mnt/sda5/etc/security/otacerts.zip /tmp/mnt/sda5/etc/security/otacerts-bad.bad

/tmp/busybox rm -rf /tmp/mnt/spectra2/vmlinux.trf
/tmp/busybox cp /tmp/mnt/diskb1/copy/vmlinux.trf /tmp/mnt/spectra2/

/tmp/busybox rm -rf /tmp/mnt/spectra2/sony_logo_480.bmp.gz
/tmp/busybox cp /tmp/mnt/diskb1/copy/sony_logo_480.bmp.gz /tmp/mnt/spectra2/

#Flash Player - mutate the ID String
cp /tmp/mnt/diskb1/copy/mutate /tmp/mutate
```

# Sony Google TV
# Unsigned Kernels

*“kexec (kernel execution) is a mechanism of the Linux kernel that allows "live" booting of a new kernel "over" the currently running kernel. “ ~ Wikipedia*

- Kexec is normally built into the kernel, so we opted to build it as a kernel module.

- Kexec allows us to boot the system, have it kick over after in less than 1 second, and load our unsigned kernel.

But what about that init hash, and those RSA signatures?

# Sony Google TV
# Unsigned Kernels

- Chain of Trust needed to be broken
- kexec had to to be called before the platform's security firmware was loaded.
- Where do we attack?
- /bin/e2fsck
- / is mounted from sda1, on the SSD, that we can now write to

# Sony Google TV
# Unsigned Kernels

/bin/e2fsck was replaced with a script which:

- Mounted /system
- insmod our kexec modules
- kexec to load our new kernel

Our new kernel, apart from featuring no hash on init, had a few other tweaks:

- no initd hash
- no signed init.rc
- no signed init.(eagle/asura).rc
- modified init.rc
- modified init.(eagle/asura).rc
- modified default.prop
  - ro.secure=0
  - ro.debuggable=1

# Sony Google TV
# Content Provider Bypass

But wait – there's more!

Our update script pulled Chrome's Flash player and mutated the Flash plug-in string randomly per each install.

Since each box has a unique ID, content providers will have a harder time blocking streaming content for Google TV users.

# Google TV
# “Future” / ARM Devices

In the last few months we've seen a release of the second generation of Google TV devices, all of which are ARM:

Sony NSZ-GS7 – Network Streamer
Sony NSZ-GP9 – Blu-Ray Player *unreleased*

Vizio VAP430 (CoStar) – Network Streamer *unreleased*
Vizio VBR430 – Blu-Ray Player *unreleased*
Vizio R3D*0VS (42/47/55/65) – Google TV *unreleased*

LG 47/55G2 (LMG620) – Google TV

# Google TV
# Sony - ARM Devices



- The Sony ARM devices feature a Marvell 88DE3100 SoC, which has a 1.2GHZ Dual Core Processor.

- The Blu-Ray variant should be close to identical specs wise, but with a Blu-ray drive, and a BD playback app.

- Sony has yet to branch off into TV integration, as they may have jumped the gun the first time around.

# Google TV
# Vizio - ARM Devices


- The Vizio ARM devices, like the Sony's feature a Marvell 88DE3100 SoC, which has a 1.2GHZ Dual-Core processor.

- Again, the Blu-Ray variant should be close to identical specs wise, but with a Blu-ray drive, and a BD playback app.

- Multiple devices, a streamer, BD player, and integrated TV.

- Hey, you – guy on stage. Is the streamer out yet?

# Google TV
# LG - ARM Devices


- LG Google TV's are a bit more mysterious.
- 47” & 55” (G2 / LMG620)
- Mostly, there have been few purchases, and at $1200 each, a bit out of our price ranges!
- Dual Core ??? MHZ processor
- Anyone care to donate one?

# GTVHacker Timeline

| Date | Event |
|---|---|
| 12/2010 | Logitech UART found (and live) |
| 1/2011 | Root package released (content bypass) |
| 7/2011 | Sony (Blu-ray) unit acquired |
| | Sony unit rooted (SATA modification) |
| | Sony recovery command execution found |
| | Software root method found |
| | Sony update encryption keys found, reversed, decrypted |
| 8/2011 | Revue 3.1 "Honeycomb" leaked |
| 9/2011 | Sony 3.1 Released |
| | Sony TV acquired |
| | Sony TV rooted |

# GTVHacker Timeline

| Date | Event (Continued) |
|---|---|
| 10/2011 | Sony bootloader shell found/downgrade achieved |
| 11/2012 | kexec ported as module to x86, unsigned kernels for Sony (saving for 3.2 rls) |
| 12/2012 | 3.2 for Sony released |
| 1/2012 | Sony nodev recovery downgrade released |
| 1/2012 | Sony exploit package released (unsigned kernels) |
| 3/2012 | Revue signed module exploit achieved (needed root privileges) |
| 4/2012 | Logitech Revue kernel exploit (awaiting 3.2 release) |
| 5/2012 | Revue 3.2 Released |
| 6/30/2012 | NSZ-GP7 Acquired |
| 6/30/2012 | NSZ-GP7 Root Exploit |

# Sony NSZ-GP7

- Newest Sony device
- Released this month
- Tear down posted at GTVHacker.com
- CN2000 looks familiar!


http://gtvhacker.com/pres/dc20.ppt

GTVHacker

DEFCON

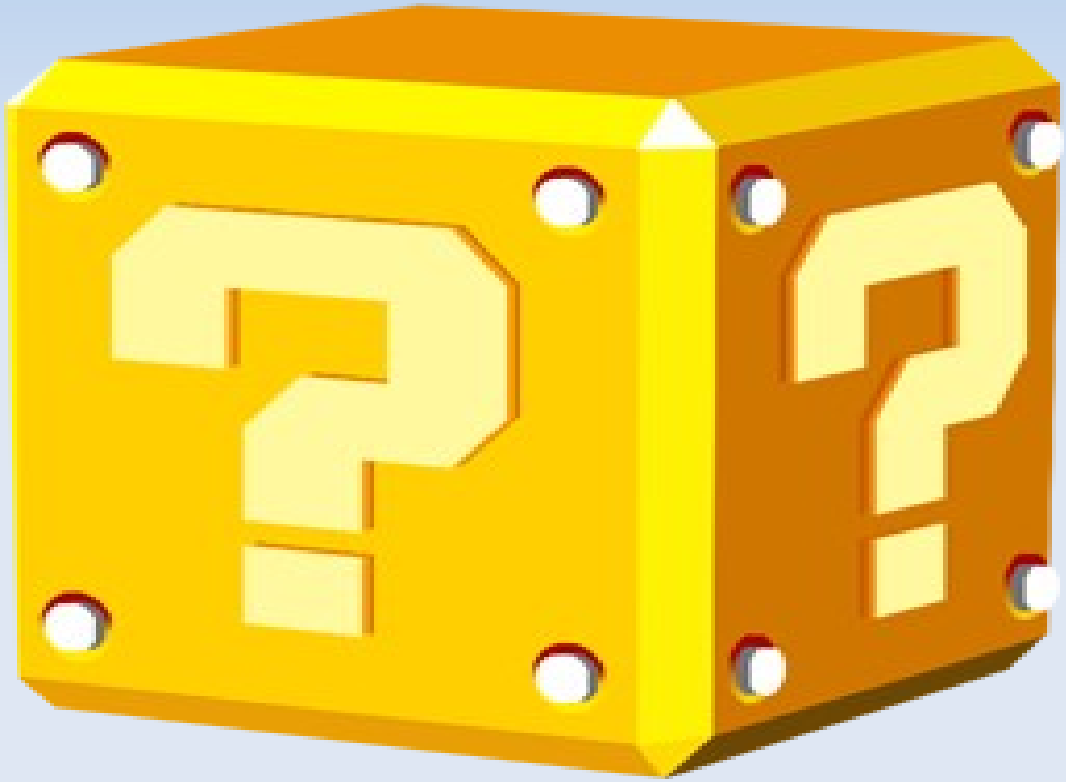
# NSZ-GP7 Root Demo

- Noticed that last bit on the time line? Yeah.
- We gained root access on 6/30, and proceeded to explore
- Our goal is to get unsigned kernels running before a release, which may or may not be done already (you, with the microphone?)

# NSZ-GP7 Root Demo

Demo

# Questions?



GTVHacker

http://gtvhacker.com/pres/dc20.ppt

DEFCON

# Thank you!

**More information can be found at:**

http://www.GTVHacker.com/

http://forum.GTVHacker.com/

http://blog.GTVHacker.com/