Mess with the best, die like the rest (mode)

Volodymyr Pikhur

@vpikhur

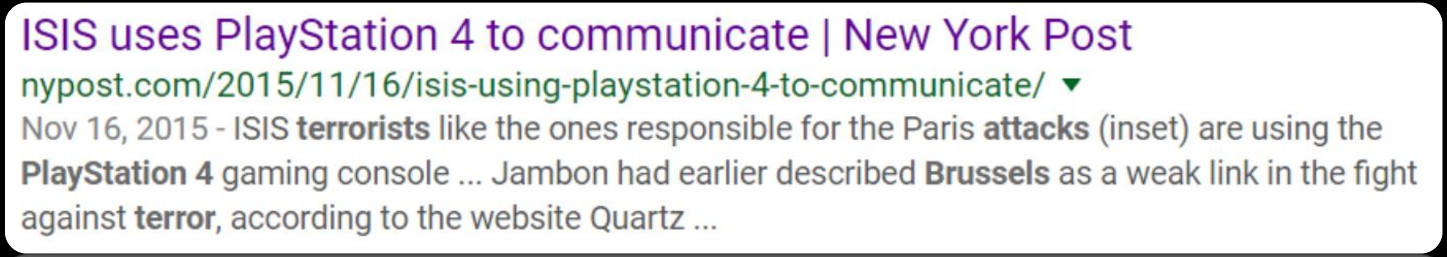
About

- **Been doing RE for more than 15 years**
  - Privately wrote multiple tools for deobfuscation and binary analysis, PE unpackers, software VM disassemblers/decompilers, etc.
  - Kernel and hypervisor based security exploitation
  - First time public speaker
- **Past 5 years been learning hardware**
  - Starting from basics Firmware, SPI, UART, etc.
  - Silicon decapsulation, fault injection
  - Past year+ been working on HW for side-channel analysis.

Why doing this?

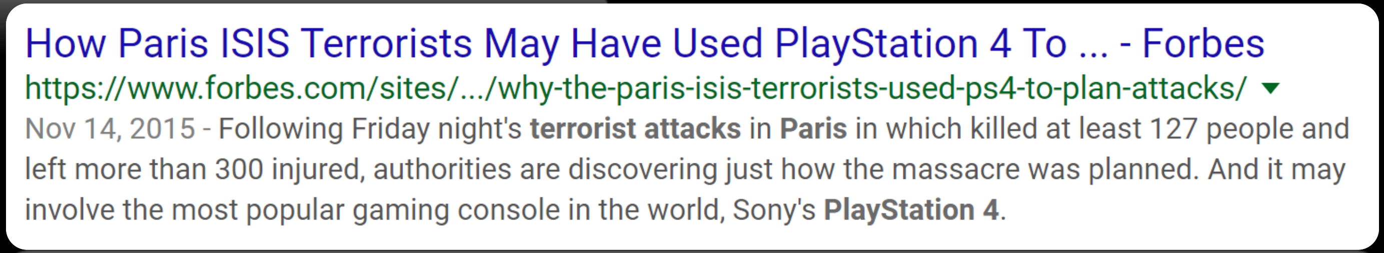
- Learning and a challange.
- Hardware and silicon isn’t your magic black box.
- Sony has no bug bounties.
- I’ve been sitting on this for 2 years.

Why presenting here?

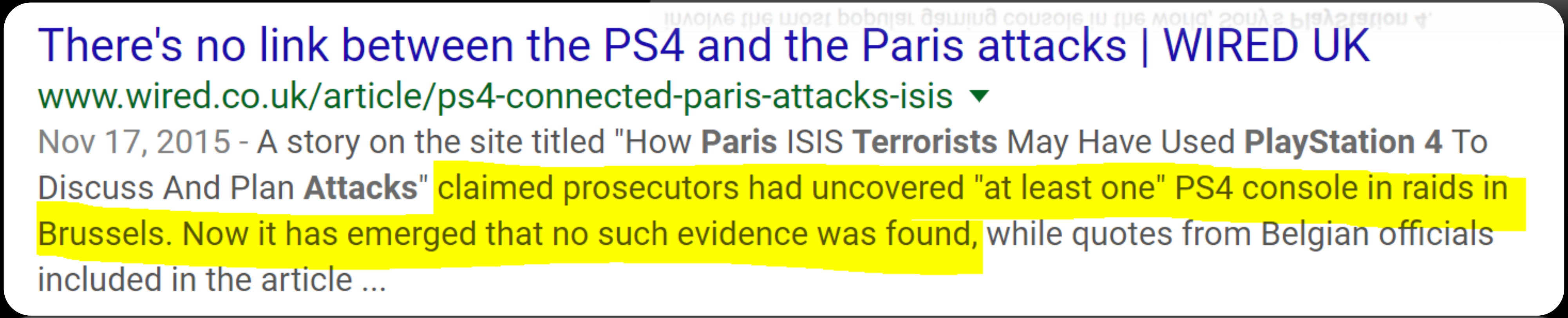


ISIS uses PlayStation 4 to communicate | New York Post
nypost.com/2015/11/16/isis-using-playstation-4-to-communicate/ ▾
Nov 16, 2015 - ISIS **terrorists** like the ones responsible for the Paris **attacks** (inset) are using the **PlayStation 4** gaming console ... Jambon had earlier described **Brussels** as a weak link in the fight against **terror**, according to the website Quartz ...

FAKE NEWS!



How Paris ISIS Terrorists May Have Used PlayStation 4 To ... - Forbes
https://www.forbes.com/sites/.../why-the-paris-isis-terrorists-used-ps4-to-plan-attacks/ ▾
Nov 14, 2015 - Following Friday night's **terrorist attacks** in **Paris** in which killed at least 127 people and left more than 300 injured, authorities are discovering just how the massacre was planned. And it may involve the most popular gaming console in the world, Sony's **PlayStation 4**.

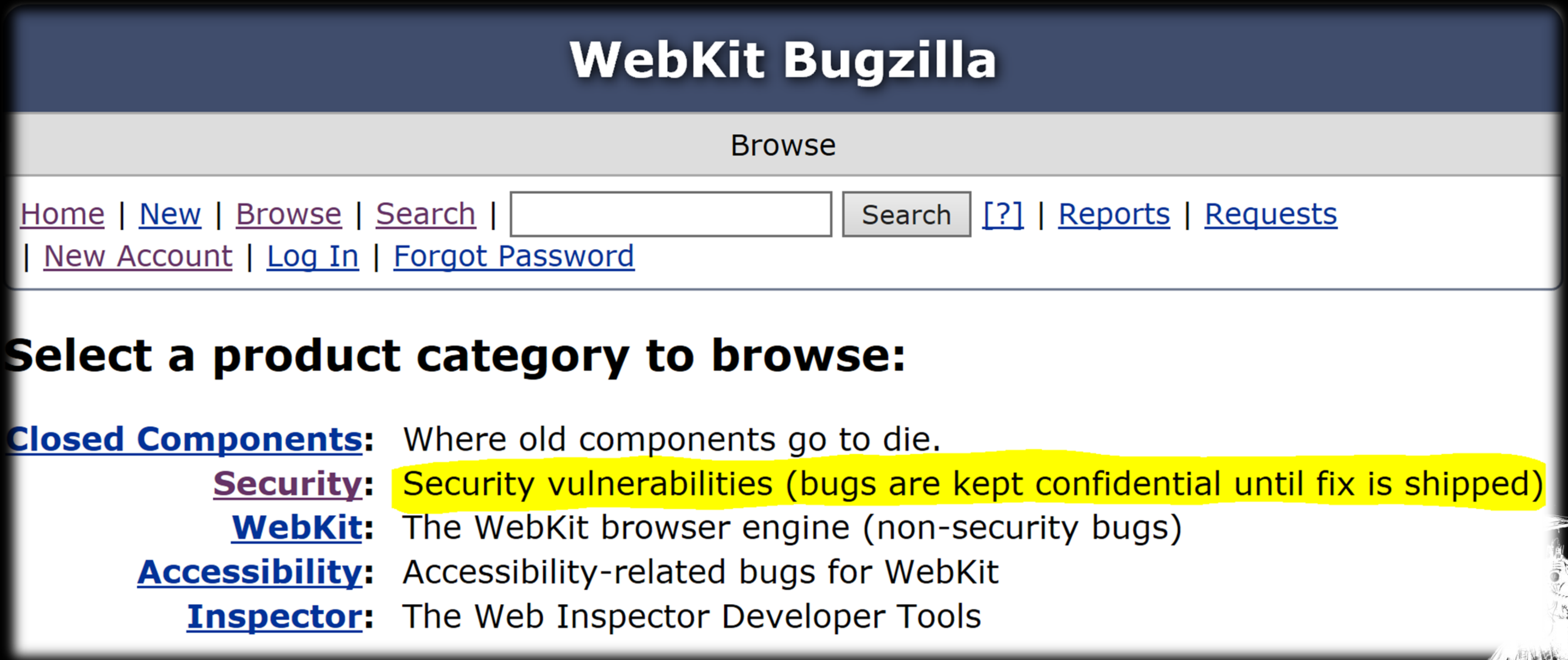


There's no link between the PS4 and the Paris attacks | WIRED UK
www.wired.co.uk/article/ps4-connected-paris-attacks-isis ▾
Nov 17, 2015 - A story on the site titled "How **Paris** ISIS **Terrorists** May Have Used **PlayStation 4** To Discuss And Plan **Attacks**" claimed prosecutors had uncovered "at least one" PS4 console in raids in Brussels. Now it has emerged that no such evidence was found, while quotes from Belgian officials included in the article ...

Agenda

- WebKit exploitation
- FreeBSD x86_64 exploitation
- Hardware and firmware
- Dumping FreeBSD ARM kernel of southbridge
- Running user code on ARM
- FreeBSD ARM exploitation
- Hardware attacks and kernel bootloader extraction
- Future research

Finding WebKit exploit


Home | New | Browse | Search | Search [?] | Reports | Requests
| New Account | Log In | Forgot Password

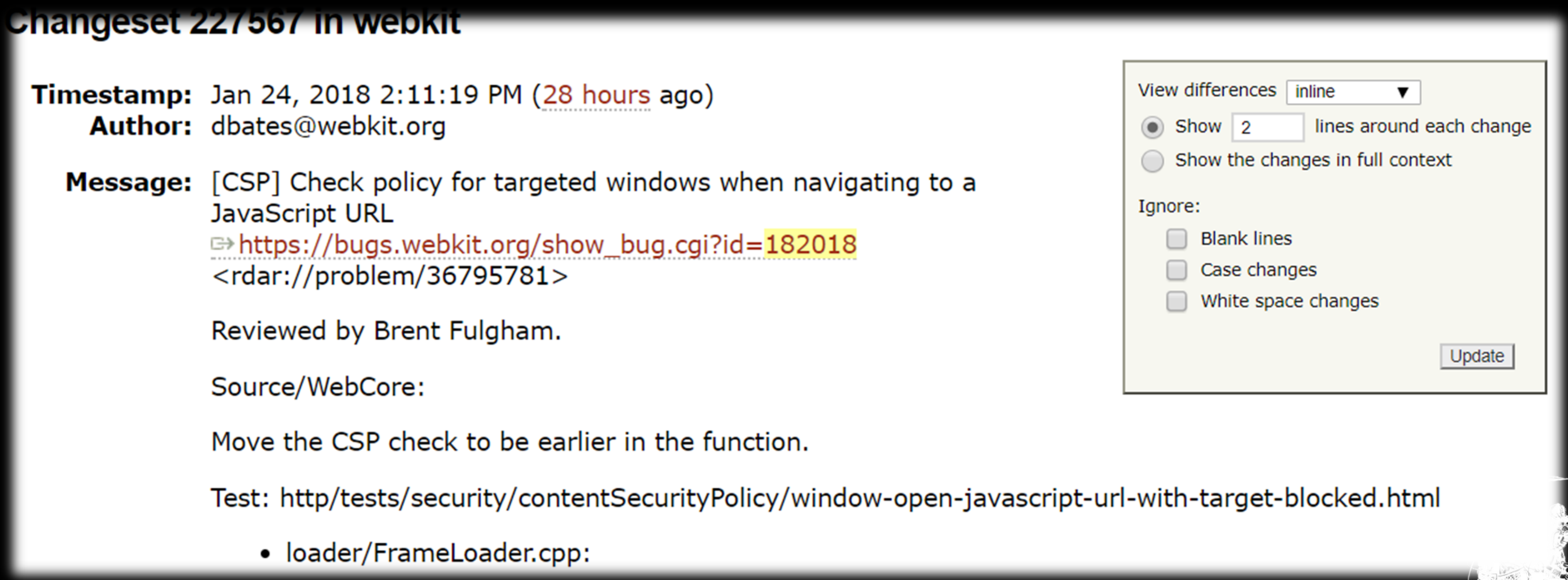
WebKit Bugzilla

Browse

Select a product category to browse:

Closed Components:	Where old components go to die.
Security:	Security vulnerabilities (bugs are kept confidential until fix is shipped)
WebKit:	The WebKit browser engine (non-security bugs)
Accessibility:	Accessibility-related bugs for WebKit
Inspector:	The Web Inspector Developer Tools

Changelog open for all!



Changeset 227567 in webkit

Timestamp: Jan 24, 2018 2:11:19 PM (28 hours ago)
Author: dbates@webkit.org

Message: [CSP] Check policy for targeted windows when navigating to a JavaScript URL
https://bugs.webkit.org/show_bug.cgi?id=182018
<rdar://problem/36795781>

Reviewed by Brent Fulgham.

Source/WebCore:

Move the CSP check to be earlier in the function.

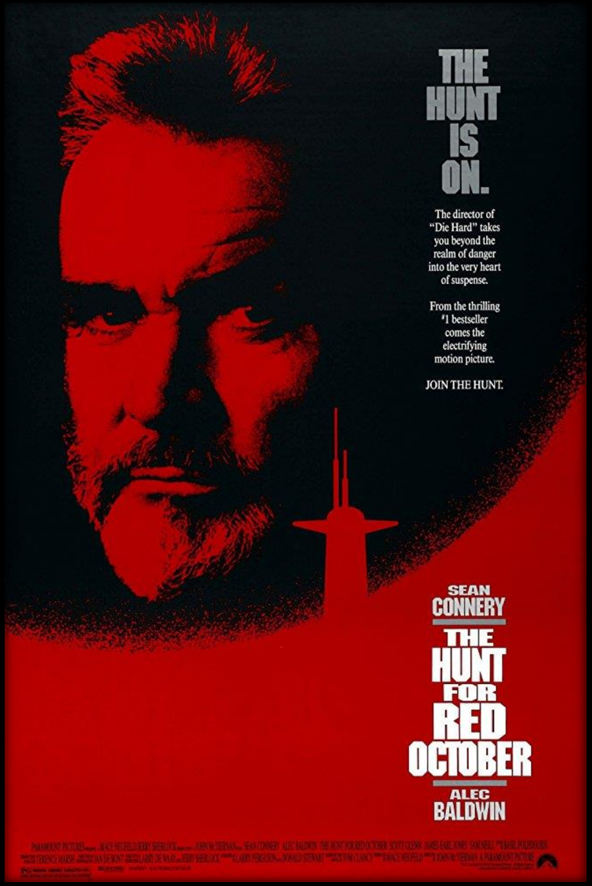
Test: http/tests/security/contentSecurityPolicy/window-open-javascript-url-with-target-blocked.html

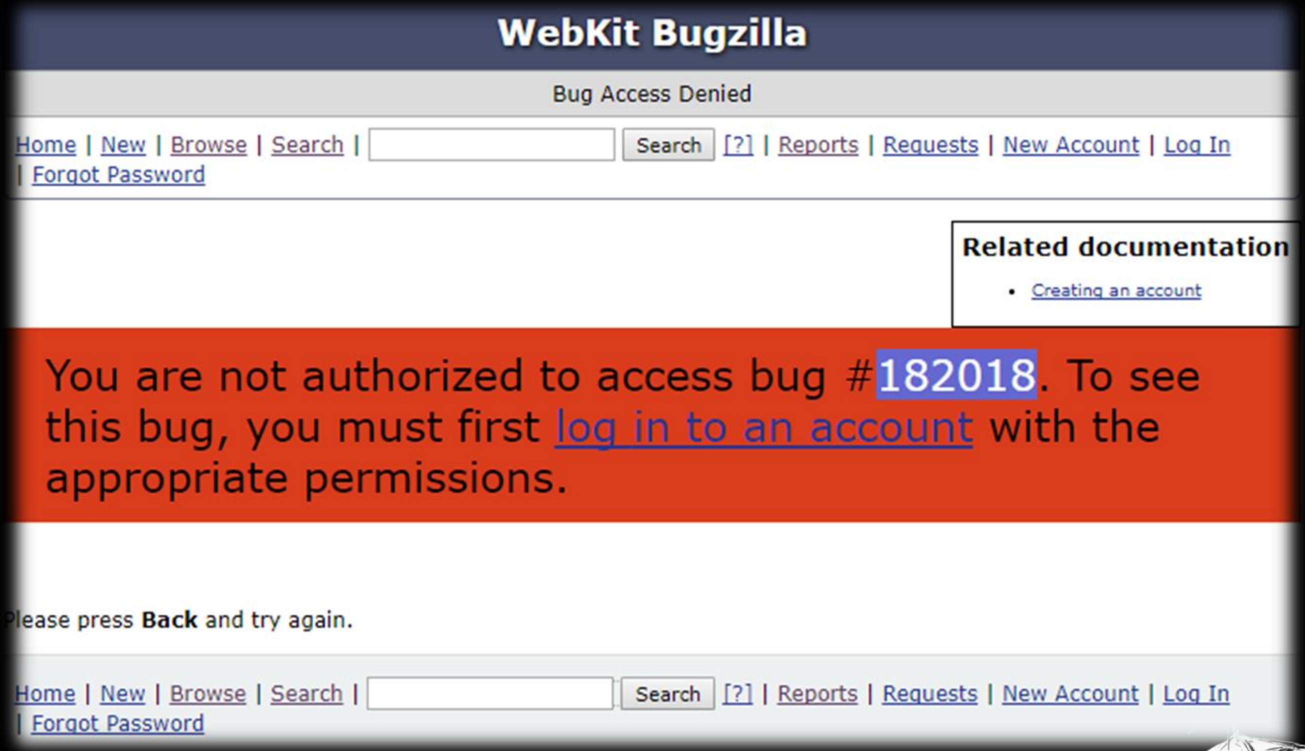
- loader/FrameLoader.cpp:

View differences inline ▼
● Show 2 lines around each change
○ Show the changes in full context

Ignore:
□ Blank lines
□ Case changes
□ White space changes

The Hunt for Red October


THE HUNT IS ON.
The director of "Die Hard" takes you beyond the realm of danger into the very heart of suspense.
From the thrilling #1 bestseller comes the electrifying motion picture.
JOIN THE HUNT.
SEAN CONNERY
THE HUNT FOR RED OCTOBER
ALEC BALDWIN



WebKit Bugzilla

Bug Access Denied

Home | New | Browse | Search | Search [?] | Reports | Requests | New Account | Log In | Forgot Password

Related documentation

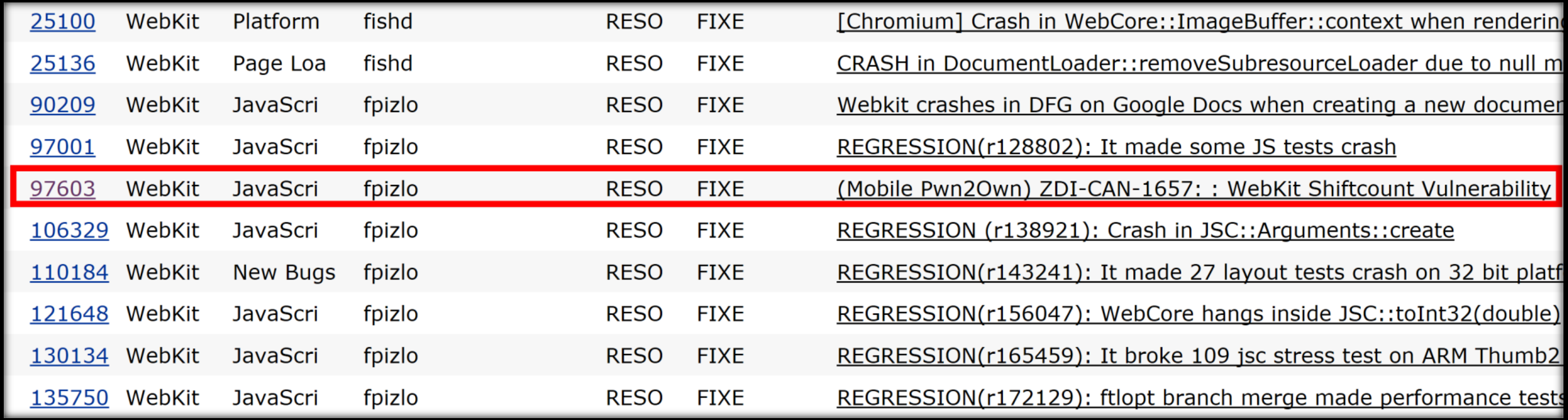
- Creating an account

You are not authorized to access bug #182018. To see this bug, you must first log in to an account with the appropriate permissions.

Please press **Back** and try again.

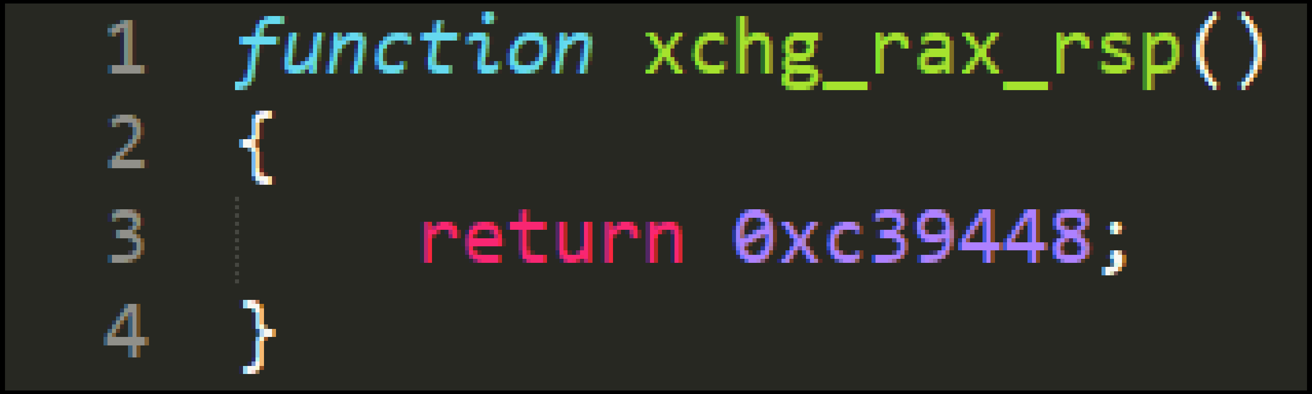
Home | New | Browse | Search | Search [?] | Reports | Requests | New Account | Log In | Forgot Password

Use existing exploit CVE-2012-3748



25100	WebKit	Platform	fishd	RESO	FIXE	[Chromium] Crash in WebCore::ImageBuffer::context when renderin
25136	WebKit	Page Loa	fishd	RESO	FIXE	CRASH in DocumentLoader::removeSubresourceLoader due to null m
90209	WebKit	JavaScri	fpizlo	RESO	FIXE	Webkit crashes in DFG on Google Docs when creating a new documen
97001	WebKit	JavaScri	fpizlo	RESO	FIXE	REGRESSION(r128802): It made some JS tests crash
97603	WebKit	JavaScri	fpizlo	RESO	FIXE	(Mobile Pwn2Own) ZDI-CAN-1657: : WebKit Shiftcount Vulnerability
106329	WebKit	JavaScri	fpizlo	RESO	FIXE	REGRESSION (r138921): Crash in JSC::Arguments::create
110184	WebKit	New Bugs	fpizlo	RESO	FIXE	REGRESSION(r143241): It made 27 layout tests crash on 32 bit platf
121648	WebKit	JavaScri	fpizlo	RESO	FIXE	REGRESSION(r156047): WebCore hangs inside JSC::toInt32(double)
130134	WebKit	JavaScri	fpizlo	RESO	FIXE	REGRESSION(r165459): It broke 109 jsc stress test on ARM Thumb2
135750	WebKit	JavaScri	fpizlo	RESO	FIXE	REGRESSION(r172129): ftlopt branch merge made performance tests

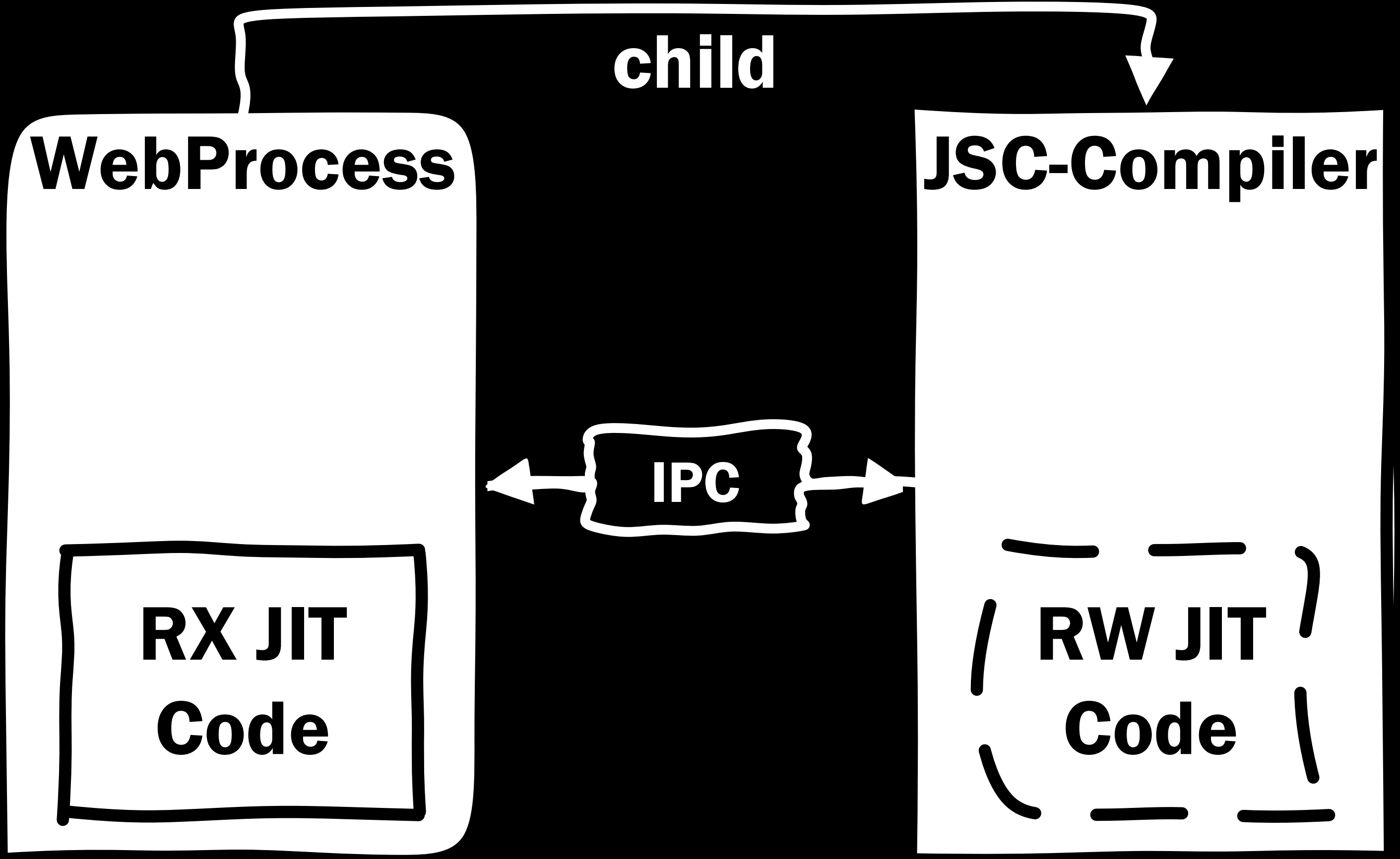
https://www.exploit-db.com/exploits/28081/



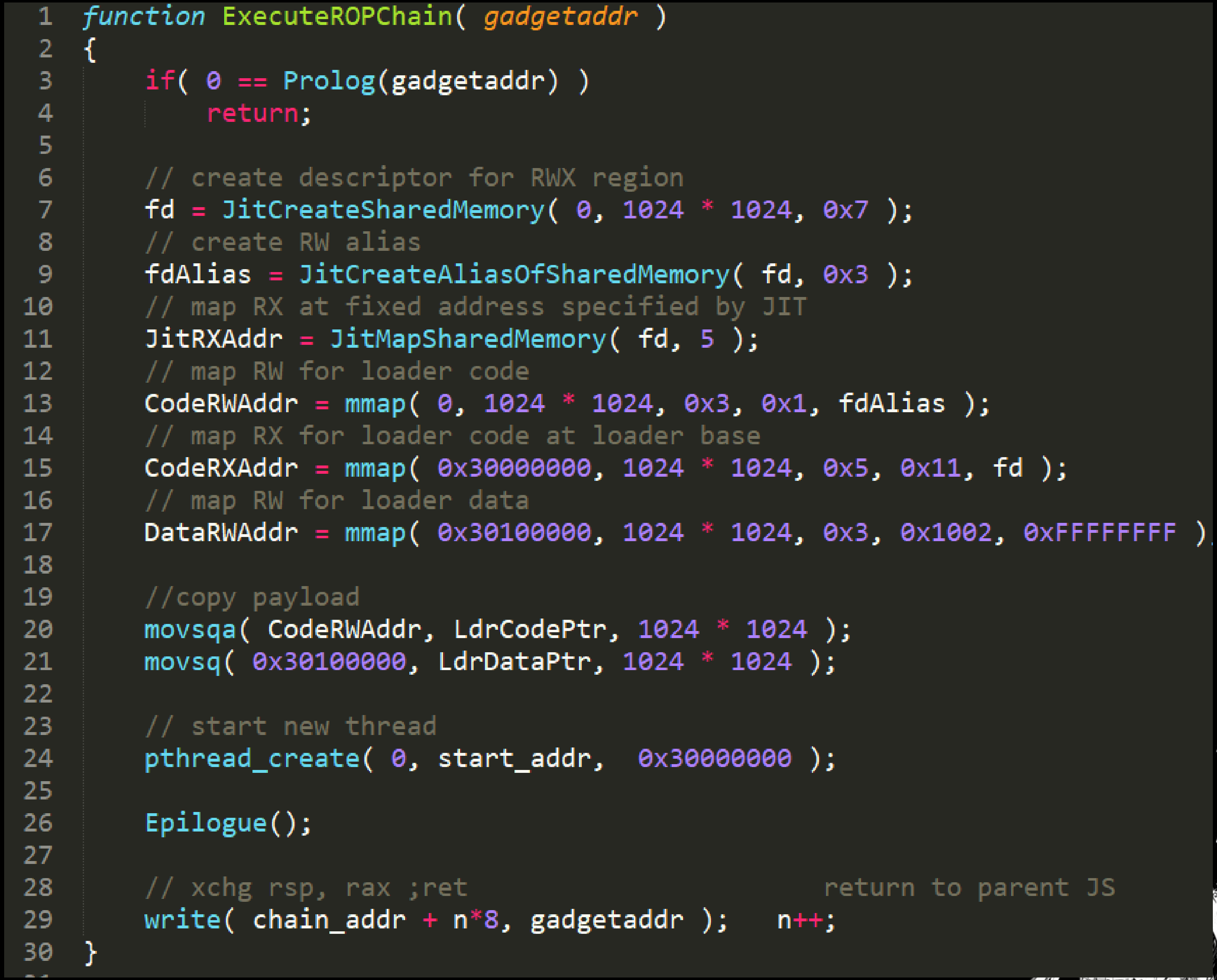
```
function xchg_rax_rsp()
{
    return 0xc39448;
}
```

ROP ONLY no RWX memory!

JIT how does it work? (magnets?)

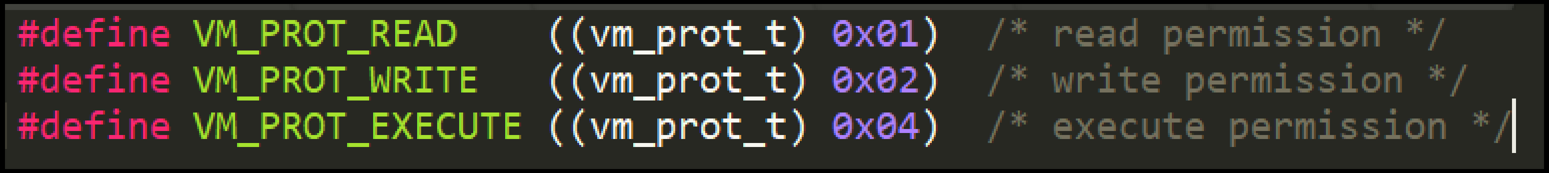


- Create RWX JIT shared memory (SHM)
- Create alias of this SHM with RW access
- Map RX JIT SHM using original FD
- Map RW JIT SHM using alias.
- Map RX 0x30000000
- Map RW 0x30100000
- Pthread_create



```
function ExecuteROPChain( gadgetaddr )
{
    if( 0 == Prolog(gadgetaddr) )
        return;

    // create descriptor for RWX region
    fd = JitCreateSharedMemory( 0, 1024 * 1024, 0x7 );
    // create RW alias
    fdAlias = JitCreateAliasOfSharedMemory( fd, 0x3 );
    // map RX at fixed address specified by JIT
    JitRXAddr = JitMapSharedMemory( fd, 5 );
    // map RW for loader code
    CodeRWAddr = mmap( 0, 1024 * 1024, 0x3, 0x1, fdAlias );
    // map RX for loader code at loader base
    CodeRXAddr = mmap( 0x30000000, 1024 * 1024, 0x5, 0x11, fd );
    // map RW for loader data
    DataRWAddr = mmap( 0x30100000, 1024 * 1024, 0x3, 0x1002, 0xFFFFFFFF )

    //copy payload
    movsqa( CodeRWAddr, LdrCodePtr, 1024 * 1024 );
    movsq( 0x30100000, LdrDataPtr, 1024 * 1024 );

    // start new thread
    pthread_create( 0, start_addr,  0x30000000 );

    Epilogue();

    // xchg rsp, rax ;ret                      return to parent JS
    write( chain_addr + n*8, gadgetaddr );   n++;
}
```

RWX without JIT



```
#define VM_PROT_READ    ((vm_prot_t) 0x01)  /* read permission */
#define VM_PROT_WRITE   ((vm_prot_t) 0x02)  /* write permission */
#define VM_PROT_EXECUTE ((vm_prot_t) 0x04)  /* execute permission */
```

Start		End	prot	maxprot	Info
0x000007ff3e4000	-	0x000007ff3e8000	0	3	stack guard
0x000007ff3e8000	-	0x000007ff5e8000	3	3	Thread1
0x000007ff5e8000	-	0x000007ff5ec000	0	3	stack guard
0x000007ff5ec000	-	0x000007ff7ec000	3	3	Thread2
0x000007ff7ec000	-	0x000007ff7f0000	0	3	stack guard
0x000007ff7f0000	-	0x000007ff9f0000	3	3	Thread3
0x000007ff9f0000	-	0x000007ff9f4000	0	3	stack guard
0x000007ff9f4000	-	0x000007ffbf4000	3	3	Thread4
0x000007ffdf8000	-	0x000007ffdfc000	0	33	
0x000007ffdfc000	-	0x000007ffffc000	3	3	main stack
0x000007ffffc000	-	0x000008000000000	5	37	

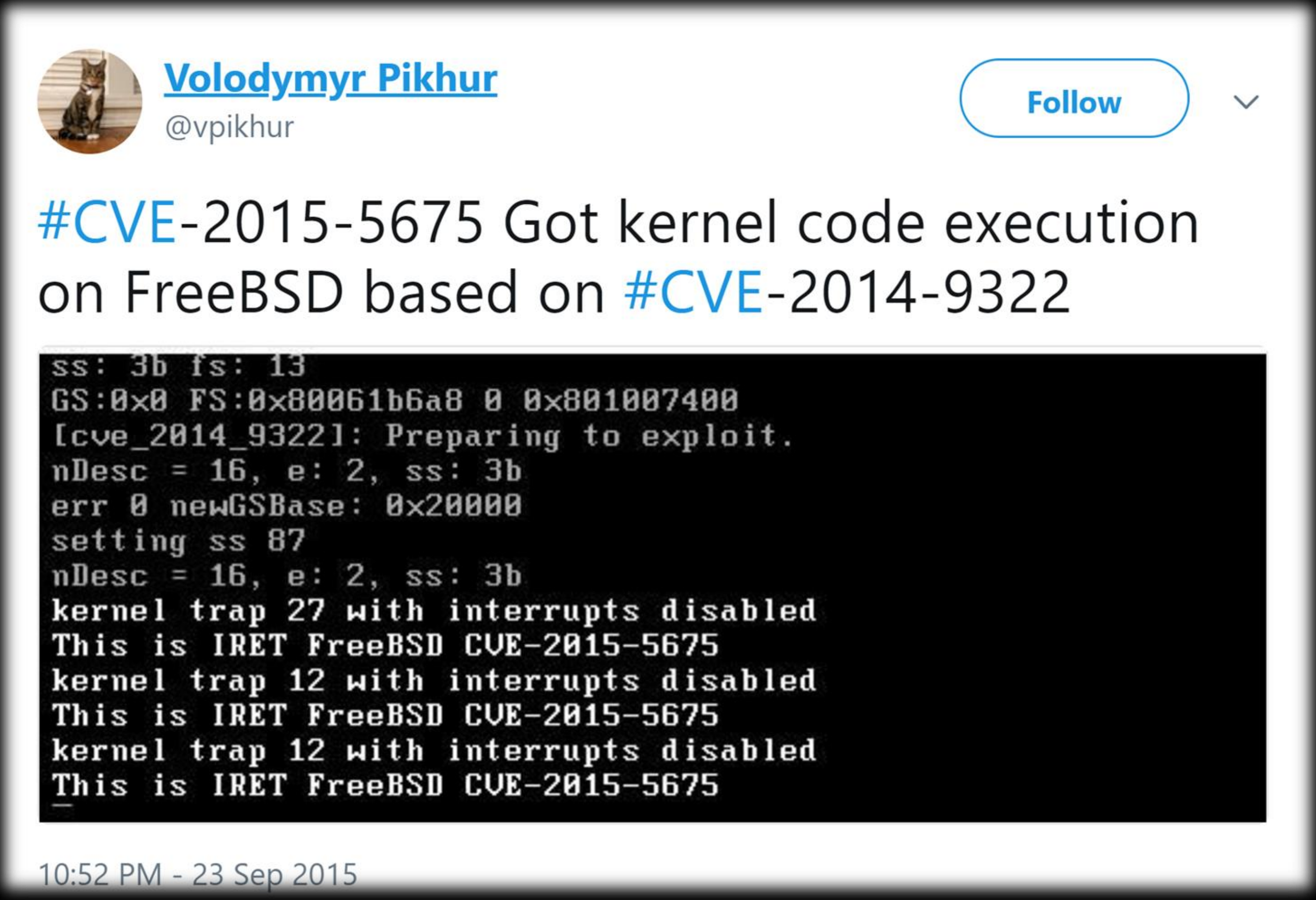
Privilege escalation

- Kernel
  - Syscall exploitation is difficult black box isn’t fun ☹
  - Maximum what we can get are info leaks in FreeBSD
    - Kernel callstack using sysctl KERN_PROC_KSTACK (requires two threads)
    - Pointer leak (CVE-2014-8476)
- Services
  - Still in their own jail but have more priviledges able to call more syscalls
  - Bugs are present but unable to get code exec
    - Multiple crashes via IPC

Kernel code execution

- BadIRET (CVE-2014-9322, CVE-2015-5675)
  - CVE-2015-5675 (2015-08-25)
    - https://www.freebsd.org/security/advisories/FreeBSD-SA-15:21.amd64.asc
  - CVE-2014-9322
    - Rafal’s excellent guide on this bug
      - https://blogs.bromium.com/exploiting-badiret-vulnerability-cve-2014-9322-linux-kernel-privilege-escalation/

FreeBSD PoC



Volodymyr Pikhur
@vpikhur

#CVE-2015-5675 Got kernel code execution on FreeBSD based on #CVE-2014-9322

```
ss: 3b fs: 13
GS:0x0 FS:0x80061b6a8 0 0x801007400
[cve_2014_9322]: Preparing to exploit.
nDesc = 16, e: 2, ss: 3b
err 0 newGSBase: 0x20000
setting ss 87
nDesc = 16, e: 2, ss: 3b
kernel trap 27 with interrupts disabled
This is IRET FreeBSD CVE-2015-5675
kernel trap 12 with interrupts disabled
This is IRET FreeBSD CVE-2015-5675
kernel trap 12 with interrupts disabled
This is IRET FreeBSD CVE-2015-5675
_
```

10:52 PM - 23 Sep 2015

- “This is a POC to reproduce vulnerability. No exploitation here, just simple kernel panic.”
- https://www.exploit-db.com/exploits/36266/

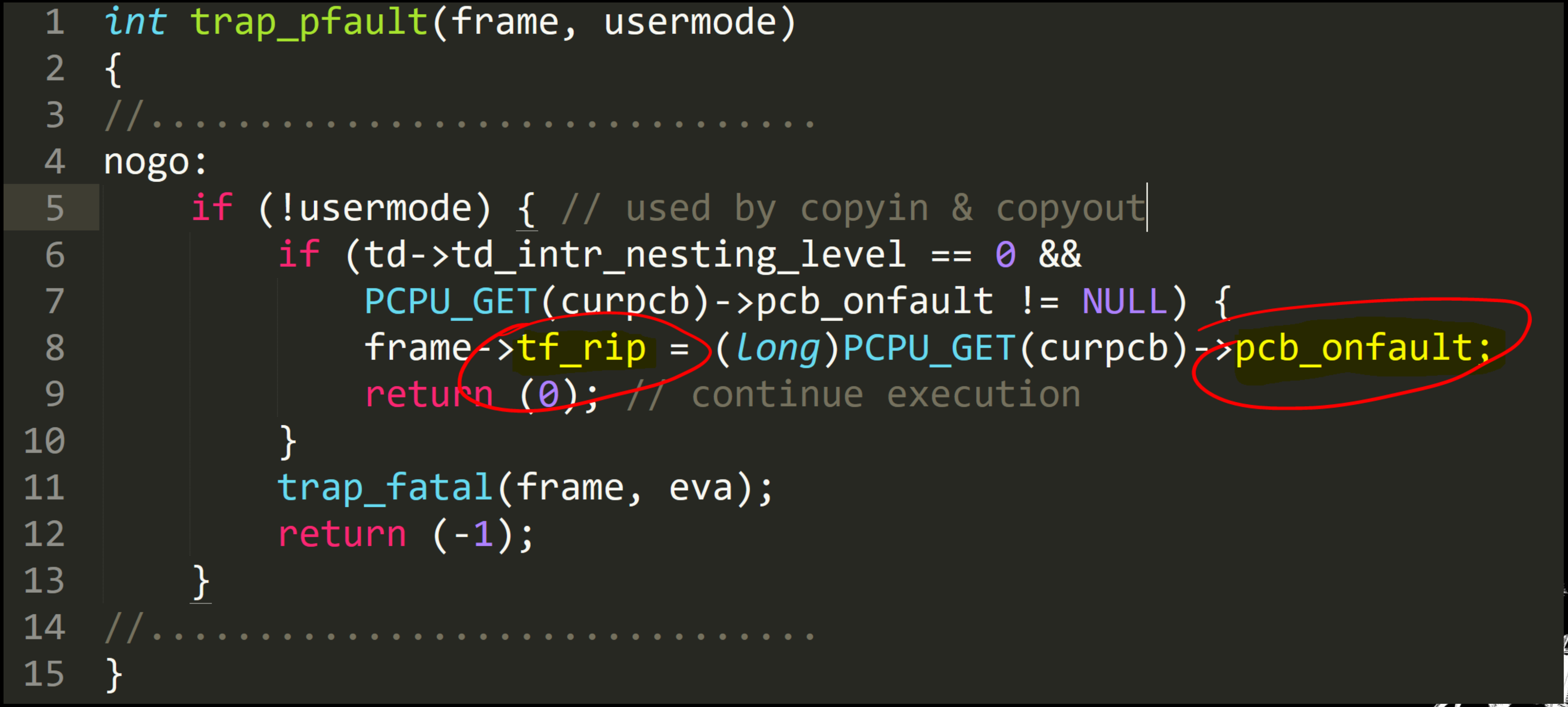
Rafal’s IDT pointer redirection

- Rafal’s approach

```
action = &t->sighand->action[sig-1];
action->sa.sa_handler = SIG_DFL; // SIG_DFL = 0
```

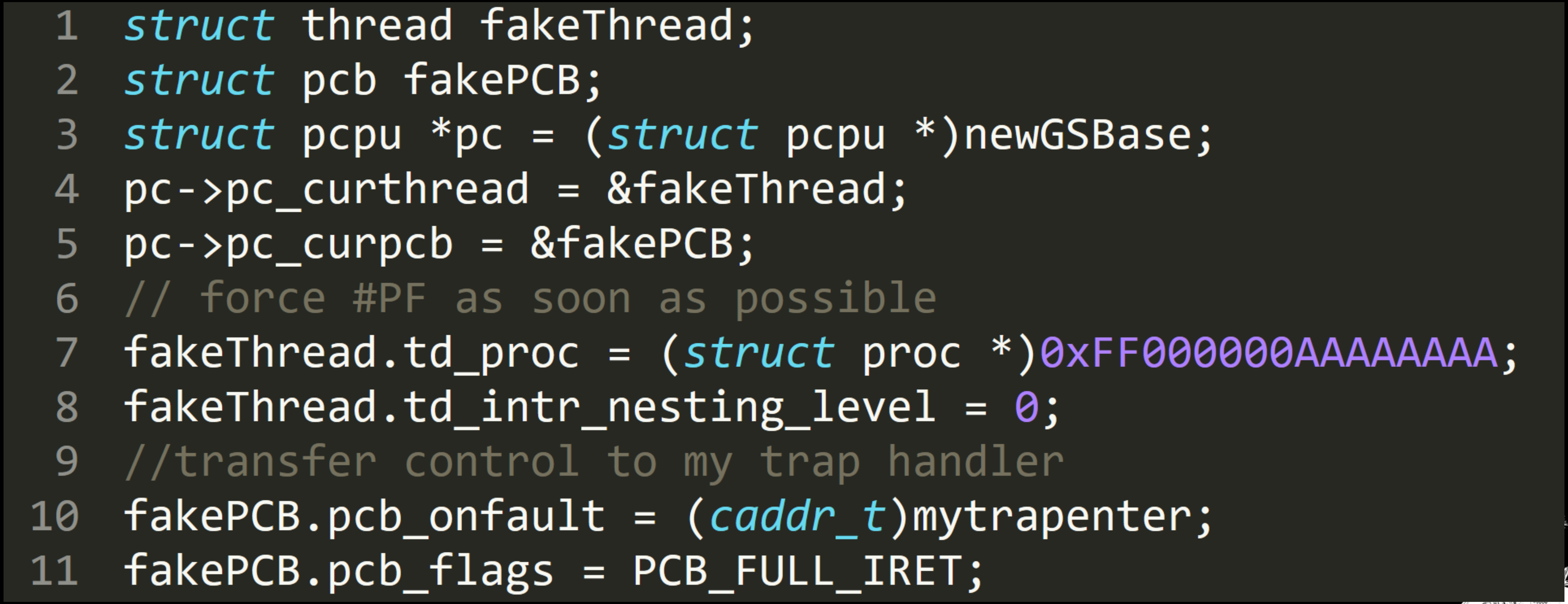
- IDT overwrite
  - Overwrite #PF handler address in IDT
  - IDT[#PF] = 0xFFFFFFFF’XXXXXXXX
  - IDT[#PF] = 0x00000000’XXXXXXXX
- FreeBSD increment primitive
  - td->td_critnest++
  - 0xFFFFFFFF + 1 = 0x0

PoC implementation #SS -> #PF -> pcb_onfault



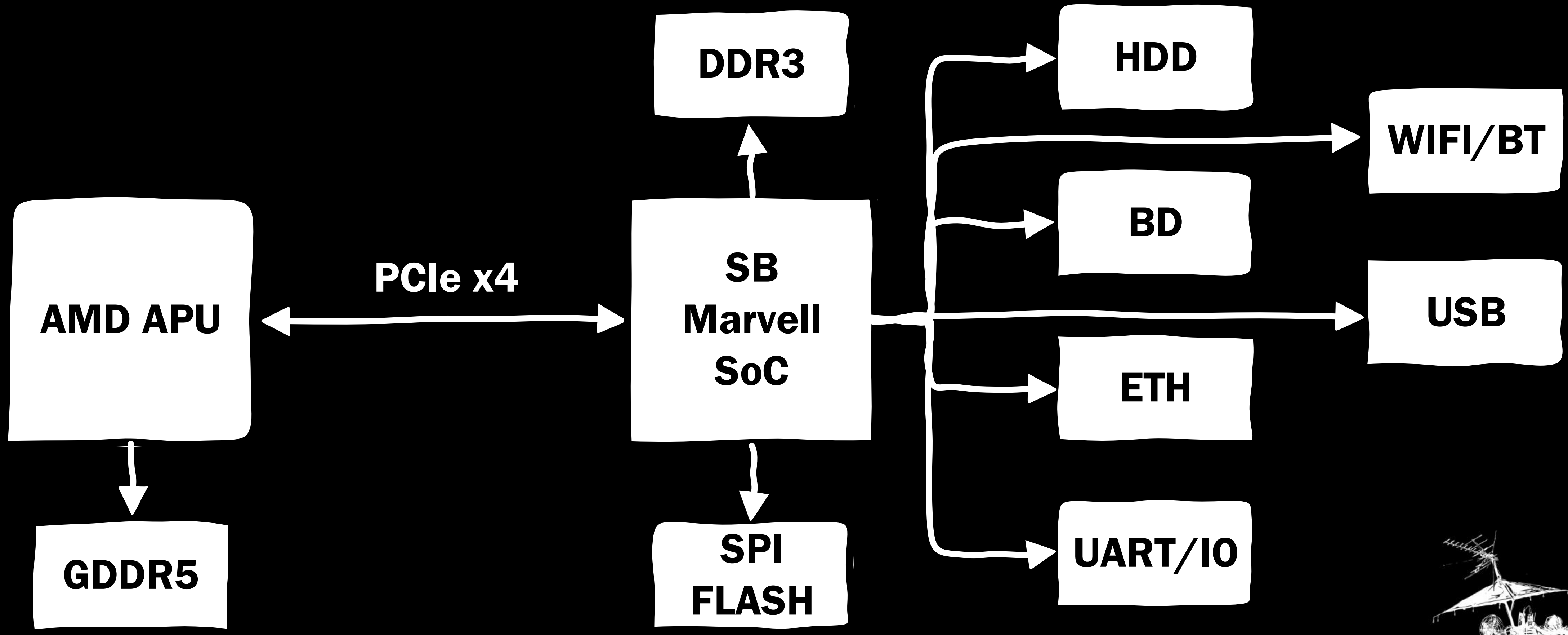
```
int trap_pfault(frame, usermode)
{
//................................
nogo:
    if (!usermode) { // used by copyin & copyout
        if (td->td_intr_nesting_level == 0 &&
            PCPU_GET(curpcb)->pcb_onfault != NULL) {
            frame->tf_rip = (long)PCPU_GET(curpcb)->pcb_onfault;
            return (0); // continue execution
        }
        trap_fatal(frame, eva);
        return (-1);
    }
//................................
}
```

BadIRET FreeBSD PoC implementation



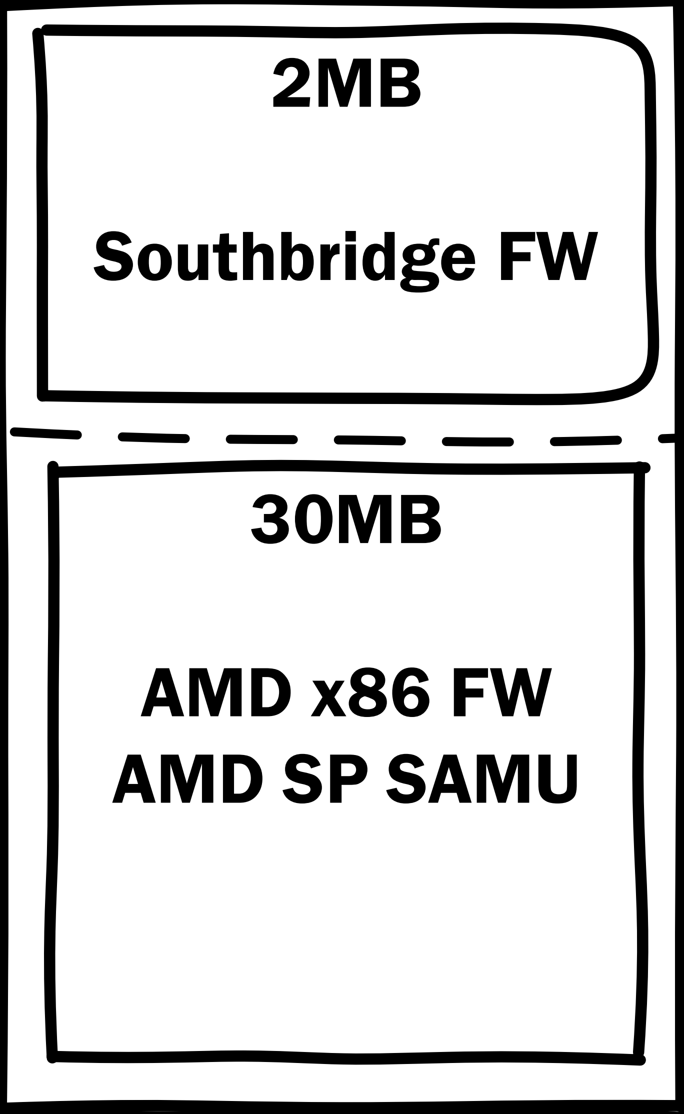
```c
struct thread fakeThread;
struct pcb fakePCB;
struct pcpu *pc = (struct pcpu *)newGSBase;
pc->pc_curthread = &fakeThread;
pc->pc_curpcb = &fakePCB;
// force #PF as soon as possible
fakeThread.td_proc = (struct proc *)0xFF000000AAAAAAAA;
fakeThread.td_intr_nesting_level = 0;
//transfer control to my trap handler
fakePCB.pcb_onfault = (caddr_t)mytrapenter;
fakePCB.pcb_flags = PCB_FULL_IRET;
```

Hardware overview



https://wikidevi.com/wiki/Marvell https://media.ccc.de/v/33c3-7946-console_hacking_2016

SPI Flash Firmware



2MB

Southbridge FW

30MB

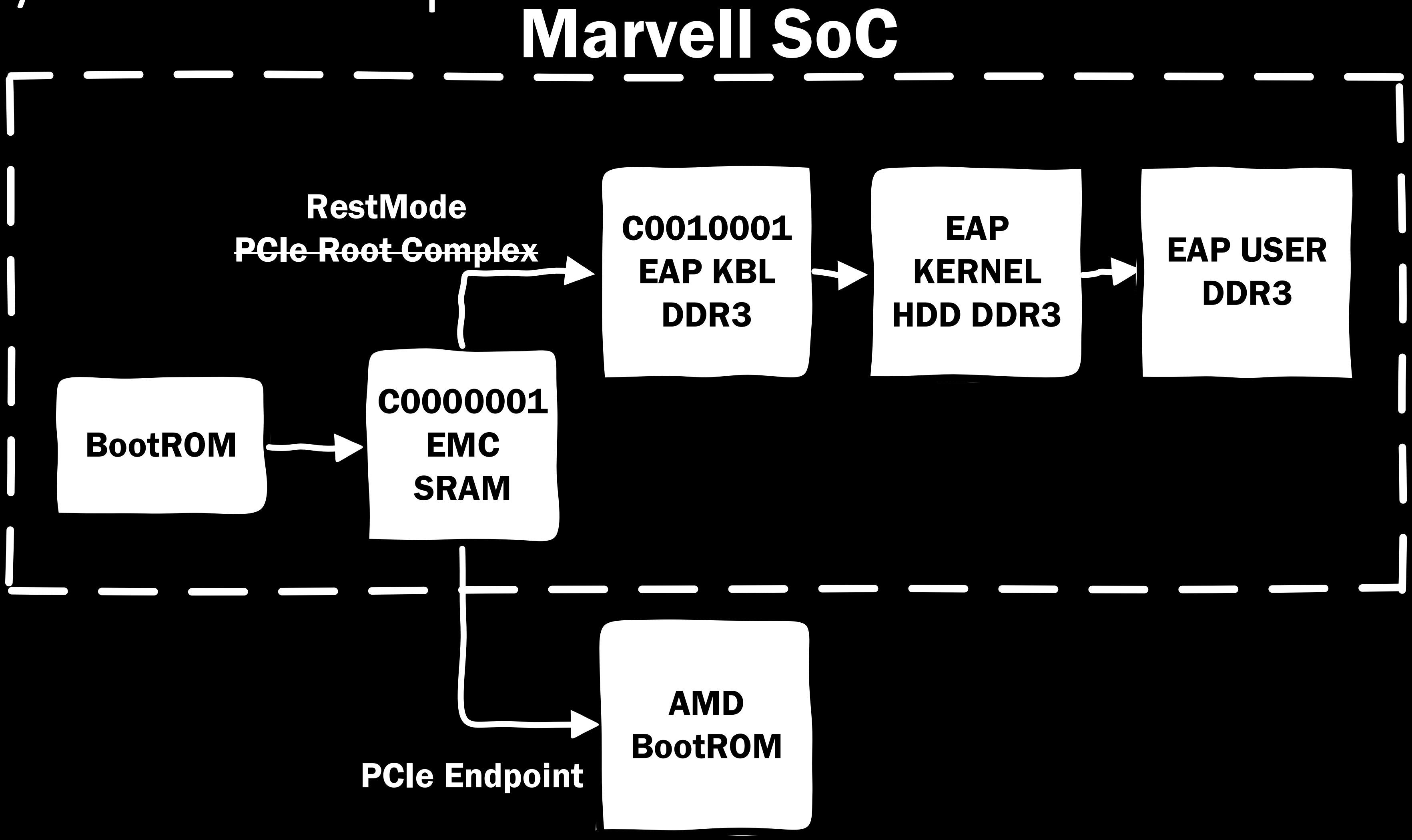
AMD x86 FW
AMD SP SAMU

- Marvell SoC “Aeolia/Belize/Baikal”
  - C0000001 (IPL – SRAM) aka EMC
  - C0010001 (KBL – DDR3) aka EAP
  - Torus WIFI/BT
  - NVS (config etc.)
- AMD APU
  - AES XTS encrypted with per console key
  - Secure Loader/Kernel/Modules
  - X86 BIOS/Kernel

HDD structure overview

- 15 GPT partitions
  - Encrypted with two sets of keys
- AMD SP
  - X86 Services/Modules/GUI C# Mono
  - Updates
- Southbridge
  - User files - 400GB+ UFS2
    - User files, Games, Settings, Browser history ;)
  - EAP ARM User - 128MB FAT
  - EAP ARM Kernel - not a FS (encrypted/signed blob)

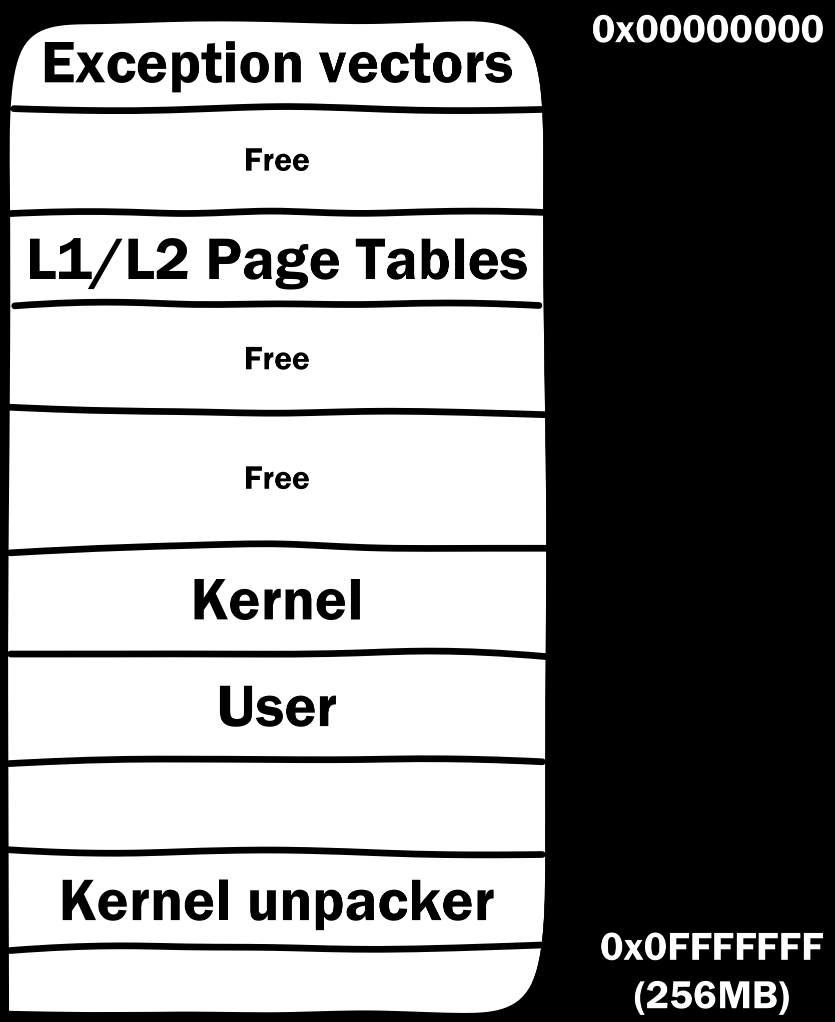
Boot/Power sequence



Cold Boot without cooling

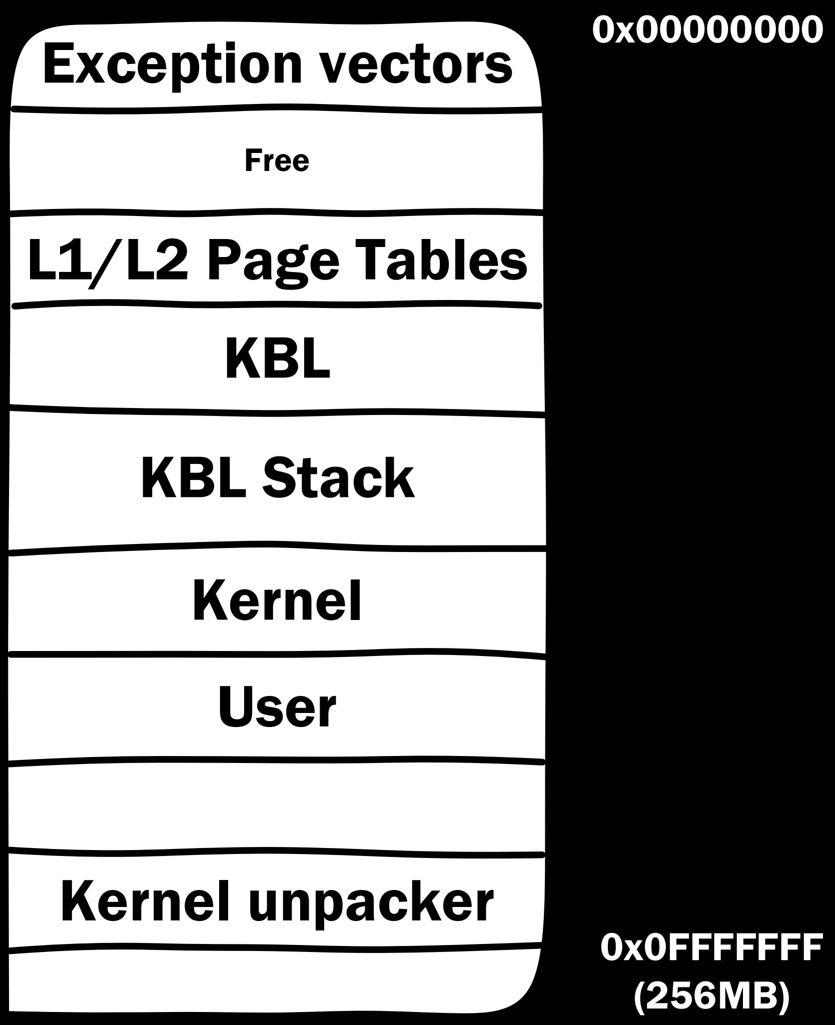
- DDR3 memory is directly mapped at 0xfffffe0080000000
    - sbram0: <Aeolia DDR3 memory> mem 0x80000000-0xbfffffff at device 20.6 on pci0
- DRAM stays without power for very small period of time during power cycle which is enough that contents of DRAM persist hence an attacker is able to dump it!

DDR3 Dump Analysis



- Kernel
  - Contiguous
  - 1:1 mapping
  - Raw binary no ELF header
  - No ASLR
- Kernel unpacker
  - Minimal ELF binary
  - Custom compression
- User
  - ASLR on newer FW
  - HMAC-SHA256 signing >2.xx FW

DDR3 Dump Analysis



- KBL
  - memset(bootp.kbl, 0, bootp.kbl_size);
- KBL Stack
  - Stack cookies
  - Return address to Kernel unpacker
  - Garbage
  - No keys! ☹

Running code on ARM

- No signing required on 1.xx (HMAC-SHA256 on 2.xx+)
  - Signing key still can be dumped from DRAM using cold boot on newer FW
- Crossbuild FreeBSD to support ARM
  - Override some structures and types to match correct size Sony decided default one aren’t good enough.
- Mount /eap_vsh and replace binary SceEapCore.elf
  - No network and other things ☹
  - No RWX
  - LDSCRIPT
    - Inject your payload inside the binary and place hook to spawn new thread!
- We are Root!

Kernel code exec

- Limited number of syscalls even less than on x86 kernel
- NOT an x86 can't use BadIRET exploit
- No Sony's syscalls like sys_dlclose, sys_namedobj, etc.
  - http://cturt.github.io/dlclose-overflow.html
  - https://fail0verflow.com/blog/2017/ps4-namedobj-exploit/
- Old exploits? I didn't find anything useful. ☹
- sys_kldload JACKPOT!
  - Basic FreeBSD functionality to load kernel modules was left behind!
  - Load helloworld.ko module -> CRASH! ☹

sys_kldload crash root cause analysis

- Bad ELF format?
- Correct kernel version?
- Did Sony change something?
- Trying different binaries gives inconsistent behavior
  - Sometimes crashes sometimes not
  - Load success but no execution!?
- Malloc! – kernel uses malloc to allocate memory for kernel modules
  - pmap_enter strips X bit and returns RW memory

```
if ( prot & VM_PROT_WRITE )
   prot = prot & ~VM_PROT_EXECUTE;
```

ROP validation

- To validate that I have working kernel module I had to redirect entry point to executable code inside kernel itself
  - BX LR - just return should not crash
  - Invalid pointer – should crash
- **DECLARE_MODULE macro**
  - FreeBSD already points inside of kernel!
    - MODULE_METADATA(_md_##name, MDT_MODULE, &data, #name);
    - SYSINIT(name##module, sub, order, module_register_init, &data);
- **PC and R0 control**
  - void module_register_init(const void *arg)

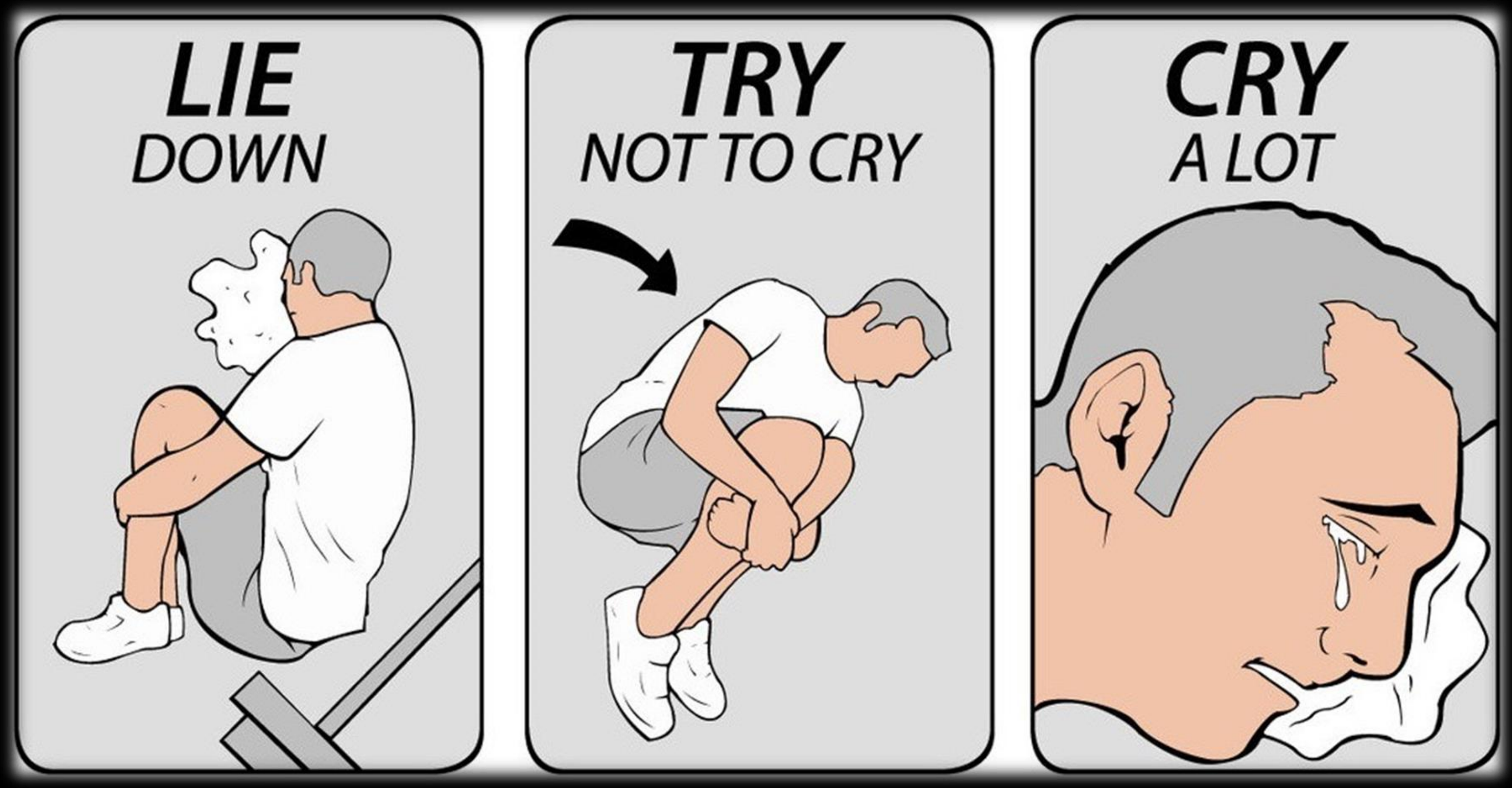
Arbitrary kernel code execution

1) **Load 1st module**
   - Patch L1 table to make kernel pages RWX instead RX only
2) **Load 2nd module**
   - patch pmap_enter and allow RWX memory
   - Conveniently when kernel loads new module it does TLB and cache invalidate
   - Otherwise if we would try to do write to kernel right after we patch L1 it would crash so don’t do ROP-chain.
3) **Load 3rd module**
   - We able to load kernel module and run own kernel code
   - PROFIT! (SHOTS!)

Now what!?

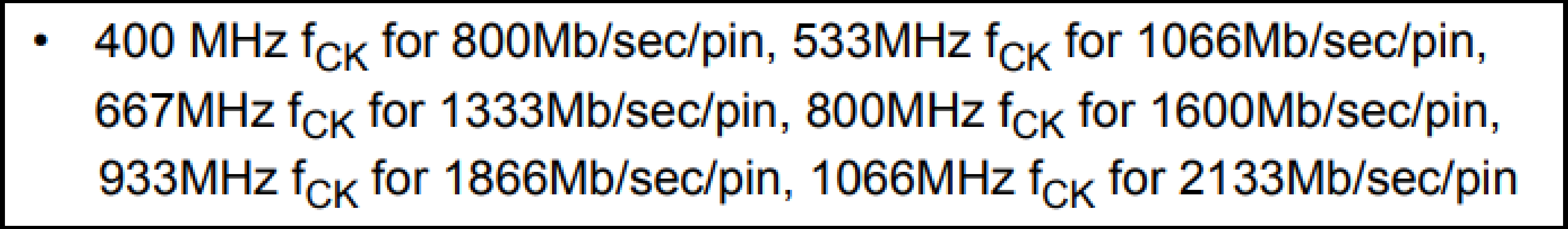
- **Co-processor registers**
  - CP0, CP14, CP15
  - CP14 - ARM debug registers available to software
- **Data abort handler**
  - Allows to scan memory and resume if that memory is unavailable
    - No other MMIO than what is already referenced in kernel
    - No 1MB register configuration space https://patchwork.kernel.org/patch/6169481/
  - When no paging enabled ARM says it is undefined behavior
    - I found hard limit of 256 failed aborts until unrecoverable crash
  - Hangs on certain MMIO which requires power cycle manually

Nothing except ability to run code in kernel



Hardware specs

- **What kernel tells us**
  - CPU: PJ4C B0 rev 1 (Marvell core)
  - CPU clock : 500MHz, DDR clock : 800MHz
    - http://www.samsung.com/global/business/semiconductor/file/product/DS_K4B2G1646Q-BC_Rev103.pdf
    - At least 400MHz



- 400 MHz f_{CK} for 800Mb/sec/pin, 533MHz f_{CK} for 1066Mb/sec/pin, 667MHz f_{CK} for 1333Mb/sec/pin, 800MHz f_{CK} for 1600Mb/sec/pin, 933MHz f_{CK} for 1866Mb/sec/pin, 1066MHz f_{CK} for 2133Mb/sec/pin

Hardware attack vectors

- **SoC glitch**
  - Try to glitch when memset is executed to prevent KBL clear
    - Requires desoldering A LOT of decoupling capacitors
    - Unable to make it skip instructions
- **DRAM glitch**
  - Address/Data corruption?
  - Address aliasing?
  - Bank Aliasing?
  - Prevent memory writes?

DRAM attack vectors

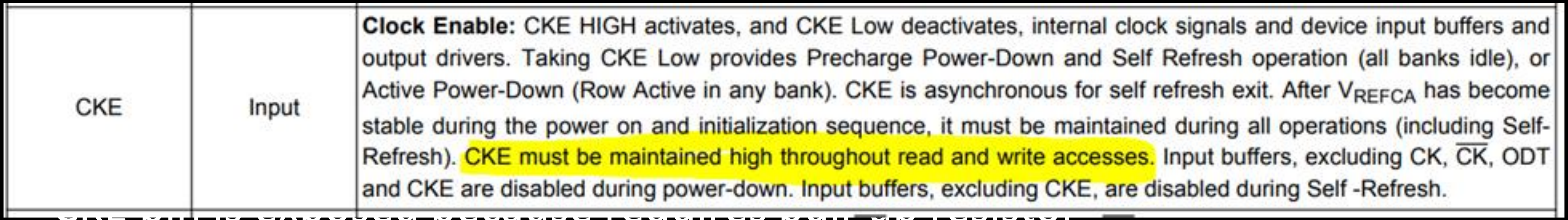
- **Address/Data corruption**
  - Need access to actual physical traces because it is BGA and data is deffirential they are located in inner layers
  - No all address pins are exposed due to BGA package
  - Trying to glitch address pins resulted in ‘byteswap’ instead of address change
- **Address aliasing**
  - Short some pins to make them HIGH e.g A0 and A8
  - Same problem pins not exposed
  - Probably should work on PC when attacking DIMMs

DRAM attack vectors

- **DRAM bank aliasing**
  - Similar to address aliasing except this time pins are exposed!
  - Connect e.g. B0 and B3 to make write happen to both
  - Disconnect when not needed (when KBL finished decrypting)
  - Read out secrets because they were written to both banks
  - It should work in theory but I couldn’t make it working or maybe I didn’t try hard enough ☹

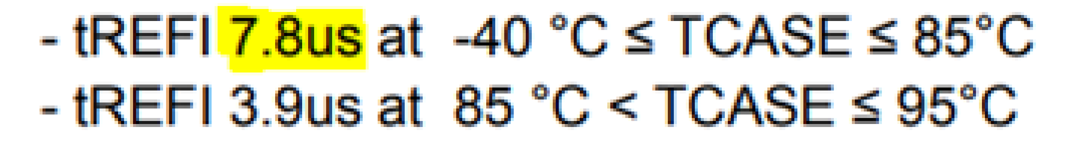
DRAM data write prevention

- CKE Must be maintained HIGH throughout read and write accesses.



CKE	Input	**Clock Enable:** CKE HIGH activates, and CKE Low deactivates, internal clock signals and device input buffers and output drivers. Taking CKE Low provides Precharge Power-Down and Self Refresh operation (all banks idle), or Active Power-Down (Row Active in any bank). CKE is asynchronous for self refresh exit. After V_{REFCA} has become stable during the power on and initialization sequence, it must be maintained during all operations (including Self-Refresh). CKE must be maintained high throughout read and write accesses. Input buffers, excluding CK, $\overline{CK}$, ODT and CKE are disabled during power-down. Input buffers, excluding CKE, are disabled during Self -Refresh.

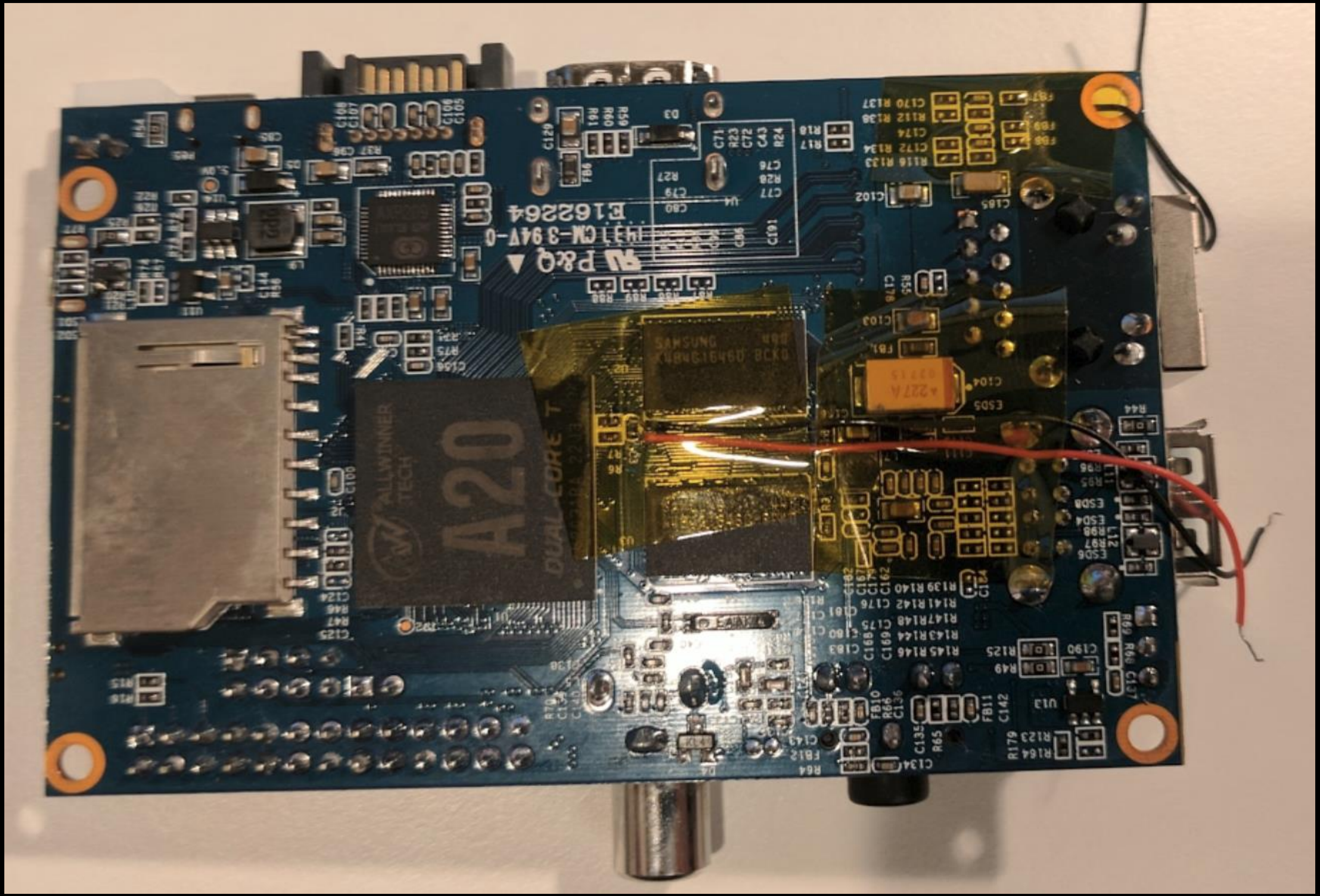
- Not just READ/WRITE also refresh and other commands



- tREFI 7.8us at -40 °C ≤ TCASE ≤ 85°C
- tREFI 3.9us at 85 °C < TCASE ≤ 95°C

- https://twitter.com/vpikhur/status/68089967414763520 (Dec 2015)
- Easy to identify the pin on target board with oscilloscope

Recon mission



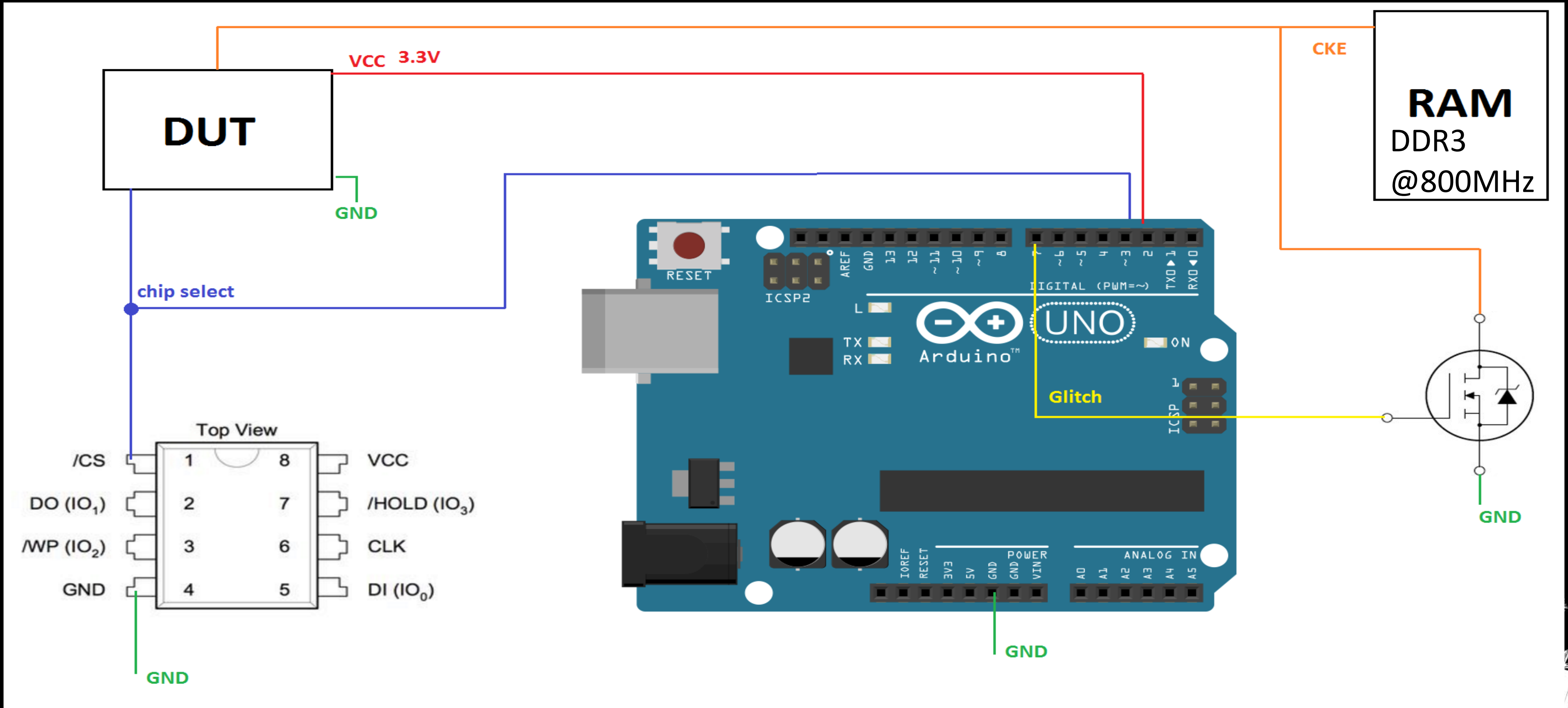
- Banana Pi
  - ARM
  - DDR3 1GB
  - Uboot

https://en.wikipedia.org/wiki/Banana_Pi

Hardware tools

- Oscilloscope
  - http://www.dreamsourcelab.com/order.html ($199)
  - Initially had pretty bad software now it’s OKish
  - Drivers have no digital signature ☹
- Arduino Uno ($10)
  - Signal sensing
  - Timing delays
  - Trigger
- MOSFET ($0)
  - Connects CKE to GND on trigger to generate glitch

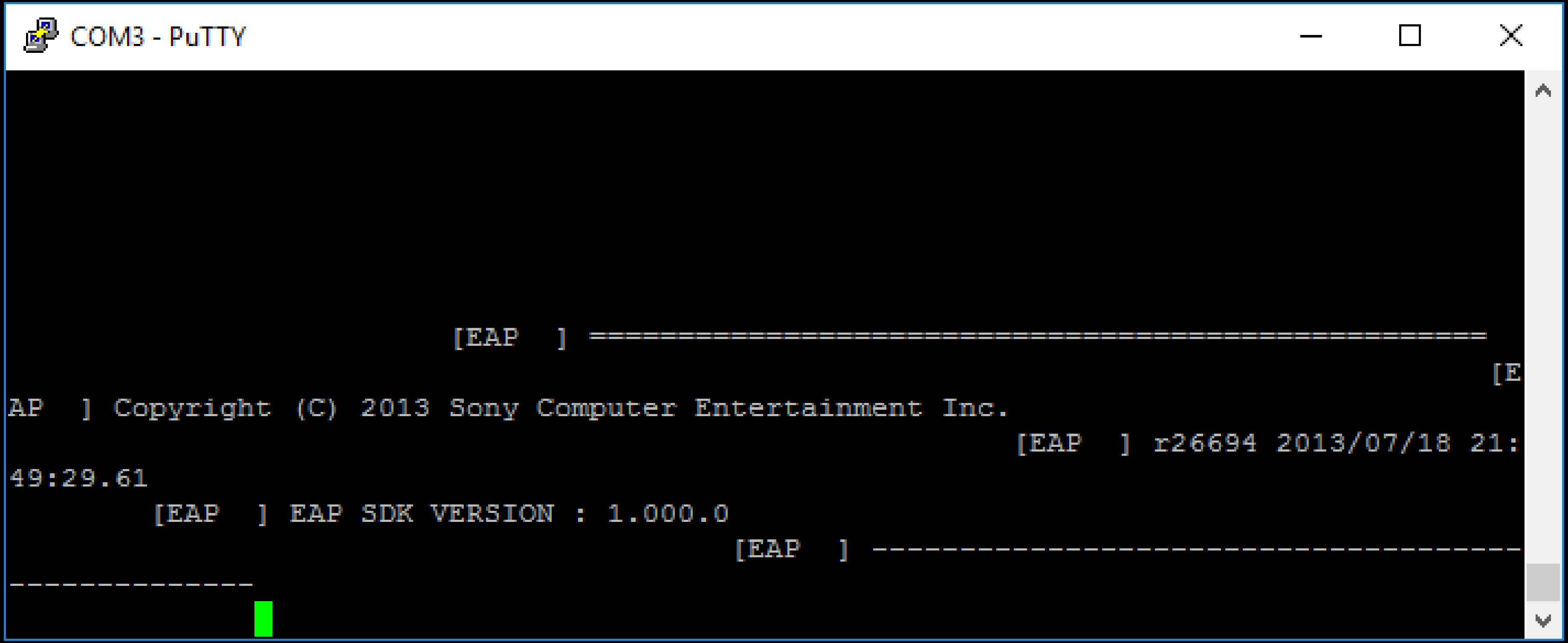
Glitch setup



Memset glitch vs KBL glitch

- Impossible to guess when exactly it is happening
  - HDD creates inconsistent delays
  - Even SSD doesn’t work well enough
- KBL glitch (code injection)
  - From main OS x86 using kexploit spray DDR3 memory with MOV PC, 0x3C and at 0x3C offset we place our payload
  - Enter rest mode spray will remain in memory
  - Glitch when KBL gets loaded to gain code execution then dump KBL via UART our payload

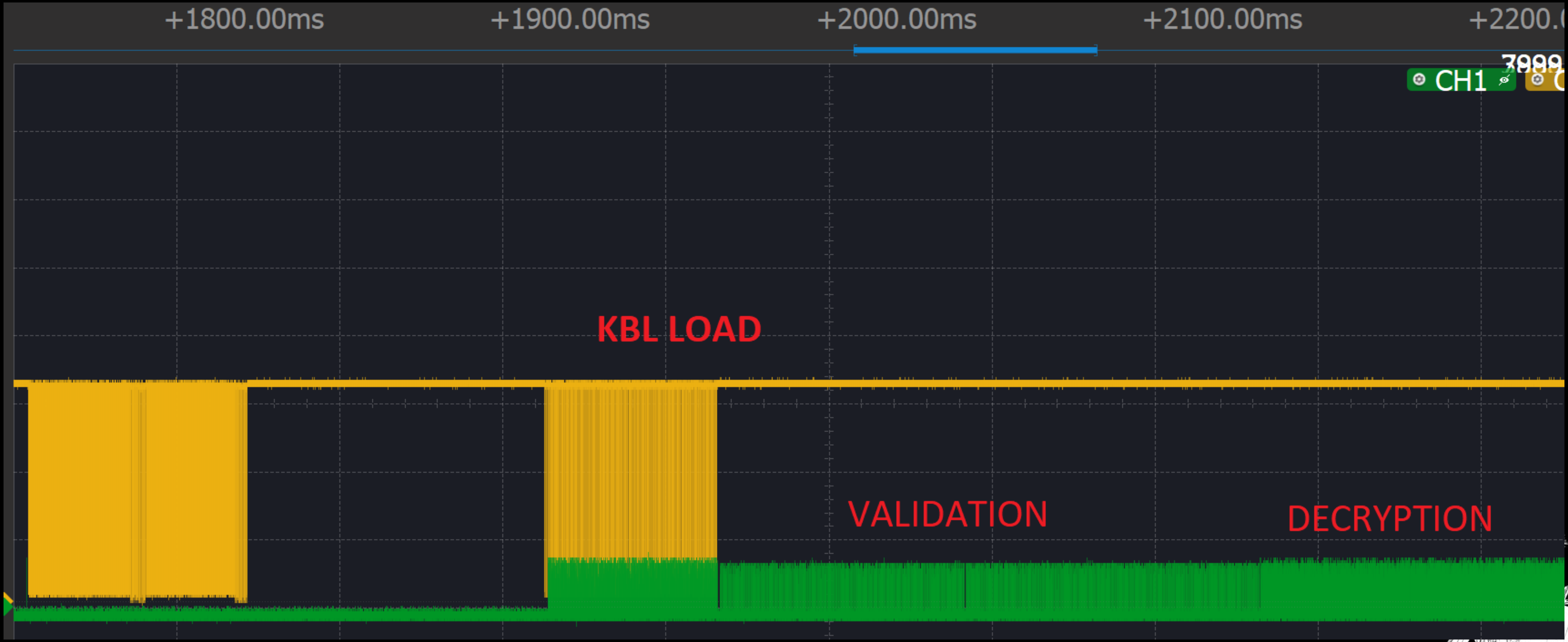
UART log <= v1.05 FW



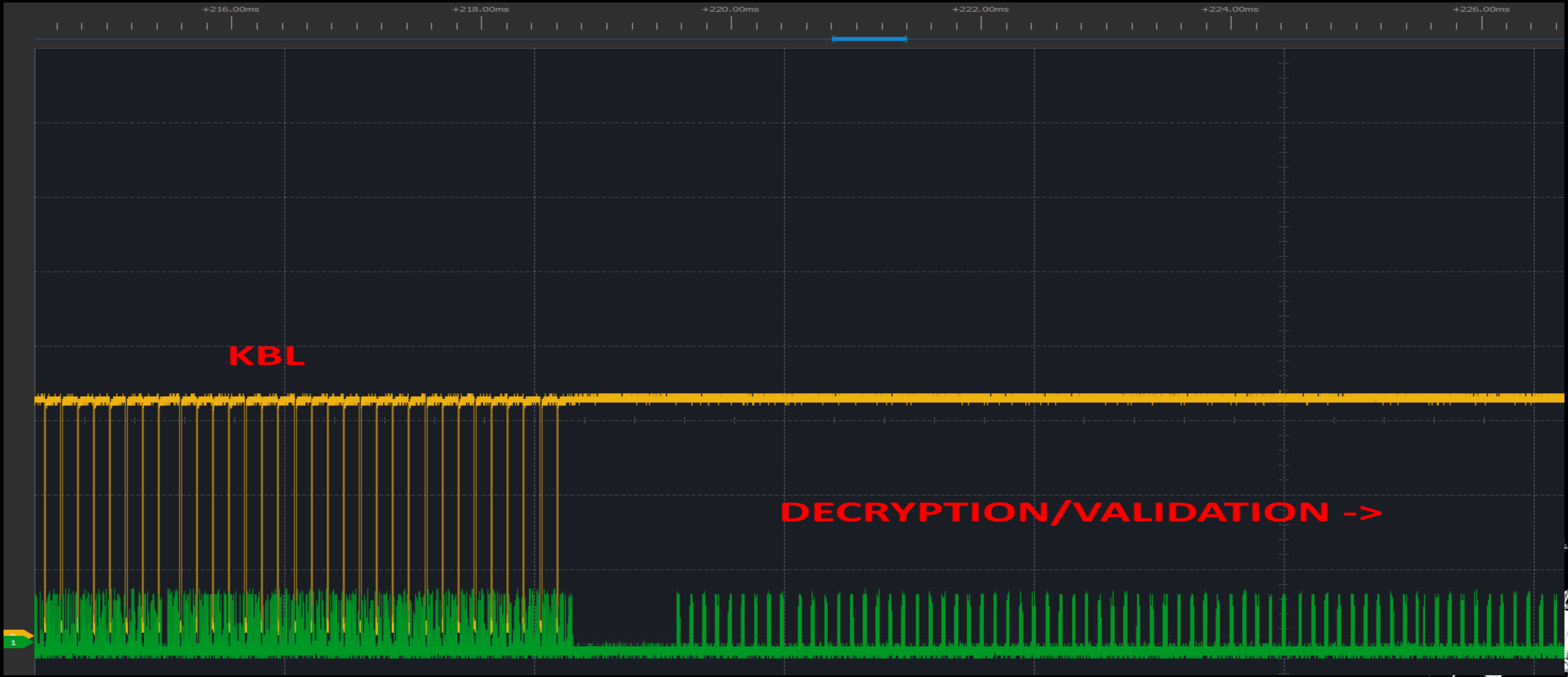
```
                          [EAP  ] ====================================================
                                                                                     [E
AP  ] Copyright (C) 2013 Sony Computer Entertainment Inc.
                                                          [EAP  ] r26694 2013/07/18 21:
49:29.61
        [EAP  ] EAP SDK VERSION : 1.000.0
                                          [EAP  ] -------------------------------------
--------------
```

UART pinout on motherboard http://jaicrab.org/?&a=Ps4/Tools/UART

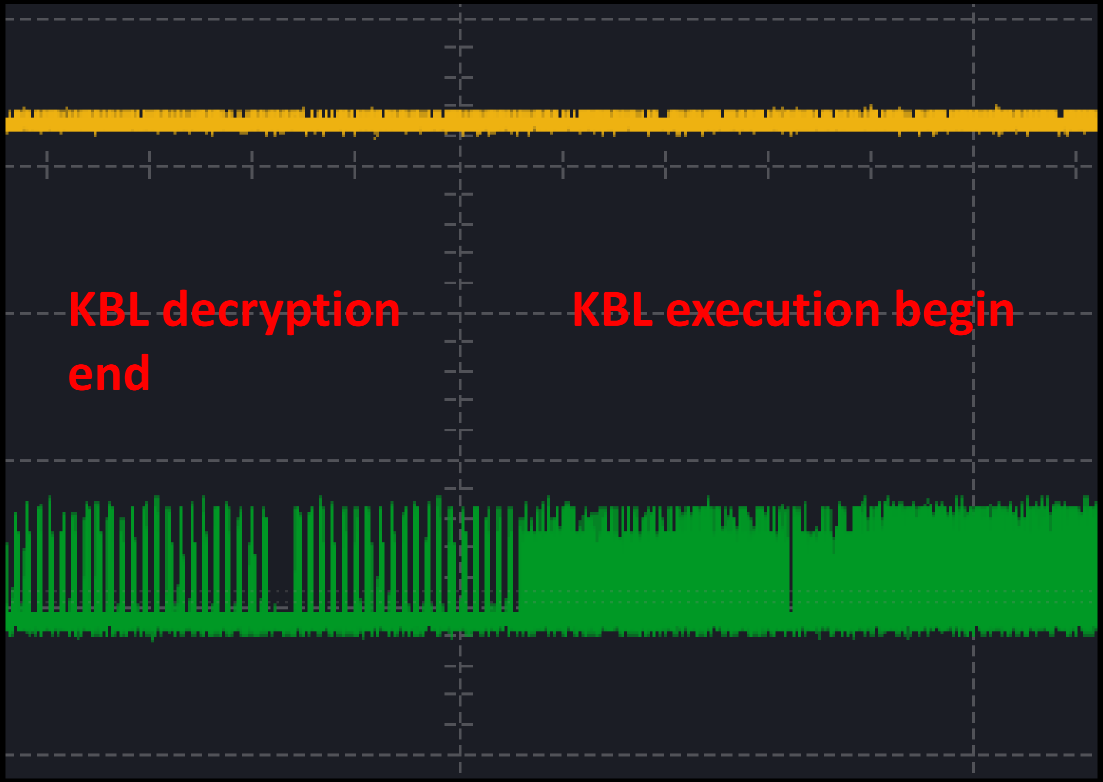
SPI.CS and CKE analysis



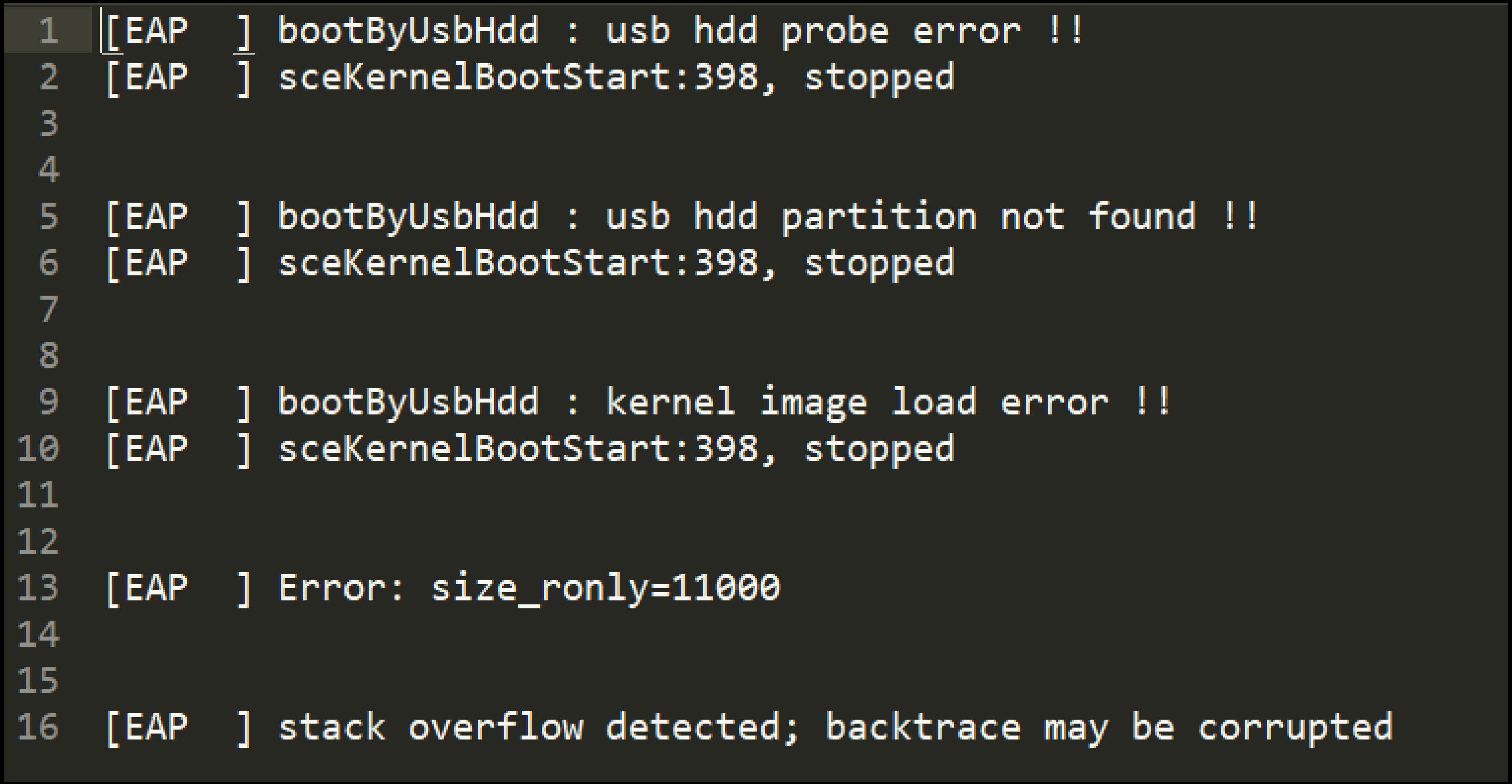
SPI.CS analysis



Glitch after KBL decryption

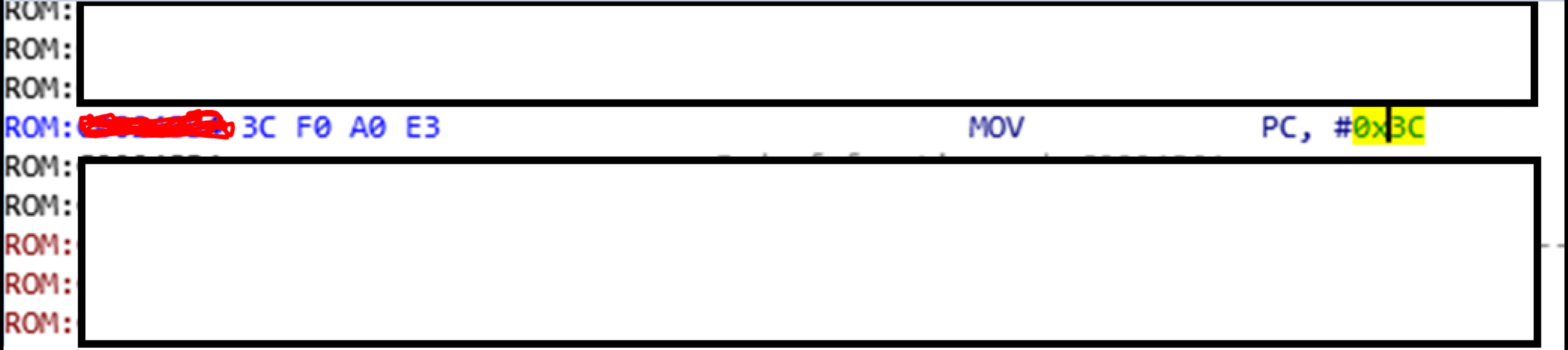


KBL message glitch debug



```
[EAP  ] bootByUsbHdd : usb hdd probe error !!
[EAP  ] sceKernelBootStart:398, stopped


[EAP  ] bootByUsbHdd : usb hdd partition not found !!
[EAP  ] sceKernelBootStart:398, stopped


[EAP  ] bootByUsbHdd : kernel image load error !!
[EAP  ] sceKernelBootStart:398, stopped


[EAP  ] Error: size_ronly=11000


[EAP  ] stack overflow detected; backtrace may be corrupted
```

Single instruction injection



```
ROM:[redacted]
ROM:
ROM:
ROM:[redacted] 3C F0 A0 E3                MOV             PC, #0x3C
ROM:[redacted]
ROM:
ROM:
ROM:
ROM:
```

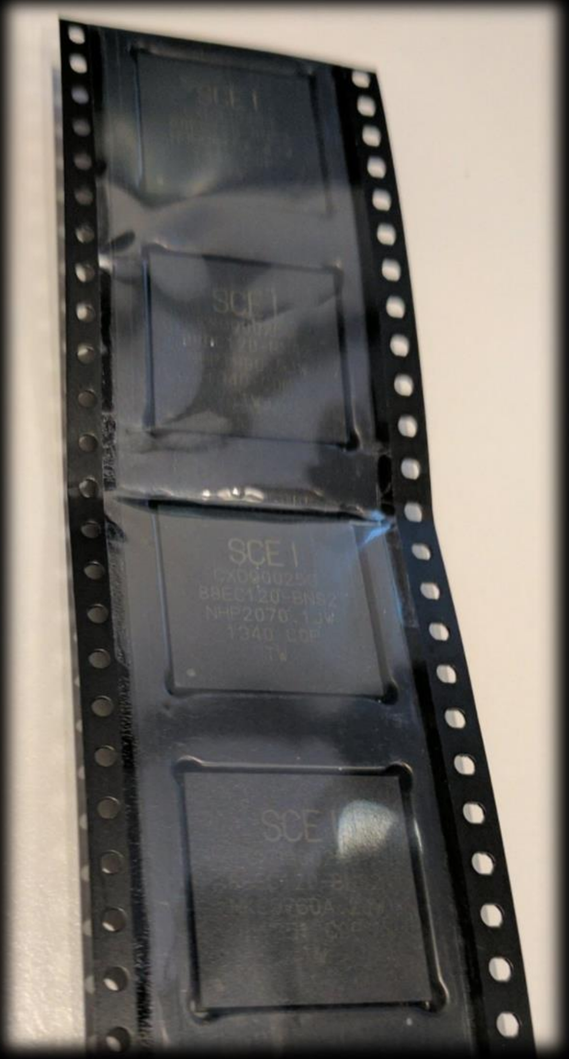
DEMO!

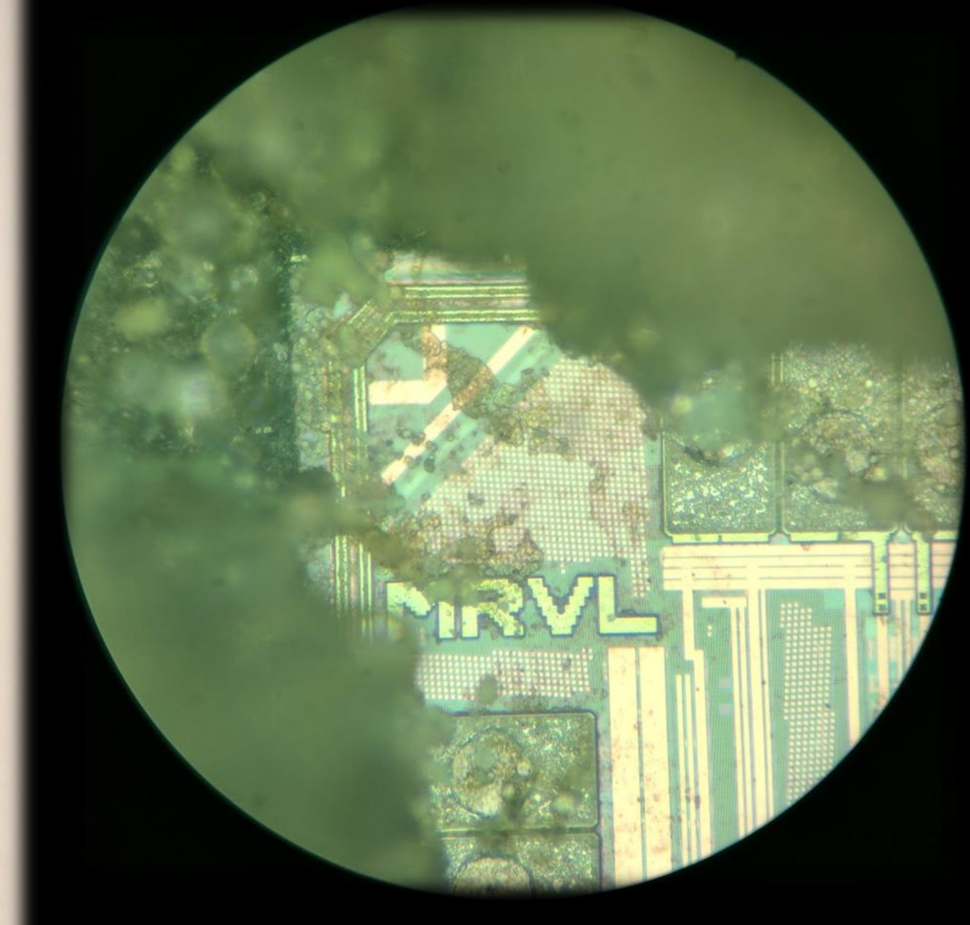
https://youtu.be/sMroXa-zYxk

Conclusions

- **Why it worked?**
  - **I don’t now ¯_(ツ)_/¯**
  - **KBL decryption is not in place**
    - KBL decryption doesn’t overwrite itself could be related to KBL image parsing etc.
  - **CPU cache**
    - No all transactions were committed
    - Probably should use uncached memory accesses
- Don’t hardcode HMAC and use same HMAC on every platform
- Don’t trust external memory

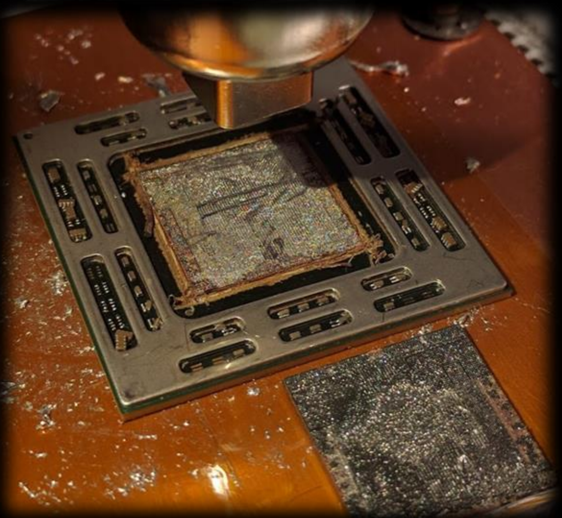
Marvell SoC

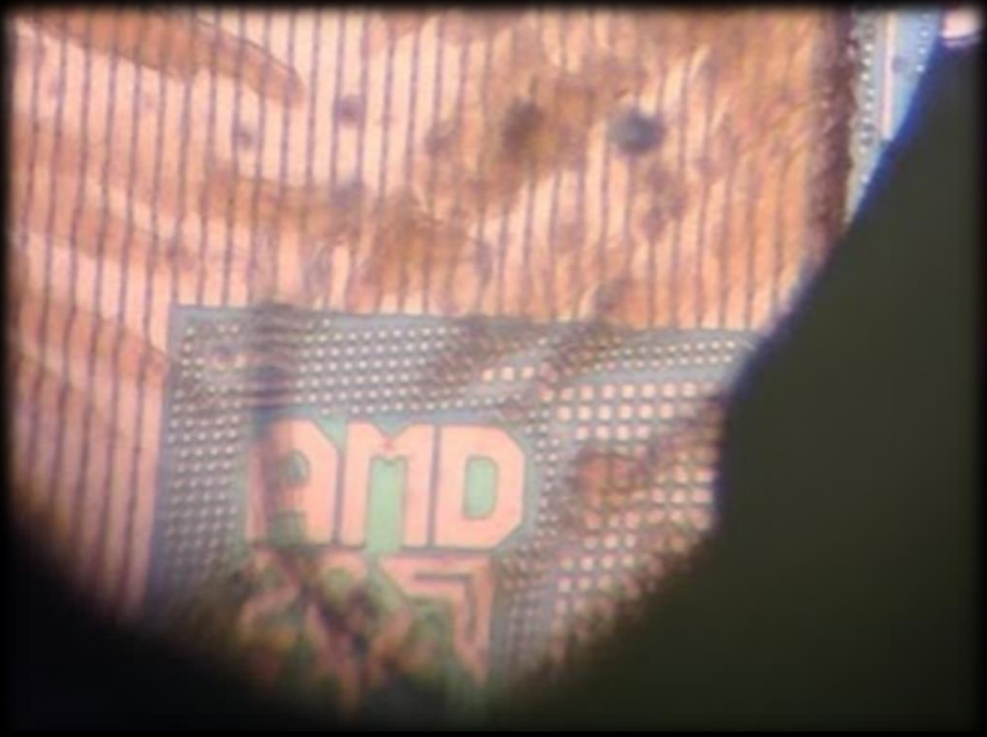


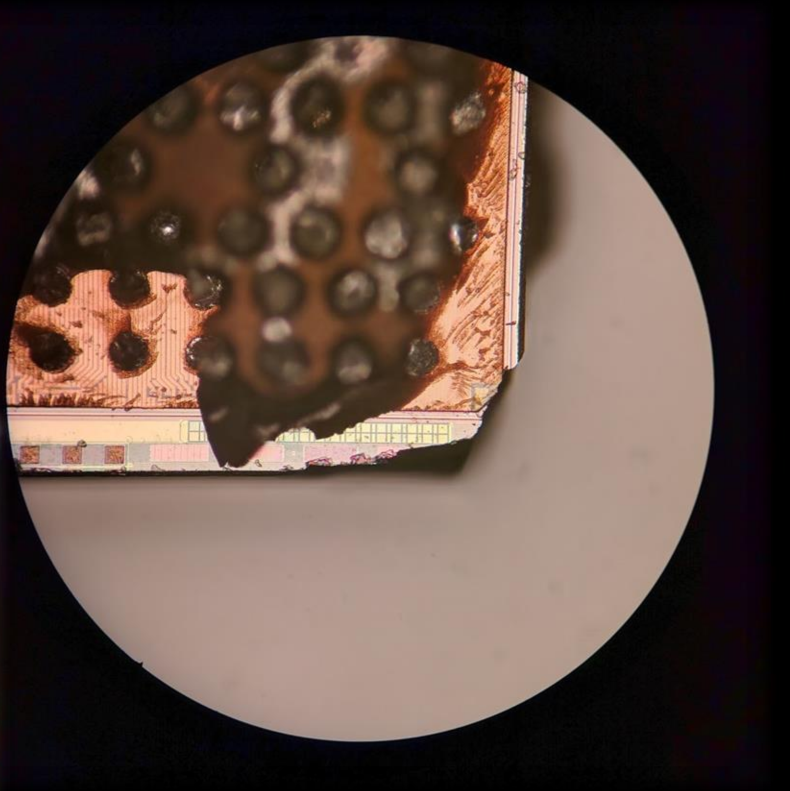


- eBay and Ali is your friend.
- Much larger feature size 180nm?
- Would take a lot of time and ROI is unknown.

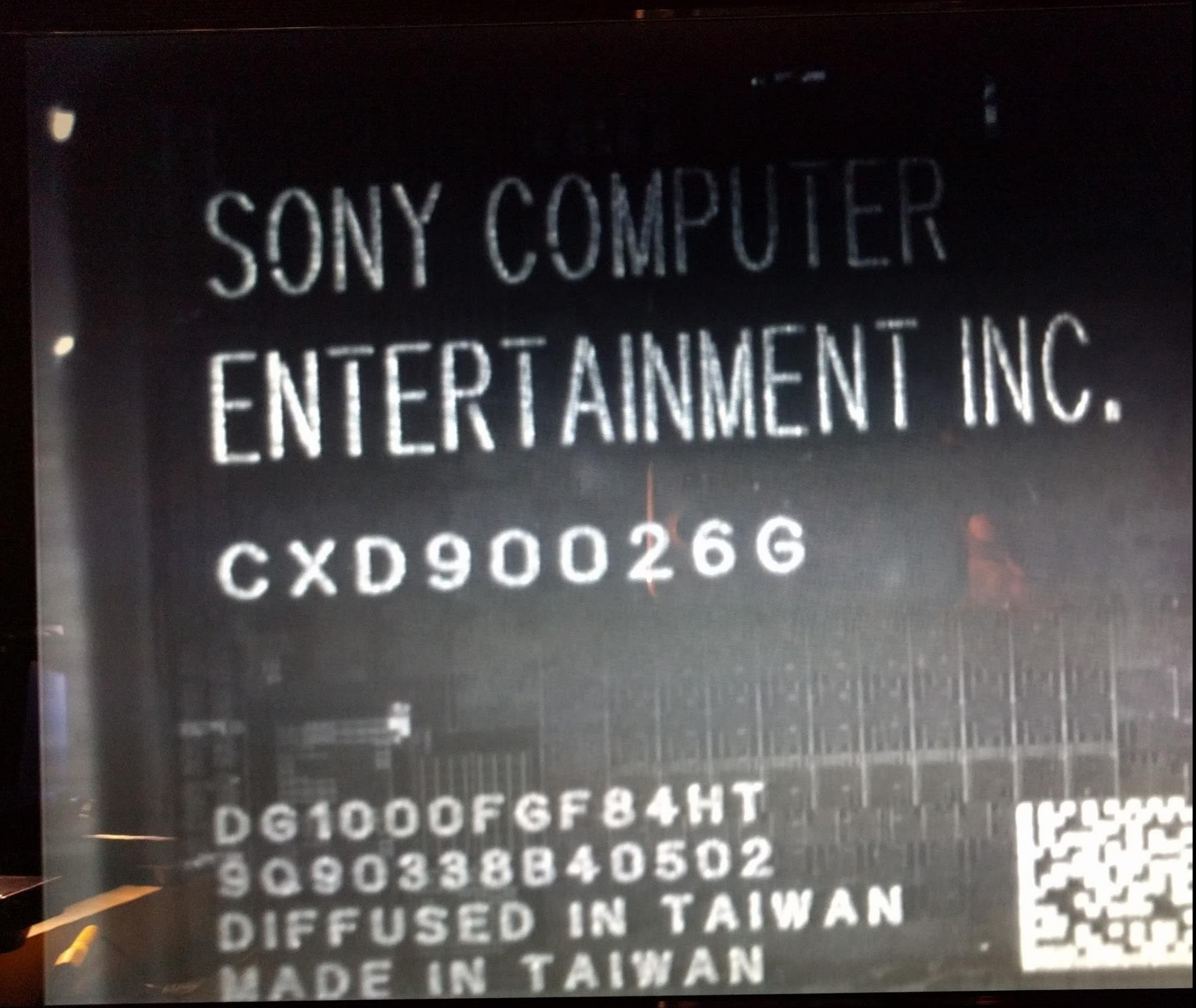
AMD APU decapsulation







- We need SEM things are really small 28nm!



IR maybe?

- AMD(TSMC) silicon lacks doping it is susceptible to backside analysis using IR light.
- Laser fault injection is possible!
- Requires sophisticated optical stage.

Questions?