# LCDPwn: Breaking Enterprise-Things with Layer 2 Discovery Protocol Vulnerabilities Again

Qian Chen | December 2022

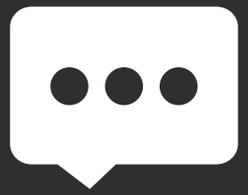

# About Me

- Senior security engineer from Codesafe Team of Legendsec at QI-ANXIN Group
- Mainly focus on the IoT and protocol security
- Speaker at conferences: POC2019, HITB2021AMS, POC2022

cq674350529

# Agenda

Introduction



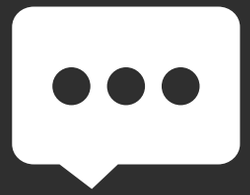
CDP & LLDP Protocol

LCDPwn Research

Summary

# Agenda

Introduction



CDP & LLDP Protocol

LCDPwn Research

Summary

# What is LCDPwn?

- Series of zero-day vulnerabilities in layer-2 protocol (CDP/LLDP)
- Affecting several CDP and LLDP components
- Affecting nearly 70 kinds of devices from 30 well-known vendors

LCDPwn

# What is LCDPwn?

| | | | |
|---|---|---|---|
| CVE-2022-20846 | CVE-2022-20766 | CVE-2022-20691 | CVE-2022-20690 |
| CVE-2022-20689 | CVE-2022-20688 | CVE-2022-20687 | CVE-2022-20686 |
| CVE-2021-46082 | CVE-2021-34780 | CVE-2021-34779 | CVE-2021-34778 |
| CVE-2021-34776 | CVE-2021-34775 | CVE-2021-34774 | CVE-2021-34734 |
| CVE-2021-34618 | CVE-2021-33317 | CVE-2021-33316 | CVE-2021-33315 |
| CVE-2021-29219 | CVE-2021-28801 | CVE-2021-26111 | CVE-2021-25849 |
| CVE-2021-25848 | CVE-2021-25847 | CVE-2021-25846 | CVE-2021-25845 |
| CVE-2021-20024 | CVE-2021-1598 | CVE-2021-1597 | CVE-2021-1596 |
| CVE-2021-1595 | CVE-2021-1564 | CVE-2021-1563 | CVE-2021-1521 |
| CVE-2021-1379 | CVE-2021-1309 | CVE-2021-1308 | CVE-2021-1251 |
| CVE-2021-1131 | CVE-2021-0277 | CVE-2020-3544 | CVE-2020-3543 |
| CVE-2020-3507 | CVE-2020-3506 | CVE-2020-3505 | |

# What is LCDPwn?

CVE-2022-20846 CVE-2022-20766 CVE-2022-20691 CVE-2022-20690
CVE-2022-20689 CVE-2022-20688 CVE-2022-20687 CVE-2022-20686
CVE-2021-46082 CVE-2021-34780 CVE-2021-34779 CVE-2021-34778
CVE-2021-34775 CVE-2021-34774 CVE-2021-34734
CVE-2021-33315
CVE-2021-29219 CVE-2021-28801 CVE-2021-26111 CVE-2021-25849
CVE-2021-25848 CVE-2021-25847 CVE-2021-25846 CVE-2021-25845
CVE-2021-1596
CVE-2021-1595 CVE-2021-1564 CVE-2021-1563 CVE-2021-1521
CVE-2021-1379 CVE-2021-1309 CVE-2021-1308 CVE-2021-1251
CVE-2021-1131 CVE-2021-0277 CVE-2020-3544 CVE-2020-3543
CVE-2020-3507 CVE-2020-3506 CVE-2020-3505

- Out-of-Bounds Read
- Out-of-Bounds Write
- Buffer Overflow (Stack/Heap/Bss)
- Memory Leak
- Integer Underflow
- Integer Overflow
- NULL Pointer Dereference
- Reachable Assertion

# What is LCDPwn?

- Cisco Discovery Protocol (CDP): Cisco proprietary protocol

- Link Layer Discovery Protocol (LLDP): vendor-neutral protocol

| | | | |
|---|---|---|---|
| lldpd[1] | OpenLLDP[2] | openvswitch/lldp[3] | ocelot-vsc6825/lldp[4] |
| l2g_lldp.ko | switchdrvr | ISS.exe | |

1. https://lldpd.github.io/
2. https://openlldp.sourceforge.net/
3. https://github.com/openvswitch/ovs/tree/master/lib/lldp
4. https://github.com/microchip-ung/ocelot-vsc6825/tree/master/src/lldp

# What is LCDPwn?

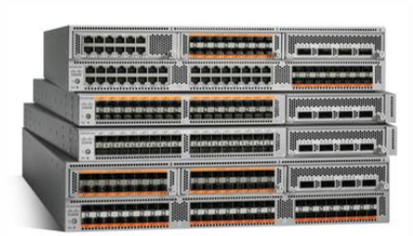
# Background

CDPwn: 5 zero-days in the implementation of Cisco's CDP protocol

| CVE | Title | CVSS v3 |
|---|---|---|
| CVE-2020-3110 | Cisco Video Surveillance 8000 Series IP Cameras Cisco Discovery Protocol Remote Code Execution and Denial of Service Vulnerability | 8.8 |
| CVE-2020-3111 | Cisco IP Phone Remote Code Execution and Denial of Service Vulnerability | 8.8 |
| CVE-2020-3118 | Cisco IOS XR Software Cisco Discovery Protocol Format String Vulnerability | 8.8 |
| CVE-2020-3119 | Cisco NX-OS Software Cisco Discovery Protocol Remote Code Execution Vulnerability | 8.8 |
| CVE-2020-3120 | Cisco FXOS, IOS XR, and NX-OS Software Cisco Discovery Protocol Denial of Service Vulnerability | 7.4 |

1. https://www.armis.com/research/cdpwn/

## Affecting a wide array of Cisco products



Cisco NX-OS Switches

Cisco NCS Systems

Cisco 8000 IP Cameras Series

Cisco IP Phone 7800 Series

Cisco Firepower Firewalls

Cisco IOS XR Routers

Cisco IP Phone 8800 Series

1. https://www.armis.com/research/cdpwn/

# Motivation


- Were these vulnerabilities well fixed?
- Was there any variant?
- What about other vendors, or it’s just Cisco’s own business?

# Agenda

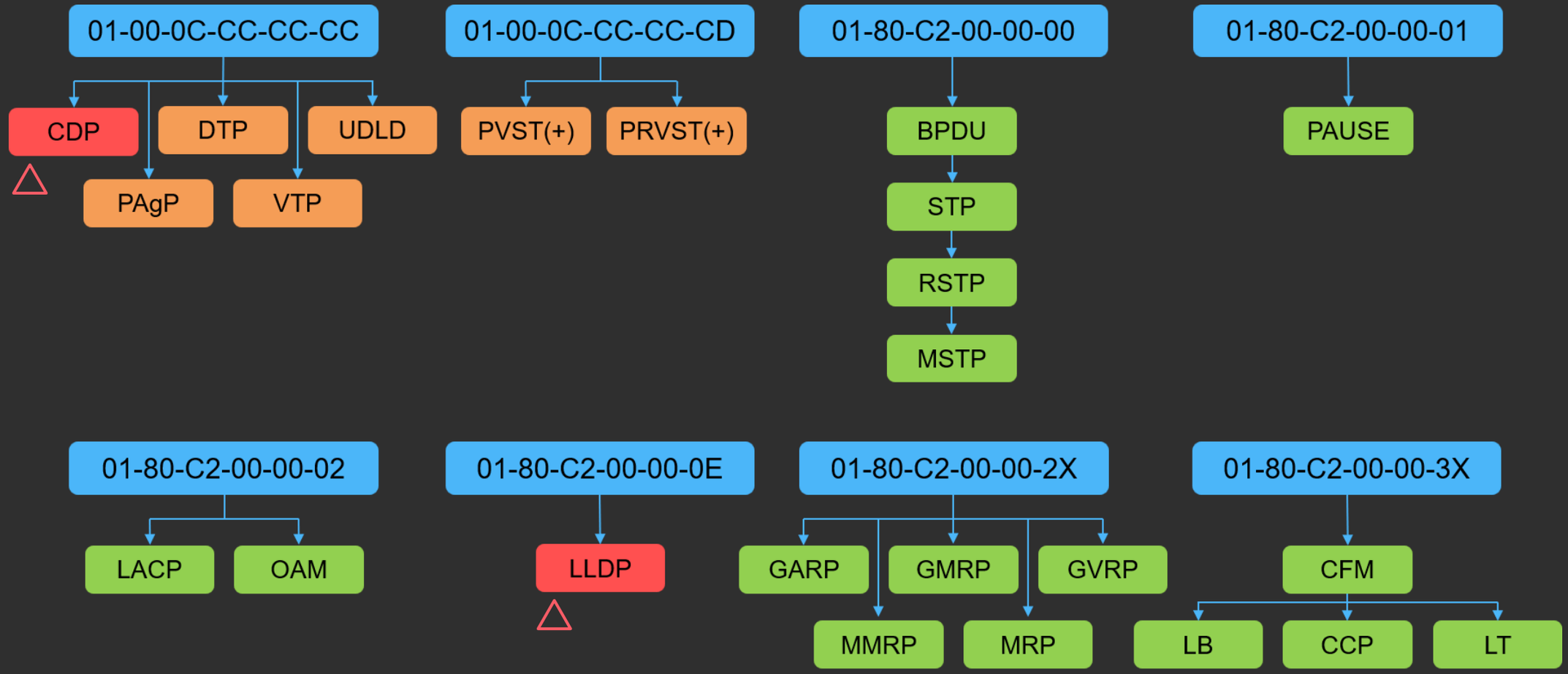
Introduction

CDP & LLDP Protocol

LCDPwn Research

Summary

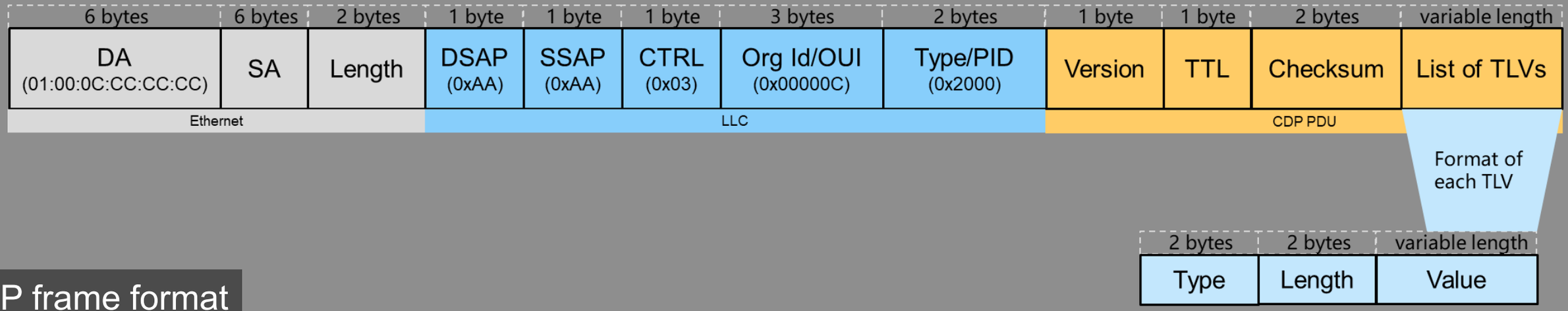
# Common Layer 2 Protocol




1. https://info.armis.com/rs/645-PDC-047/images/Armis-CDPwn-WP.pdf

# Cisco Discovery Protocol (CDP)

- A Cisco proprietary protocol used for collecting directly connected neighbor device information
  - CDPv1: initial version, capable only to collect device information connected to next end
  - CDPv2: the most recent release, providing more intelligent device tracking features




CDP frame format

1. https://learningnetwork.cisco.com/s/article/cisco-discovery-protocol-cdp-x

| Field | Description | |
|---|---|---|
| Version | The version of CDP being used | |
| Time-to-Live (TTL) | The amount of time, in seconds, that a receiver should retain the information contained in this packet. Default in 180 sec | |
| Checksum | The standard IP checksum | |
| Type/Length/Value (TLV) | Type | Possible CDP TLV types |
| | Length | The total length in bytes of the type, length and value fields |
| | Value | Contains value/data of type |

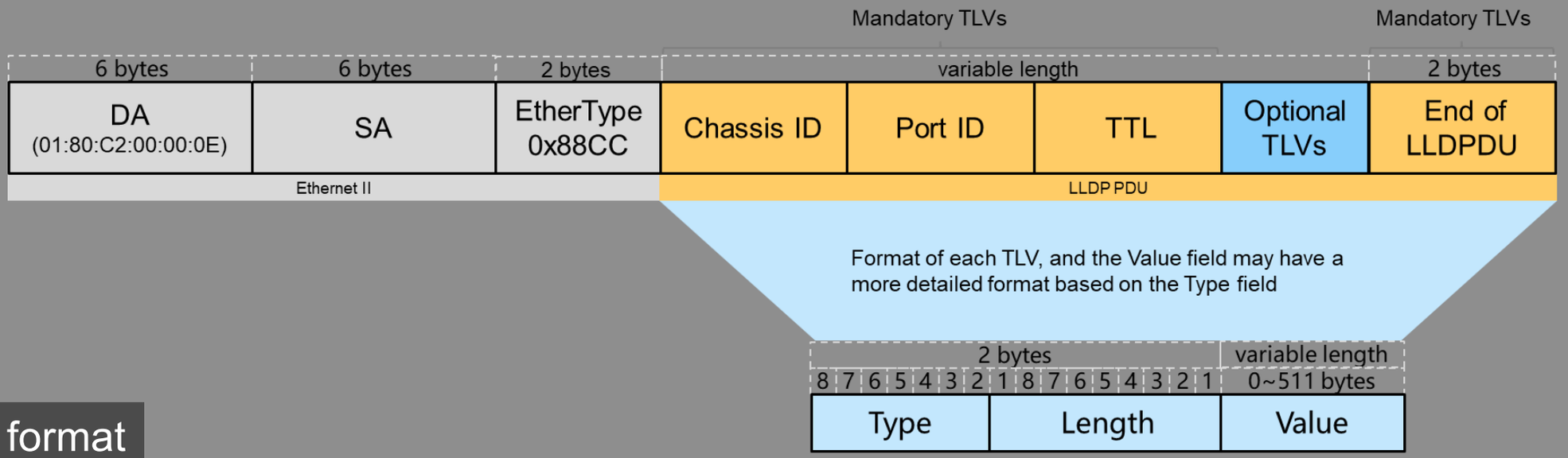
1. https://learningnetwork.cisco.com/s/article/cisco-discovery-protocol-cdp-x

| Type | TLV | Description |
|---|---|---|
| 0x0001 | Device ID TLV | Identifies the device name |
| 0x0002 | Addresses TLV | Contains network addresses of both receiving and sending devices |
| 0x0004 | Capabilities TLV | Identifies the device type, which indicates its functional capability |
| 0x0005 | Software Version TLV | Identifies the software version |
| 0x0006 | Platform TLV | Identifies the hardware platform |
| 0x0007 | IP Network Prefix TLV | Contains a list of network prefixes used for IP packets forwarding |
| 0x0009 | VTP Management Domain TLV | Advertises the configured VTP-management-domain name |
| 0x000a | Native VLAN TLV | Indicates the assumed VLAN for untagged packets on the interface |
| 0x000b | Duplex TLV | Indicates the duplex configuration of the CDP broadcast interface |
| 0x0017 | Location TLV | Delivers location-based information to endpoint devices |
| ... | ... | ... |

1. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cdp/configuration/15-mt/cdp-15-mt-book/nm-cdp-discover.html

# Link Layer Discovery Protocol (LLDP)

- A vendor-neutral protocol that allows a network device to advertise its identity and capabilities on the local network




LLDP frame format

1. https://learningnetwork.cisco.com/s/article/link-layer-discovery-protocol-lldp-x

| Type | TLV | Description |
|---|---|---|
| 0x00 | End of LLDPDU TLV | Marks the end of the TLV sequence |
| 0x01 | Chassis ID TLV | Identifies the chassis |
| 0x02 | Port ID TLV | Identifies the port component of the MSAP identifier |
| 0x03 | Time to Live (TTL) TLV | The amount of time, in seconds, that a receiver should regard the information associated with this MSAP identifier to be valid |
| 0x04 | Port Description TLV | Identifies the port's description |
| 0x05 | System Name TLV | Identifies the system's assigned name |
| 0x06 | System Description TLV | Identifies the system's description |
| 0x07 | System Capabilities TLV | Identifies the primary functions of the system and whether or not these primary functions are enabled |
| 0x08 | Management Address TLV | Indicates an address used to assist discovery by network management |
| 0x7f | Organizationally Specific TLV | Used to advertise information by different organizations |

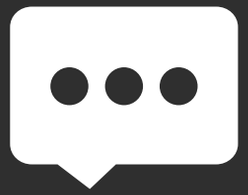
1. https://learningnetwork.cisco.com/s/article/link-layer-discovery-protocol-lldp-x

# Differences

| | CDP | LLDP |
|---|---|---|
| TLV Order | - | Starts with Chassis ID, Port ID and Time to Live TLV in correct order, ends with the End Of LLDPDU TLV if present |
| TLV Length | [0x4, 0xffff], including the length of type and length fields | [0x0, 0x1ff], excluding the length of type and length fields |

# Agenda

Introduction



CDP & LLDP Protocol

LCDPwn Research

Summary

# Learn from CDPwn

analyze 5 zero-day vulnerabilities to learn vulnerability patterns

# Vulnerability Patterns

- Lack of TLV length field validation
- Improper use of protocol_type variable
- Lack of number_of_address field validation
- Lack of power request count validation
- Improper use of device_id/port_id/software_version value
- Discrepancy between the size used for allocation and copy operation

# Learn from CDPwn

variant analysis: analyze the code to parse CDP packets again

# Results

20+ new bugs found in 5 types of devices

| Device | Bugs |
|---|---|
| Cisco NX-OS Switches | 5 |
| Cisco IP Phones | 4 |
| Cisco IOS XR Routers | 3 |
| Cisco 8000 Series IP Cameras | 8 |
| Cisco ATA 190 Series Analog Telephone Adapter | 5 |

- Out-of-Bounds Read
- Integer Underflow

- Out-of-Bounds Read
- Out-of-Bounds Write

- CVE-2022-20846

- CVE-2021-1521
- CVE-2021-1131
- CVE-2020-3544
- CVE-2020-3543
- CVE-2020-3507
- CVE-2020-3506
- CVE-2020-3505

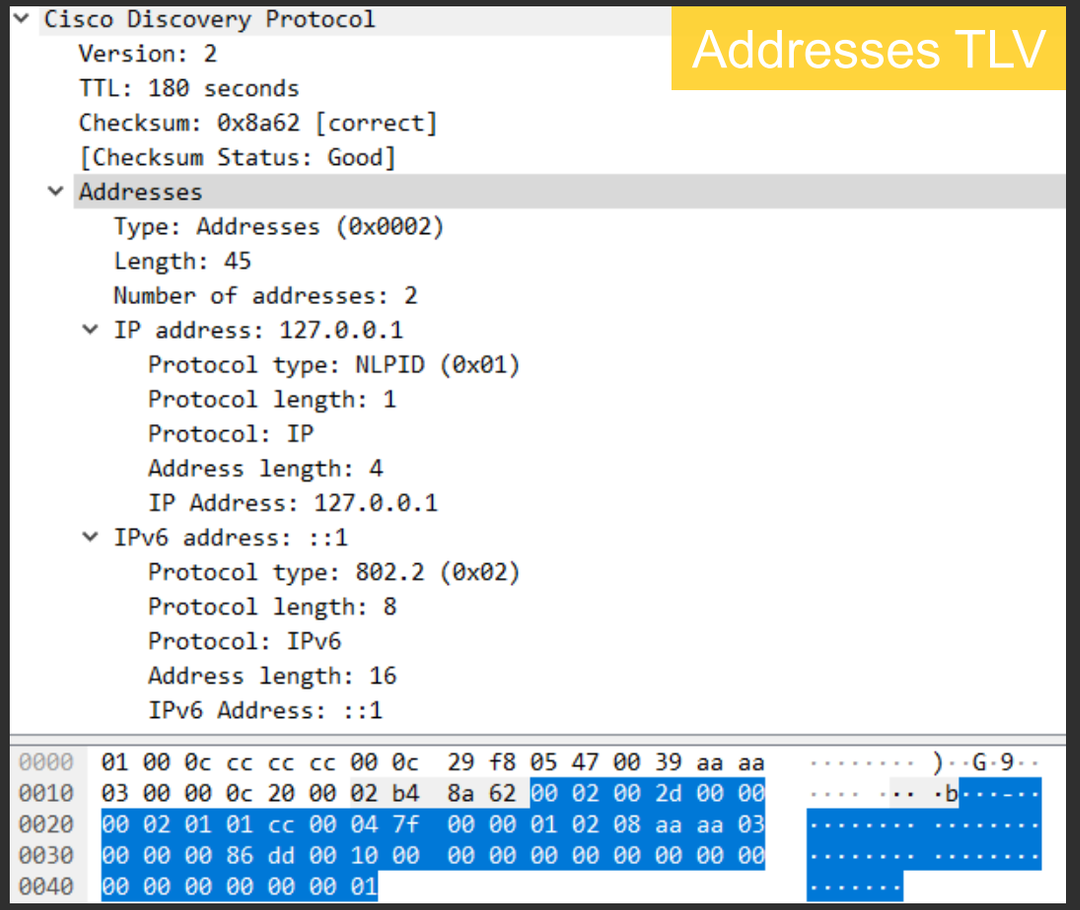
- CVE-2022-20766
- CVE-2022-20691
- CVE-2022-20690
- CVE-2022-20689
- CVE-2022-20688

# Case Study

# Cisco IP Phone

## #1 Out-of-bounds Write

- Triggered by sending a CDP packet containing multiple Addresses TLV
- IP Conference Phone 7832：Release 14.1(1)



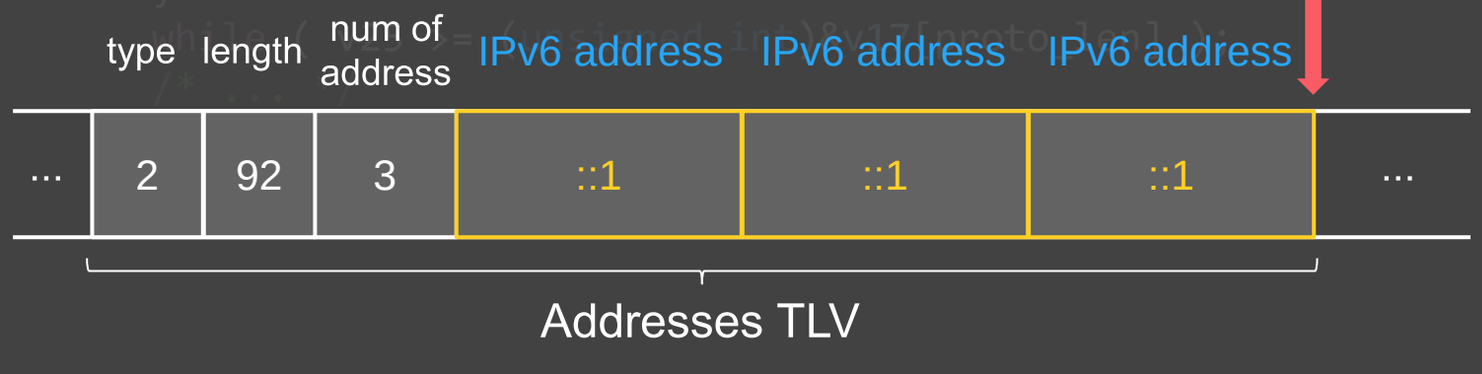
```
int cdpRcvPkt(int pCmdMsg)
{
  /* ... */
  v71 = 0;  //(1)
LABEL_123:
  cdp_pdu_end_ptr = (unsigned int)&cdp_pdu_ptr[cdp_pdu_len];
  tlv_type = HIBYTE(dest) | (unsigned __int16)(dest << 8);  // get tlv_type
  tlv_length = HIBYTE(v83) | (unsigned __int16)(v83 << 8);  // get tlv_length
  switch ( tlv_type )
  {
    case 2:  // Addresses TLV
      /* ... length check ... */
      num_of_addr = HIBYTE(v80) | (v80 << 24) | ((v80 & 0xFF0000) >> 8) | ((v80 & 0xFF00) << 8);
      v80 = num_of_addr;
      /* ... length check ... */
      *(v18 - 110) = cdp_pdu_cur_ptr[8];        // get proto_type
      proto_len = (unsigned __int8)cdp_pdu_cur_ptr[9];// get proto_len
      /* ... length check ... */
      v78 = cdp_pdu_cur_ptr + 2;
      v23 = 0;
      v24 = v71;
      v25 = (unsigned int)&cdp_pdu_ptr[cdp_pdu_len];
      break;
  }
```

```
  do {
    /* ... get proto and addr_len ... */
    if ( addr_len == 4 ) { memcpy(&s[10 * v24 + 63], v27, sizeof(int)); }  // get ipv4_addr
    else { memcpy(&s[10 * v24 + 63], v26 + 2, addr_len); }  //(2) get ipv6_addr
    // condition: addr_len == 4 || (v34 = ip_addr[0] & 0xE0) != 0 && v34 != 0xE0
    if ( *((_WORD *)v30 - 46) == 4 || (v34 = s[10*v24+63] & 0xE0) != 0 && v34 != 0xE0 )
      ++v24;  //(3)
    if ( v24 > 1 ) {
LABEL_59:
      v12 += v83 - 2;
LABEL_122:
      cdp_pdu_cur_ptr = v12;
      goto LABEL_123;
    }
    if ( ++v23 == num_of_addr ) {
      goto LABEL_59;
    }
    /* ... */
  }
  while ( v25 >= (unsigned int)&v17[proto_len] );
  /* ... */
```

```
do {
  /* ... get proto and addr_len ... */
  if ( addr_len == 4 ) { memcpy(&s[10 * v24 + 63], v27, sizeof(int)); }  // get ipv4_addr
  else { memcpy(&s[10 * v24 + 63], v26 + 2, addr_len); }  //(2) get ipv6_addr
  // condition: addr_len == 4 || (v34 = ip_addr[0] & 0xE0) != 0 && v34 != 0xE0
  if ( *((_WORD *)v30 - 46) == 4 || (v34 = s[10*v24+63] & 0xE0) != 0 && v34 != 0xE0 )
    ++v24;  //(3)
  if ( v24 > 1 ) {
LABEL_59:
    v12 += v83 - 2;
LABEL_122:
    cdp_pdu_cur_ptr = v12;
    goto LABEL_123;
  }
  if ( ++v23 == num_of_addr ) {
    goto LABEL_59;
  }
  /* ... */
}
while ( v23 < (unsigned int)&v17[proto_len] );
```

Case 1

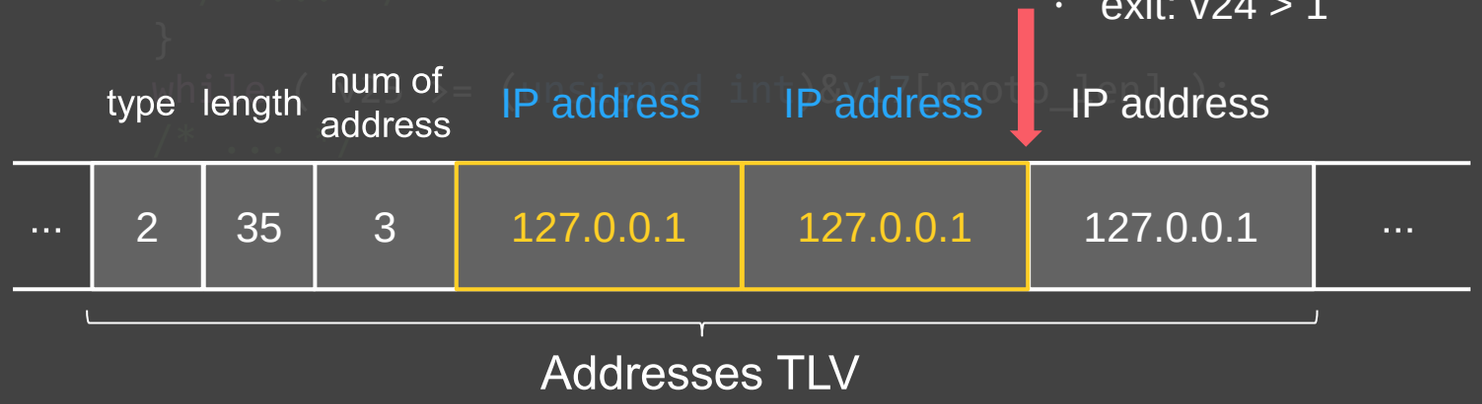
- v24 = 0
- exit: v23 == num_of_addr



```
do {
  /* ... get proto and addr_len ... */
  if ( addr_len == 4 ) { memcpy(&s[10 * v24 + 63], v27, sizeof(int)); }  // get ipv4_addr
  else { memcpy(&s[10 * v24 + 63], v26 + 2, addr_len); }  //(2) get ipv6_addr
  // condition: addr_len == 4 || (v34 = ip_addr[0] & 0xE0) != 0 && v34 != 0xE0
  if ( *((_WORD *)v30 - 46) == 4 || (v34 = s[10*v24+63] & 0xE0) != 0 && v34 != 0xE0 )
    ++v24;  //(3)
  if ( v24 > 1 ) {
LABEL_59:
    v12 += v83 - 2;
LABEL_122:
    cdp_pdu_cur_ptr = v12;
    goto LABEL_123;
  }
  if ( ++v23 == num_of_addr ) {
    goto LABEL_59;
```

Case 2

- v24 = 2
- exit: v24 > 1



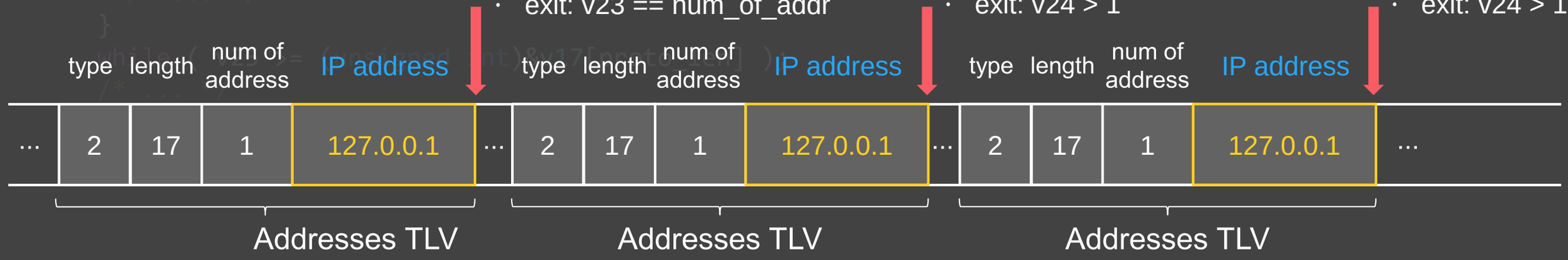
```
do {
  /* ... get proto and addr_len ... */
  if ( addr_len == 4 ) { memcpy(&s[10 * v24 + 63], v27, sizeof(int)); }  // get ipv4_addr
  else { memcpy(&s[10 * v24 + 63], v26 + 2, addr_len); }  //(2) get ipv6_addr
  // condition: addr_len == 4 || (v34 = ip_addr[0] & 0xE0) != 0 && v34 != 0xE0
  if ( *((_WORD *)v30 - 46) == 4 || (v34 = s[10*v24+63] & 0xE0) != 0 && v34 != 0xE0 )
    ++v24;  //(3)
  if ( v24 > 1 ) {
LABEL_59:
    v12 += v83 - 2;
LABEL_122:
    cdp_pdu_cur_ptr = v12;
    goto LABEL_123;
  }
  if ( ++v23 == num_of_addr ) {
    goto LABEL_59;
```

Case 3

- v24 = 1
- exit: v23 == num_of_addr

- v24 = 2
- exit: v24 > 1

- v24 = 3
- exit: v24 > 1



# Cisco IOS XR Router

# #2 CVE-2022-20846

Heap Overflow

## Cisco IOS XR Software Cisco Discovery Protocol Denial of Service Vulnerability

**Medium**

| | | | |
|---|---|---|---|
| **Advisory ID:** | cisco-sa-xr-cdp-wnALzvT2 | CVE-2022-20846 | Download CSAF |
| **First Published:** | 2022 September 14 16:00 GMT | CWE-120 | Download CVRF |
| **Version 1.0:** | Final | | Email |
| **Workarounds:** | No workarounds available | | |
| **Cisco Bug IDs:** | CSCwb23263 | | |
| **CVSS Score:** | Base 4.3 | | |

### Summary

A vulnerability in the Cisco Discovery Protocol implementation for Cisco IOS XR Software could allow an unauthenticated, adjacent attacker to cause the Cisco Discovery Protocol process to reload on an affected device.

This vulnerability is due to a heap buffer overflow in certain Cisco Discovery Protocol messages. An attacker could exploit this vulnerability by sending a malicious Cisco Discovery Protocol packet to an affected device. A successful exploit could allow the attacker to cause a heap overflow, which could cause the Cisco Discovery Protocol process to reload on the device. The bytes that can be written in the buffer overflow are restricted, which limits remote code execution.

Target release: IOS XR 7.4.2

**Note:** Cisco Discovery Protocol is a Layer 2 protocol. To exploit this vulnerability, an attacker

Addresses TLV

```
Cisco Discovery Protocol
    Version: 2
    TTL: 180 seconds
    Checksum: 0x8a62 [correct]
    [Checksum Status: Good]
  Addresses
      Type: Addresses (0x0002)
      Length: 45
      Number of addresses: 2
    IP address: 127.0.0.1
        Protocol type: NLPID (0x01)
        Protocol length: 1
        Protocol: IP
        Address length: 4
        IP Address: 127.0.0.1
    IPv6 address: ::1
        Protocol type: 802.2 (0x02)
        Protocol length: 8
        Protocol: IPv6
        Address length: 16
        IPv6 Address: ::1
```

```
0000  01 00 0c cc cc cc 00 0c  29 f8 05 47 00 39 aa aa   ........ )..G.9..
0010  03 00 00 0c 20 00 02 b4  8a 62 00 02 00 2d 00 00   .... ... .b...-..
0020  00 02 01 01 cc 00 04 7f  00 00 01 02 08 aa aa 03   ........ ........
0030  00 00 00 86 dd 00 10 00  00 00 00 00 00 00 00 00   ........ ........
0040  00 00 00 00 00 00 01                              .......
```

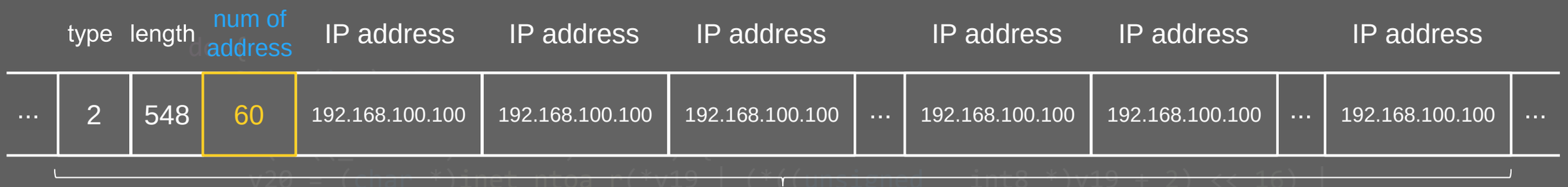
```
__int64 cdp_events_entry_common(__int64 a1, int a2)
{
  /* ... */
  v6 = (char *)calloc(1uLL, v5);  // 0x49e
  if ( v6 )
  {
    v7 = v6;
    /* ... */
    v8 = (const char *)sub_AAF0(a1);
    snprintf(v7 + 28, 0x1EuLL, "%s", v8);
    v9 = (const char *)sub_A8C0(a1);
    snprintf(v7 + 58, 0x28uLL, "%s", v9);
    v10 = (const char *)sub_A920(a1);
    snprintf(v7 + 98, 0x20uLL, "%s", v10);
    /* ... */
    num_of_addr = *(unsigned __int16 *)(a1 + 112);
    *((_DWORD *)v11 + 289) = num_of_addr;
    if ( num_of_addr && v15 )
    {
      /* ... */
```

Patch for CVE-2020-3118

```
do {
  v27 = (int)v17;
  v28 = &v18[(int)v17];
  if ( *((_BYTE *)v19 - 2) == 1 ) {
    v20 = (char *)inet_ntoa_r(*v19 | (*((unsigned __int8 *)v19 + 2) << 16) |
                                     (*((unsigned __int8 *)v19 + 3) << 24));
    v21 = v20;
  } else {
    v20 = s;
    xr_inet_ntop(10LL, (__int64)v19, (__int64)s, '.');
    v21 = s;
  }
  v22 = strlen(v21);
  v23 = &v18[v27 + 2];
  v24 = v22;
  memcpy(v23, v20, v22);  //(1)
  LODWORD(v17) = v27 + v24 + 2;
  v18 = v36;
  v25 = (int)v17;
  LODWORD(v17) = v17 + 1;
  v36[v25] = ' ';  //(2)
  v19 += 12;
  v26 = (num_of_addr - 1 < 0) ^ __OFADD__(-1, num_of_addr) | (num_of_addr == 1);
  --num_of_addr;  //(3)
} while ( !v26 );  //(4)
/* ... */
```

Convert network address into a character string

The loop condition depends on the num_of_addr field




```
      v21 = v20;
    } else {
      v20 = s;
      xr_inet_ntop(10LL, (__int64)v19, (__int64)s, '.');
      v21 = s;
    }
    v22 = strlen(v21);
    v23 = &v18[v27 + 2];
    v24 = v22;
    memcpy(v23, v20, v22);  //(1) heap overflow
    LODWORD(v17) = v27 + v24 + 2;
    v18 = v36;
    v25 = (int)v17;
    LODWORD(v17) = v17 + 1;
    v36[v25] = ' ';  //(2)
    v19 += 12;
    v26 = (num_of_addr - 1 < 0) ^ __OFADD__(-1, num_of_addr) | (num_of_addr == 1);
    --num_of_addr;  //(3)
  } while ( !v26 );  //(4)
  /* ... */
```

Convert network address into a character string

The loop condition depends on the num_of_addr field

# Cisco 8000 Series IP Camera

## #3 CVE-2021-1131

Heap Overflow

veillance 8000 Series IP Cameras Cisco Discovery Protocol Denial of Service Vulnerability

Medium

| | | | |
|---|---|---|---|
| **Advisory ID:** | cisco-sa-ipcameras-dos-9zdZcUfq | CVE-2021-1131 | Download CSAF |
| **First Published:** | 2021 January 13 16:00 GMT | CWE-119 | Download CVRF |
| **Version 1.0:** | Final | | Email |
| **Workarounds:** | No workarounds available | | |
| **Cisco Bug IDs:** | CSCvv72651 | | |
| **CVSS Score:** | Base 6.5 | | |

### Summary

A vulnerability in the Cisco Discovery Protocol implementation for Cisco Video Surveillance 8000 Series IP Cameras could allow an unauthenticated, adjacent attacker to cause an affected IP camera to reload.

The vulnerability is due to missing checks when Cisco Discovery Protocol messages are processed. An attacker could exploit this vulnerability by sending a malicious Cisco Discovery Protocol packet to an affected IP camera. A successful exploit could allow the attacker to cause the affected IP camera to reload unexpectedly, resulting in a denial of service (DoS) condition.

Target release: CIVS-IPC-8020-V1.0.9-5

**Note:** Cisco Discovery Protocol is a Layer 2 protocol. To exploit this vulnerability, an attacker

Addresses TLV

```
Cisco Discovery Protocol
    Version: 2
    TTL: 180 seconds
    Checksum: 0x8a62 [correct]
    [Checksum Status: Good]
  Addresses
      Type: Addresses (0x0002)
      Length: 45
      Number of addresses: 2
    IP address: 127.0.0.1
        Protocol type: NLPID (0x01)
        Protocol length: 1
        Protocol: IP
        Address length: 4
        IP Address: 127.0.0.1
    IPv6 address: ::1
        Protocol type: 802.2 (0x02)
        Protocol length: 8
        Protocol: IPv6
        Address length: 16
        IPv6 Address: ::1
```

```
0000  01 00 0c cc cc cc 00 0c  29 f8 05 47 00 39 aa aa   ........ )..G.9..
0010  03 00 00 0c 20 00 02 b4  8a 62 00 02 00 2d 00 00   .... ... .b...-..
0020  00 02 01 01 cc 00 04 7f  00 00 01 02 08 aa aa 03   ........ ........
0030  00 00 00 86 dd 00 10 00  00 00 00 00 00 00 00 00   ........ ........
0040  00 00 00 00 00 00 01                              .......
```

```
int cdpd_handle_addr_tlv(int a1, void **a2, int tlv_length, int tlv_value_ptr)
{
  v6 = (tlv_length - 8) << 16;  //(1) byte order conversion
  /* ... calculate v4 based on num_of_addr field ... */
  result = (int)malloc(v4);  //(2)
  a2[2] = (void *)result;
  if ( result ) {
    while ( 1 ) {  //(3) parse the IP address items and copy its contents into allocated memory
      v20 = HIWORD(v6);
      if ( !v20 )
        goto exit;
      v21 = (v20 - 2) << 16;
      /* ... */
      if ( v21 < 0 )
        goto exit;
      /* ... */
      if ( v26 || ( /* ... */ v29 = (v28 - 2) << 16, v26 = v29 < 0, v26) ) {
exit:
        return 1;
      }
      /* ... */
      v16 = (unsigned __int16)(v31 - addr_len);
      v6 = v16 << 16;
      if ( (v16 & 0x8000) != 0 )
        goto exit;
      /* ... */
```

Length validation

Length validation

Length validation

Length validation

Discrepancy: allocated memory depends on num_of_addr field, while the loop condition depends on tlv_length field

```
LABEL_26:
      memcpy(v14, v34, v17);  //(4)
      /* ... */
      if ( !v16 ) {
        return 1;
      }
    }
    if ( proto_type == 2 ) {
      /* ... */
      if ( v36 ) {
        /* ... handle difference cases and goto LABEL_26 */
        v40 = memcmp(proto_ptr, (char *)&protocol_appletalk + v37, proto_len_field);
        if ( !v40 && v45 == 3 ) {
LABEL_41:
          v16 = (unsigned __int16)(v31 - addr_len);
          v6 = v16 << 16;
          if ( (v16 & 0x8000) != 0 )
            goto exit;
LABEL_25:
          v17 = addr_len;
          goto LABEL_26;
        }
        /* ... */
      }
    }
```

Length validation

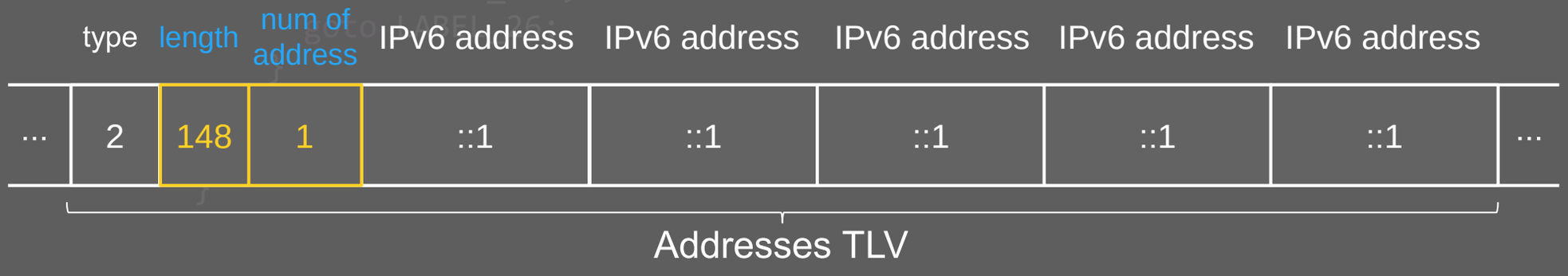
Length validation

```
LABEL_26:
      memcpy(v14, v34, v17);  //(4) heap overflow
      /* ... */
      if ( !v16 ) {
        return 1;
      }
    }
    if ( proto_type == 2 ) {
      /* ... */
      if ( v36 ) {
        /* ... handle difference cases and goto LABEL_26 */
        v40 = memcmp(proto_ptr, (char *)&protocol_appletalk + v37, proto_len_field);
        if ( !v40 && v45 == 3 ) {
LABEL_41:
          v16 = (unsigned __int16)(v31 - addr_len);
          v6 = v16 << 16;
          if ( (v16 & 0x8000) != 0 )
            goto exit;
LABEL_25:
          v17 = addr_len;
          goto LABEL_26;
```

Discrepancy: allocated memory depends on num_of_addr field, while the loop condition depends on tlv_length field

Length validation

Length validation



# What We Learnt

- Vulnerability patterns
  - Lack of bounds checks
  - Integer wraparounds
  - Discrepancy between check and use
  - State confusion
- Better understanding of the CDP protocol format

# From CDP to LLDP

CDP is a Cisco proprietary protocol, while LLDP is a vendor-neutral protocol

# Research Methodology

Vendor Selection

Device Selection

Bug Hunting

# Research Methodology

## Vendor Selection

## Device Selection

## Bug Hunting

- Documents
  - Product catalogue
  - Device specification
- Firmware database
  - Characteristic strings

Check if it supports LLDP

| | | |
|---|---|---|
| | | v1, v2, v3 |
| | | • |
| Performance | Bandwidth | 12.8 Gbps |
| | Buffer Memory | 4 MB Per Syst |
| | Jumbo Frames | • |
| Availability | Spanning Tree 802.1d/RSTP .1w/MSTP .1s | 802.1d, .1w, .1 |
| | RPS Redundant Power Supply | |
| | Modular Power Supply | |
| Traffic Management | Number of 802.1q VLAN/Number of VLAN Group | 128 |
| | Link Aggregation 802.3ad/LACP | • |
| | Discovery Protocol LLDP 802.1ab | • |
| | Rate Limiting | • |
| | Broadcast Storm Control | • |

| Networking Standards | |
|---|---|
| IP Version | IPv4, IPv6 |
| Ethernet Standards | 802.3af PoE<br>802.3at PoE+<br>802.3 10Base-T<br>802.3u 100Base-T<br>802.3ab 1000Base-T<br>802.3ae 10GBase-X<br>802.3x Flow Control<br>802.1AB LLDP |

"lldp" "tlv error" "wrong tlv length" "invalid sub type" "chassis tlv" ...

# Research Methodology

| Vendor Selection | Device Selection | Bug Hunting |
| --- | --- | --- |
|  |  |  |

- Vendor
  - Well-known
  - Open to bug reports
- Firmware
  - Available without extra contract or license needed
  - Can be unpacked easily or with small efforts

# Research Methodology

| Vendor Selection | Device Selection | Bug Hunting |
| --- | --- | --- |

- Static analysis
  - Reverse engineering
- Dynamic analysis
  - Partial emulation
- Packet construction


# Research Results

- Several common LLDP components found in firmware

| | | | |
|---|---|---|---|
| lldpd[1] | OpenLLDP[2] | openvswitch/lldp[3] | ocelot-vsc6825/lldp[4] |
| l2g_lldp.ko | switchdrvr | ISS.exe | |

1. https://lldpd.github.io/
2. https://openlldp.sourceforge.net/
3. https://github.com/openvswitch/ovs/tree/master/lib/lldp
4. https://github.com/microchip-ung/ocelot-vsc6825/tree/master/src/lldp

# Research Results

- Several common LLDP components found in firmware
- 50+ bugs found in a wide variety of devices from well-known vendors

| Component | Bugs |
|---|---|
| lldpd | 3 |
| OpenLLDP | 4 |
| openvswitch/lldp | 6 |
| ocelot-vsc6825/lldp | 2 |
| l2g_lldp.ko | 6 |
| switchdrvr | 3 |
| ISS.exe | 3 |
| Custom | 26 |

- Buffer Overflow (Stack/Heap/Bss)
- Integer Underflow
- Memory Leak
- Null Pointer Dereference
- Out-of-Bounds Read
- Double Free

# Research Results

- Several common LLDP components found in firmware
- 50+ bugs found in a wide variety of devices from well-known vendors

| | | | |
|---|---|---|---|
| CVE-2022-20687 | CVE-2022-20686 | CVE-2021-46082 | CVE-2021-34780 |
| CVE-2021-34779 | CVE-2021-34778 | CVE-2021-34776 | CVE-2021-34775 |
| CVE-2021-34774 | CVE-2021-34734 | CVE-2021-34618 | CVE-2021-33317 |
| CVE-2021-33316 | CVE-2021-33315 | CVE-2021-29219 | CVE-2021-28801 |
| CVE-2021-26111 | CVE-2021-25849 | CVE-2021-25848 | CVE-2021-25847 |
| CVE-2021-25846 | CVE-2021-25845 | CVE-2021-20024 | CVE-2021-1598 |
| CVE-2021-1597 | CVE-2021-1596 | CVE-2021-1595 | CVE-2021-1564 |
| CVE-2021-1563 | CVE-2021-1379 | CVE-2021-1309 | CVE-2021-1308 |
| CVE-2021-1251 | CVE-2021-0277 | | |

# Research Limitations

- The affected LLDP components existing in firmware may not be actually used on the device
- Due to the selection strategies, some vendors or devices may be omitted

# Detailed Analysis

# lldpd

*“* lldpd *is a 802.1AB implementation (LLDP) to help you locate neighbors of all your equipments.”*



1. https://lldpd.github.io/

# lldpd

## Past vulnerabilities

| CVE | Description | CVSS v3 | |
|---|---|---|---|
| CVE-2021-43612 | Heap overflow when parsing too short SONMP packets | - | |
| CVE-2020-27827 | Memory exhaustion attack through crafted LLDPU with duplicate TLVs | 7.5 | LLDP |
| CVE-2015-8012 | Assertion error when parsing malformed management address TLV | 7.5 | |
| CVE-2015-8011 | Buffer overflow when handling management address TLV for LLDP | 9.8 | |

# lldpd

## #4 CVE-2015-8011

```
Link Layer Discovery Protocol
  > Chassis Subtype = MAC address, Id: 00:18:ba:98:68:8f
  > Port Subtype = Locally assigned, Id: GE2
  > Time To Live = 120 sec
  v Management Address
      0001 000. .... .... = TLV Type: Management Address (8)
      .... ...0 0000 1100 = TLV Length: 12
      Address String Length: 5
      Address Subtype: IPv4 (1)
      Management Address: 127.0.0.1
      Interface Subtype: ifIndex (2)
      Interface Number: 1
      OID String Length: 0
  > End of LLDPDU
```

Management Address TLV

9 src/daemon/protocols/lldp.c

```
@@ -726,6 +726,11 @@ lldp_decode(struct lldpd *cfg, char *frame, int s,
                 case LLDP_TLV_MGMT_ADDR:
                         CHECK_TLV_SIZE(1, "Management address");
                         addr_str_length = PEEK_UINT8;
+                        if (addr_str_length > sizeof(addr_str_buffer)) {
+                                log_warnx("lldp", "too large management address on %s",
+                                    hardware->h_ifname);
+                                goto malformed;
+                        }
                         CHECK_TLV_SIZE(1 + addr_str_length, "Management address");
                         PEEK_BYTES(addr_str_buffer, addr_str_length);
                         addr_length = addr_str_length - 1;
@@ -734,7 +739,7 @@ lldp_decode(struct lldpd *cfg, char *frame, int s,
                         CHECK_TLV_SIZE(1 + addr_str_length + 5, "Management address");
                         iface_subtype = PEEK_UINT8;
                         iface_number = PEEK_UINT32;
```

# lldpd

## #4 CVE-2015-8011

Ruckus Unleashed AP R510: 200.7.10.202.118 (GA Refresh4) | 2020.06.01

```
signed int lldp_decode(int a1, _WORD *a2, unsigned int a3, int a4, _DWORD *a5, _DWORD *a6)
{
  /* ... */
  // case: Management Address TLV (type=0x8)
  if ( (v17 & 0x1FF) == 0
    || (mgmt_addr_str_len = *((unsigned __int8 *)lldp_pdu_ptr + 2),
        v31 = lldp_pdu_len - 3,
        v32 = (unsigned __int8 *)lldp_pdu_ptr + 3,
        LOBYTE(dword_33AE0) = mgmt_addr_str_len,
        tlv_length < mgmt_addr_str_len)
    || (memcpy(&dest, v32, mgmt_addr_str_len), v33=v31-mgmt_addr_str_len, tlv_length<=4))
  {
    log_warnx("lldp", "Management address TLV too short received on %s", v8);
    /* ... */
```

buffer overflow

# lldpd

## #4 CVE-2015-8011

Cisco RV132W ADSL2+ Wireless-N VPN Router: 1.0.1.14 | 2020.03.13

```
int lldp_decode(int a1, int a2, unsigned int a3, int a4, int *a5, _DWORD *a6)
{
  /* ... */
  // case: Management Address TLV
  if ( tlv_type == 8 )
  {
    if ( tlv_length )
    {
      addr_str_len = *(unsigned __int8 *)(tlv_ptr + 2);
      if ( tlv_length >= addr_str_len )
      {
        /* ... */
        memcpy(&v114, tlv_ptr + 3, addr_str_len);
        /* ... */
```

# lldpd

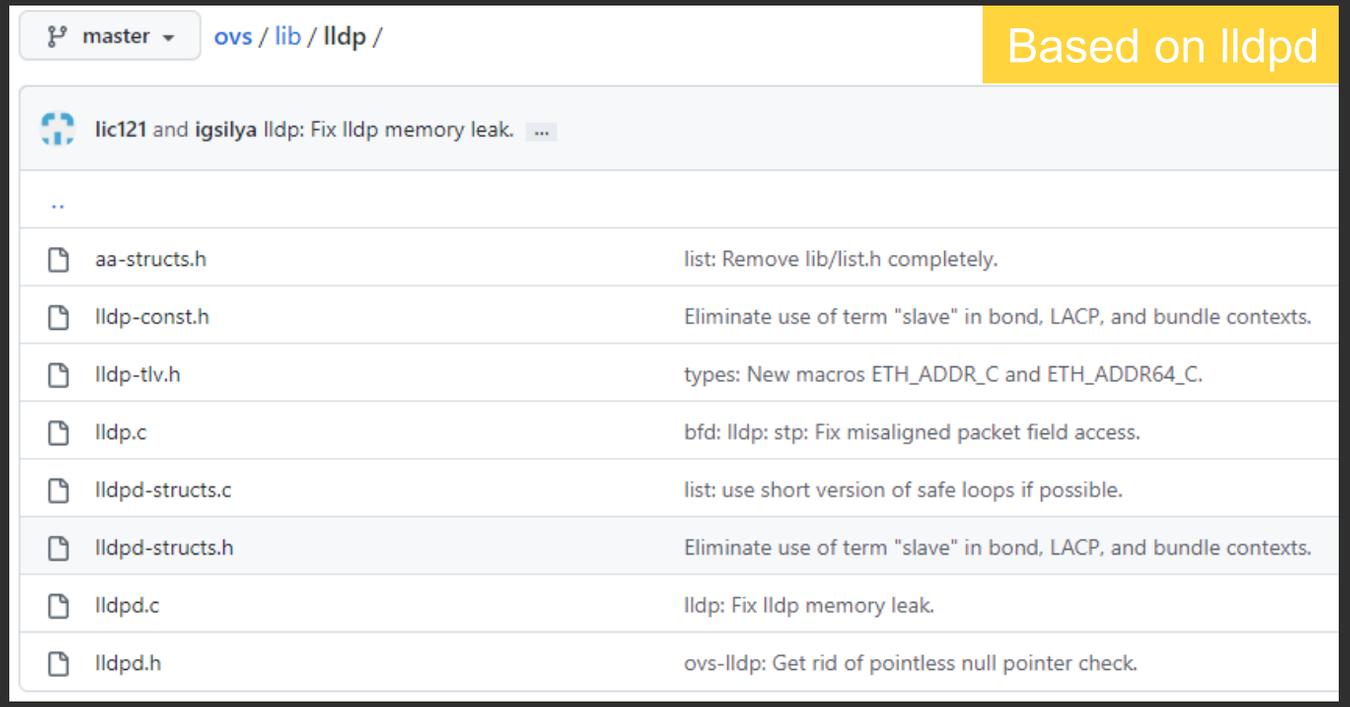
Past vulnerabilities

| CVE | Description | CVSS v3 | |
|---|---|---|---|
| CVE-2021-43612 | Heap overflow when parsing too short SONMP packets | - | |
| CVE-2020-27827 | Memory exhaustion attack through crafted LLDPU with duplicate TLVs | 7.5 | LLDP |
| CVE-2015-8012 | Assertion error when parsing malformed management address TLV | 7.5 | |
| CVE-2015-8011 | Buffer overflow when handling management address TLV for LLDP | 9.8 | |

*The n-day vulnerabilities from 6 years ago still survive in 2021.*

# openvswitch/lldp

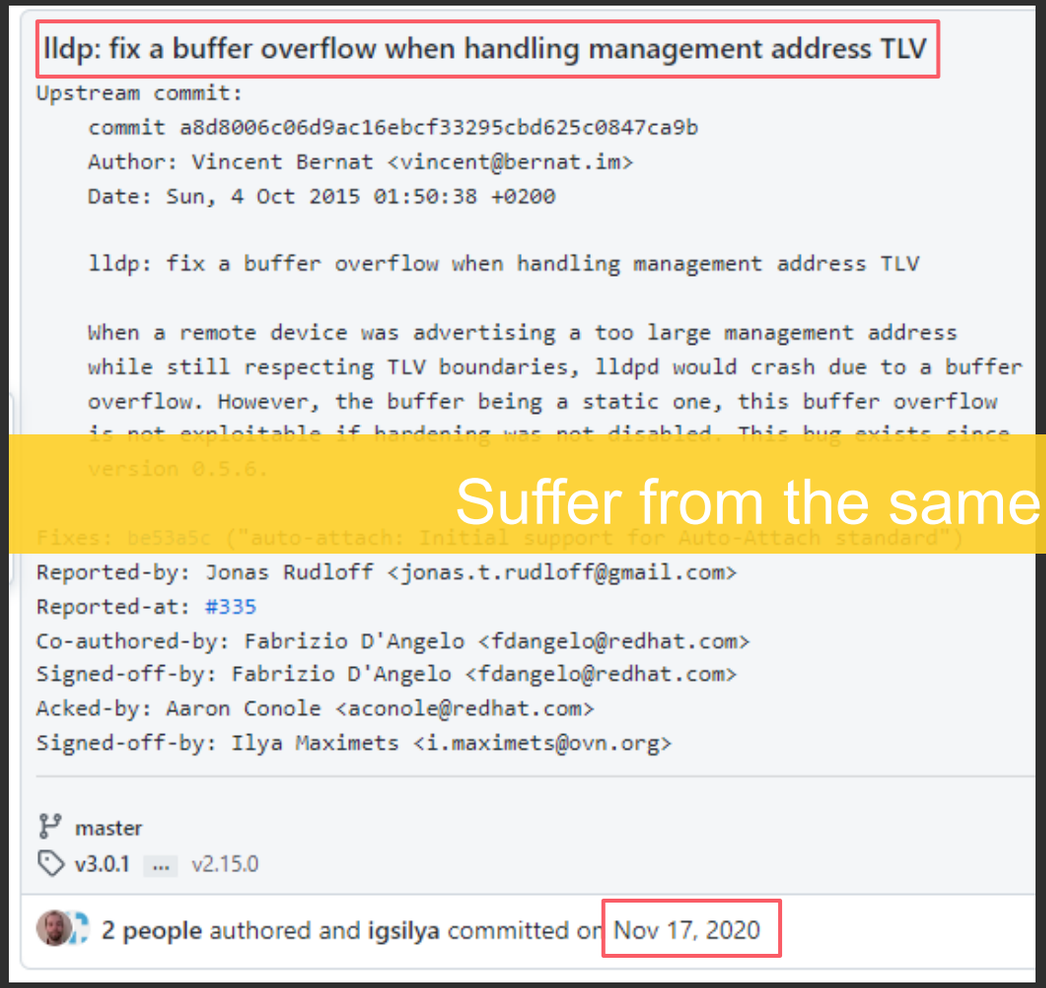
*“ Open vSwitch is suited to function as a virtual switch in VM environments. In addition to exposing standard control and visibility interfaces to the virtual networking layer, it was designed to support distribution across multiple physical servers. ”*

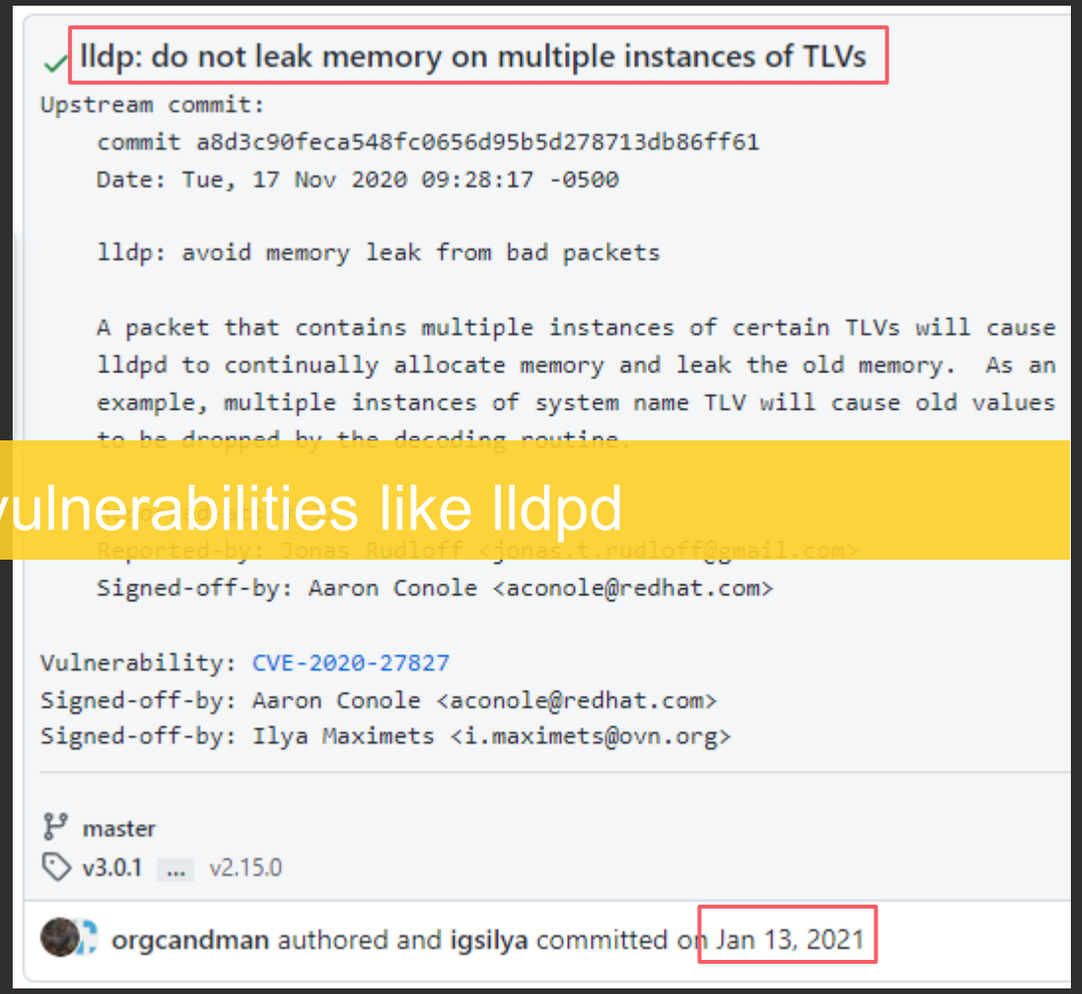



- Synology
- VMware NSX-T
- etc

1. https://github.com/openvswitch/ovs/tree/master/lib/lldp

# openvswitch/lldp




Suffer from the same vulnerabilities like lldpd

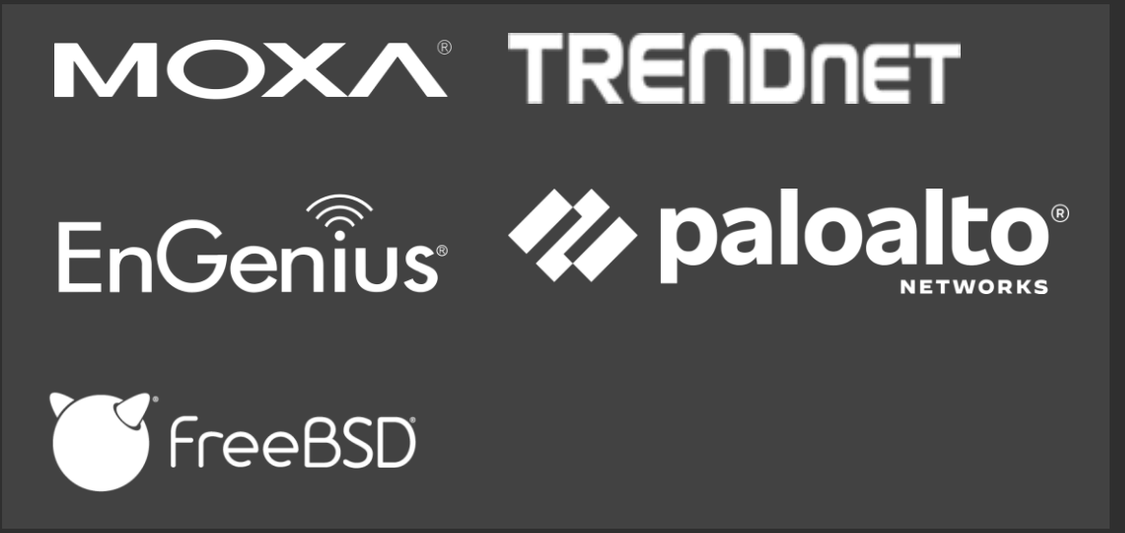
# openvswitch/lldp

## #5 Integer Underflow

```
int lldp_decode(struct lldpd *cfg OVS_UNUSED, char *frame, int s, ...) {
  /* ... */
  while (length && !gotend) {
    /* ... */
    switch (tlv_type) {
    case LLDP_TLV_ORG:   //type = 0x7f
      CHECK_TLV_SIZE(1 + sizeof orgid, “Organisational”); //(1) tlv_length >= 4
      PEEK_BYTES(orgid, sizeof orgid);
      tlv_subtype = PEEK_UINT8;
      if (memcmp(dot1, orgid, sizeof orgid) == 0) {
        /* ... */
      } else if (memcmp(avaya_oid, orgid, sizeof orgid) == 0) {  //orgid = 0x040d
        /* ... */
        switch(tlv_subtype) {  //(2)
        case LLDP_TLV_AA_ISID_VLAN_ASGNS_SUBTYPE:  //0xc
          PEEK_BYTES(&msg_auth_digest, sizeof msg_auth_digest);
          num_mappings = tlv_length - 4 - LLDP_TLV_AA_ISID_VLAN_DIGEST_LENGTH; //(3)
          /* ... */
          num_mappings /= 5; /* Each mapping is 5 Bytes */
          for(; num_mappings > 0; num_mappings--) {
            /* ... */
            PEEK_BYTES(isid, 3);  //(4) out-of-bounds read
```

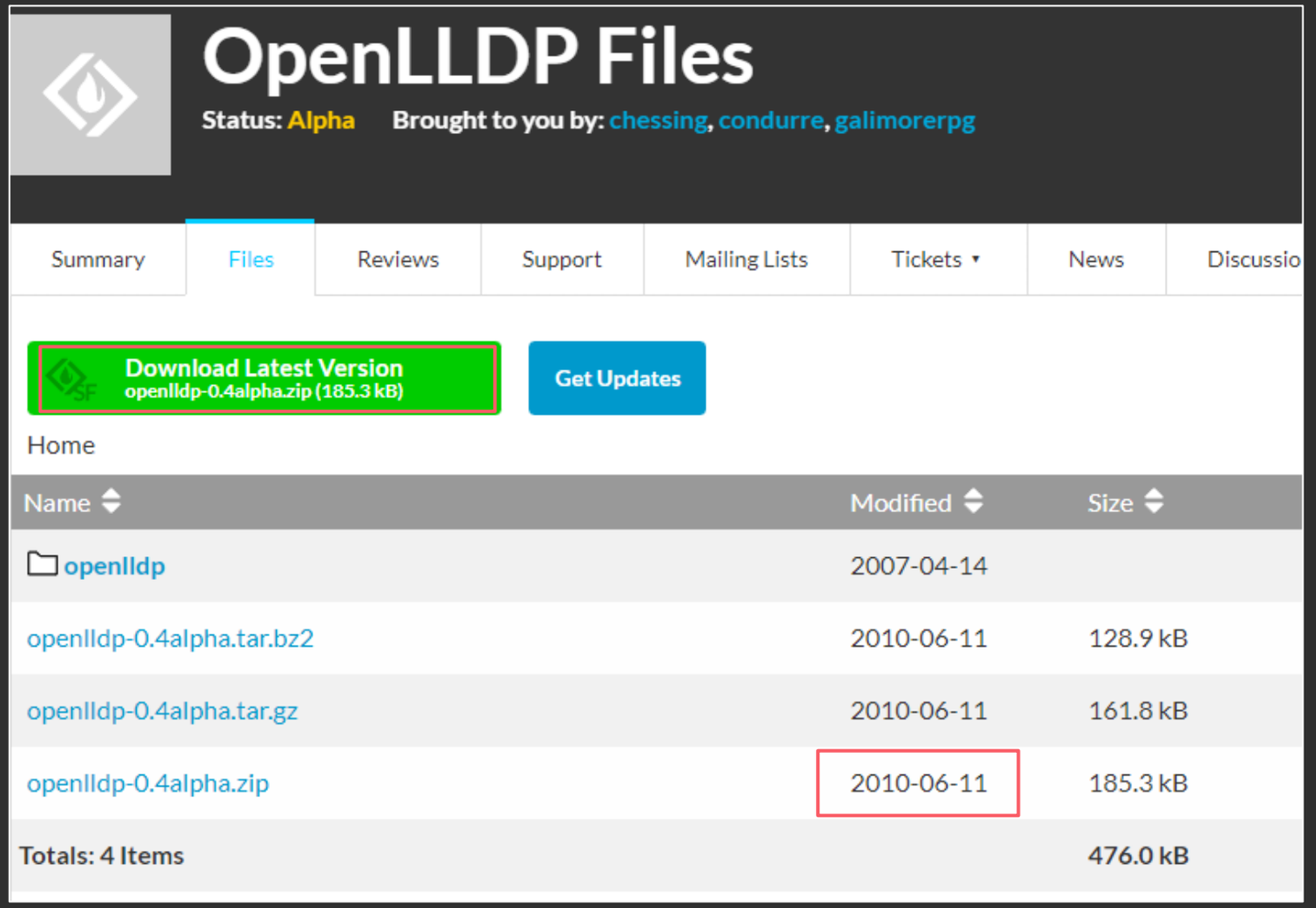
# OpenLLDP

“ *The OpenLLDP project aims to provide a comprehensive implementation of the IEEE standard 802.1AB Link Layer Discovery Protocol. The Open Source implementation of LLDP provided by the OpenLLDP project is intended to help foster wider adoption of LLDP.* ”




1. https://openlldp.sourceforge.net/

# OpenLLDP




- Null Pointer Dereference
- Integer Underflow
- Memory Leak
- Out-of-Bounds Read
- Heap Overflow

# OpenLLDP

```
int rxProcessFrame(struct lldp_port *lldp_port) {
  /* ... */
  do {
    /* ... */
    /* Validate as per 802.1AB section 10.3.2*/
    if (num_tlvs <= 3) {
      if (num_tlvs != tlv_type) {   // check tlv order
        badFrame++;
      }
    }
    /* ... */
    /* Validate the TLV */
    if (validate_tlv[tlv_type] != NULL) {
      if (validate_tlv[tlv_type](tlv) != XVALIDTLV) {
        badFrame++;
      }
    }
    else {
      if (validate_generic_tlv(tlv) != XVALIDTLV) {
        badFrame++;
      }
    }
    /* ... */
```

Root cause: the process will continue even if those checks fail

# l2g_lldp.ko


- Integer Underflow
- Buffer Overflow
- Out-of-Bounds Read

# l2g_lldp.ko

# #6 CVE-2021-34780

Buffer Overflow

## Cisco Small Business 220 Series Smart Switches Link Layer Discovery Protocol Vulnerabilities

High

| | | |
|---|---|---|
| **Advisory ID:** | cisco-sa-sb220-lldp-multivuls-mVRUtQ8T | CVE-2021-34775 |
| **First Published:** | 2021 October 6 16:00 GMT | CVE-2021-34776 |
| **Last Updated:** | 2022 January 13 21:37 GMT | CVE-2021-34777 |
| **Version 1.2:** | Final | More... |
| **Workarounds:** | No workarounds available | CWE-120 |
| **Cisco Bug IDs:** | CSCvz29108 | CWE-125 |
| | CSCvz29116 | |
| | CSCvz29120 | |
| | More... | |
| **CVSS Score:** | Base 8.8 | |

### Summary

Multiple vulnerabilities exist in the Link Layer Discovery Protocol (LLDP) implementation for Cisco Small Business 220 Series Smart Switches. An unauthenticated, adjacent attacker could perform the following:

- Execute code on the affected device or cause it to reload unexpectedly
- Cause LLDP database corruption on the affected device

For more information about th[illegible] section of this advisory.

Target release: 1.2.0.6

TIA TR-41 Committee TLV

```
Link Layer Discovery Protocol
  > Chassis Subtype = MAC address, Id: 00:18:[illegible]
  > Port Subtype = Locally assigned, Id: GE2
  > Time To Live = 120 sec
  v Telecommunications Industry Association TR-41 Committee - Location Identification
      1111 111. .... .... = TLV Type: Organization Specific (127)
      .... ...0 0000 1000 = TLV Length: 8
      Organization Unique Code: 00:12:bb (Telecommunications In
      Media Subtype: Location Identification (0x03)
      Location Data Format: ECS ELIN (3)
      ELIN: 111
  > End of LLDPDU
```

```
0000  01 80 c2 00 00 0e 00 0c  29 f8 05 47 88 cc 02 07   ······ ··)··G····
0010  04 00 18 ba 98 68 8f 04  04 07 47 45 32 06 02 00   ·····h·· ··GE2···
0020  78 fe 08 00 12 bb 03 03  31 31 31 00 00 00 00 00   x····· 111·····
0030  00 00 00 00 00 00 00 00  00 00 00 00               ···· ····
```

# l2g_lldp.ko

# #6 CVE-2021-34780

```
int lldp_pkt_rx(int a1) {
  /* ... */
  if ( a1 ) {
    if ( dword_B89D18 == 1 ) {
      memset(v138, 0, 0xC34);  //(1)
      /* ... */
      if ( v15 ) {
        while ( 1 ) {
          /* ... get tlv_type and tlv_length, call check_tlv_len() ... */
          if ( tlv_type >= 7 ) {
            /* ... */
            // case: Organizational Specific TLV (type=0x7f)
            if ( org_unique_code == 0x12BB ) {  //(2)
              switch ( v54[3] ) {  //(3) org_subtype
                /* ... */
                case 3u:
                  v72 = *v70;   //(4) loc_data_format
                  switch ( v72 ) {
                    /* ... */
                    case 3:
                      v73 = &v138[0x2EE];       //(5) (char *)&v138[0xBB8]
                      v74 = (unsigned __int16)(HIWORD(v139[2]) - 5);  //(6) tlv_length - 5
                      break;
                    /* ... */
                  }
                  memcpy(v73, v70 + 1, v74);    //(7) buffer overflow
```

check_tlv_len() is called to ensure that the range of tlv_length is [0x4, 0x1ff] for tlv_type 0x7f

0xC34 – 0xBB8 = 0x7C

# switchdrvr

UBIQUITI NETWORKS

US-16-XG | USW-Pro-24-POE | ES-16-XG | US-L2-24-POE | ...

- Buffer Overflow
- Out-of-Bounds Read

NETGEAR®

GS418TPP | GS510TPP | GS728TX | GS752TX | ...

```
Link Layer Discovery Protocol
  > Chassis Subtype = MAC address, Id: 00:18:ba:98:68:8f
  > Port Subtype = Locally assigned, Id: GE2
  > Time To Live = 120 sec
  v Management Address
      0001 000. .... .... = TLV Type: Management Address (8)
      .... ...0 0000 1100 = TLV Length: 12
      Address String Length: 5
      Address Subtype: IPv4 (1)
      Management Address: 127.0.0.1
      Interface Subtype: ifIndex (2)
      Interface Number: 1
      OID String Length: 0
  > End of LLDPDU
```

Management Address TLV

# switchdrvr

# #7 Buffer Overflow

Ubiquiti Switch US-16-150W: 4.3.22

```
int lldpPduReceiveProcess(int a1, int a2)
{
  /* ... */
  while ( !parse_tlv_type_length(&tlv_type, &tlv_length, &v86, v65) && tlv_type )
  {
    /* ... */
    if ( tlv_type > 6 )
    {
      if ( tlv_type == 8 )  // Management Address TLV
      {
        /* ... */
        v17 = sub_261444((int)v20, &v86, tlv_length);  //(1)
        /* ... */
```

# switchdrvr

# #7 Buffer Overflow

```
int sub_261444(int a1, int *a2, unsigned int a3)
{
  /* ... */
  int dest[7]; // [sp+4h] [bp-1Ch] BYREF

  *(_BYTE *)(a1 + 1) = *(_BYTE *)(*a2 + a2[1]) - 1;   //(2) addr_str_len - 1
  v5 = a2[1] + 1;
  a2[1] = v5;
  if ( (unsigned int)*(unsigned __int8 *)(a1 + 1) - 1 > 0x1E )  //(3)
    return 1;
  *(_BYTE *)a1 = *(_BYTE *)(*a2 + v5);
  v6 = a2[1] + 1;
  a2[1] = v6;
  v7 = *a2;
  if ( *(_BYTE *)a1 == 1 )  // addr_subtype
  {
    memcpy(dest, (const void*)(v7 + v6), *(unsigned __int8*)(a1 + 1)); //(4) buffer overflow
    /* ... */
```

*(_BYTE *)(a1 + 1): [1, 0x1F]

# What We Learnt

- Vulnerability patterns
  - Improper TLV order validation
  - Improper duplicated TLV handling
  - Lack of bounds checks
  - Integer wraparounds
  - Discrepancy between check and use
- Better understanding of the LLDP protocol format

# IEEE 802.1AB Standard

## Link Layer Discovery Protocol (LLDP) Specification

# IEEE 802.1AB Standard


- 802.1AB-2005
- 802.1AB-2009
- 802.1AB-2016
- 802.1ABdh-2021

1. https://ieeexplore.ieee.org/document/7433915

# Basic Management TLV Formats/Definitions

Three mandatory TLVs shall be included at the beginning of each LLDPDU and shall be in the order shown: Chassis ID TLV, Port ID TLV, Time To Live TLV

#D1 TLV order

Each LLDPDU shall contain one, and only one basic management TLV (type from 1 to 7), except Management Address TLV

#D2 TLV repetition

The TLV information string length is equal to: (management address string length) + (OID string length) + 7

#D3 Management Address TLV information string length

As a consequence of the limitation on the size of the management address field (a maximum of 31 octets)

#D4 Management Address TLV management address length

# Organizationally Specific TLV Formats/Definitions

The TLV information string length field shall contain the length, in octets, of the (VLAN name + 7)

#D5 VLAN Name TLV information string length

This field shall contain the exact length, in octets, of the Location ID data field + 5

#D6 Location Identification TLV Location ID string length

| Location data format type | Data type provided | Location ID data length (octets) |
|---|---|---|
| 1 | Coordinate-based LCI | 16 |
| 2 | Civic Address LCI | 6 to 256 |
| 3 | ECS ELIN | 10 to 25 |

#D7 Location Identification TLV Location data format type values

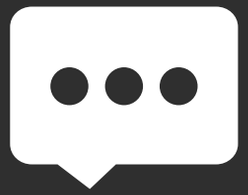
# How LLDP implementations violate the standard

| | #D1 | #D2 | #D3 | #D4 | #D5 | #D6 | #D7 |
|---|---|---|---|---|---|---|---|
| lldpd | | ✗ | | ✗ | | | ! |
| OpenLLDP | ✗ | ✗ | | | | ✗ | ✗ |
| openvswitch/lldp | | ✗ | | ✗ | | | |
| ocelot-vsc6825/lldp | | ! | ✗ | ! | | | |
| l2g_lldp.ko | ! | ! | ✗ | ! | ✗ | ✗ | ✗ |
| switchdrvr | | ! | ✗ | ✗ | | | |
| ISS.exe | | ! | ✗ | | ✗ | | |

✗ violate and lead to bugs  ! violate but no effects

# Agenda

Introduction



CDP & LLDP Protocol

LCDPwn Research

Summary

# What Can We Learn

- Variant analysis is awesome

- Learn from CDPwn: look into the same implementation
- From CDP to LLDP: look into similar protocols/components, or the same implementation on different devices/systems

# What Can We Learn

- Variant analysis is awesome
- Software Bills of Materials (SBOMs) are important in software security and software supply chain

CDP

NX-OS Switch | IP Phone | Communications Gateway | Access Point

IOS XR Router | IP Camera | Wireless Controller | Collaboration Endpoint

different implementations

LLDP

NX-OS Switch | IP Phone | Small Business Router / IP Camera

Communications Gateway | LAN Switch - Small Business

different implementations

# What Can We Learn

- Variant analysis is awesome
- Software Bills of Materials (SBOMs) are important in software security and software supply chain
- Keep components up-to-date if possible, and avoid using outdated components

lldpd

CVE-2015-8011: buffer overflow when handli
management address TLV for LLDP. This bug has been introduced in version 0.6.0. It has been fixed in commit dd4f16e7 and in version 0.7.19.

CVE-2015-8012: crash on malformed management address. This bug has been introduced in version 0.6.0. It has been fixed in commit 793526f8 and in

The n-day vulnerabilities from 6 years ago still survive in 2021

OpenLLDP

Download Latest Version
openlldp-0.4alpha.zip (185.3 kB)

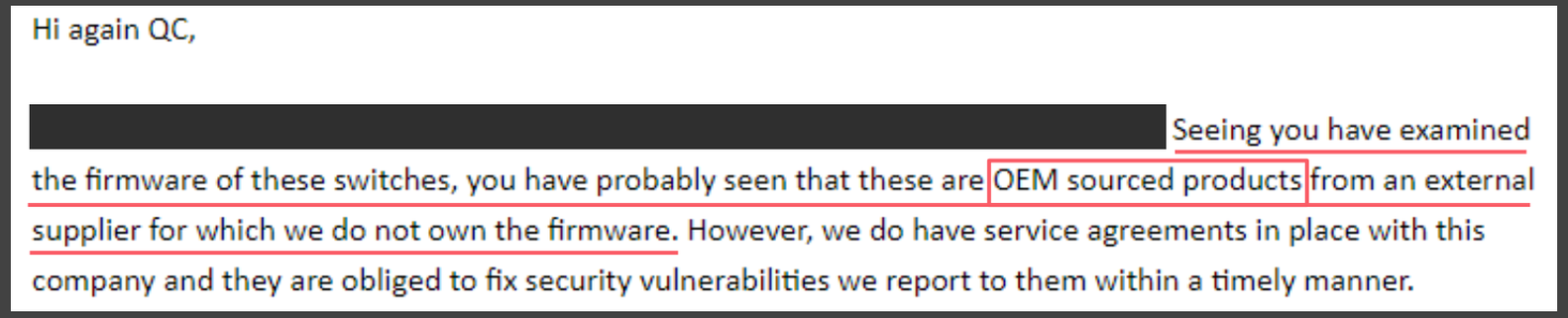
Home

| Name | Modified | Size |
|---|---|---|
| openlldp | 2007-04-14 | |
| openlldp-0.4alpha.tar.bz2 | 2010-06-11 | 128.9 kB |
| openlldp-0.4alpha.tar.gz | 2010-06-11 | 161.8 kB |
| openlldp-0.4alpha.zip | 2010-06-11 | 185.3 kB |

A 10 years outdated project

# What Can We Learn

- Variant analysis is awesome
- Software Bills of Materials (SBOMs) are important in software security and software supply chain
- Keep components up-to-date if possible, and avoid using outdated components
- Pay attention to the OEM software



Hi again QC,

[redacted] Seeing you have examined the firmware of these switches, you have probably seen that these are OEM sourced products from an external supplier for which we do not own the firmware. However, we do have service agreements in place with this company and they are obliged to fix security vulnerabilities we report to them within a timely manner.

# Thanks!



cq674350529


cq674350529@gmail.com