Reverse Engineering Archeology: Multiple Devices, Multiple Versions

CONFidence 2020- September 8th, 2020

JSOF

Who are we?

JSOF is a software security consultancy

- **Shlomi Oberman**, co-founder, JSOF
- **Moshe Kol**, Security researcher, JSOF; Finder of Ripple20
- **Ariel Schön**, Security researcher, JSOF

Agenda

- Ripple20
- Reverse engineering process:
  - Multiple binaries
- Wrap-up

Ripple20

- Series of 19 zero-day vulnerabilities in **Treck TCP/IP***
- Amplified by the supply chain
- 100’s of millions of devices
- Medical, ICS, Home, Enterprise, Transportation, Utilities

Ripple20

CVE-2020-11896	CVE-2020-11901	CVE-2020-11906	CVE-2020-11911
CVE-2020-11897	CVE-2020-11902	CVE-2020-11907	CVE-2020-11912
CVE-2020-11898	CVE-2020-11903	CVE-2020-11908	CVE-2020-11913
CVE-2020-11899	CVE-2020-11904	CVE-2020-11909	CVE-2020-11914
CVE-2020-11900	CVE-2020-11905	CVE-2020-11910	

- 4 critical remote code execution vulnerabilities

JSOF

100’s of Millions of Devices Affected

cisco intel hp DELL Rockwell Automation

Schneider Electric xerox DIGI CAT aruba a Hewlett Packard Enterprise company

And many more...

Ripple20 Research

- Reverse engineering 7 different devices with multiple versions
- Every device has a different configuration
- Ongoing research Sep’19 - Jun’20 (9 months)
- Some strange architectures and firmwares involved

2 whitepapers released (CVE-2020-11896/CVE-2020-11901)

Challenge

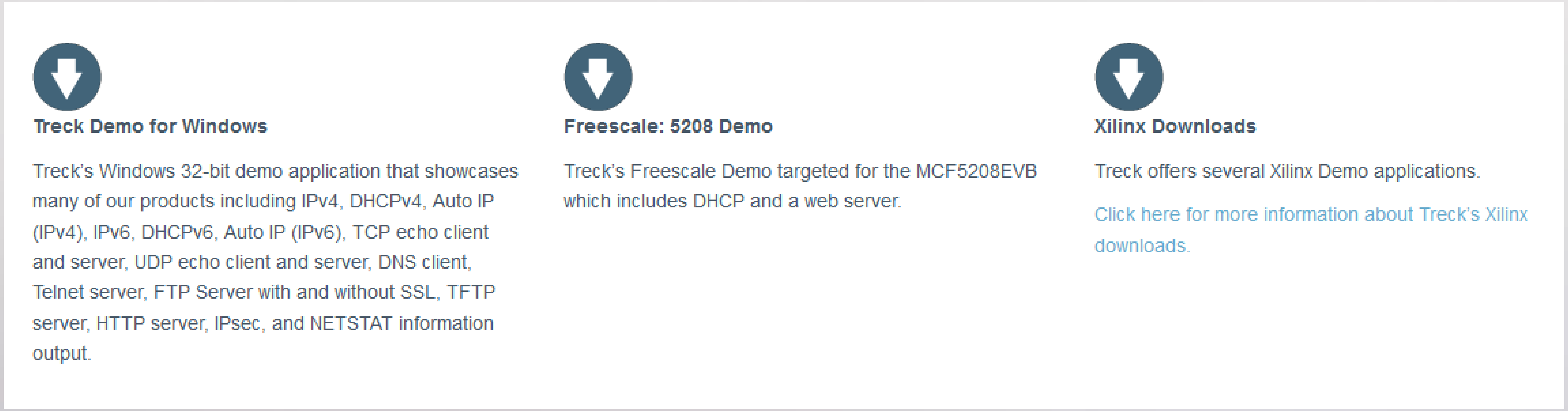
- 1 library many versions
  - Little did we know…
- Need symbols, debug, binary…
- Multiple data points
- Lots of history

Challenge

- Multiple firmwares/binaries
- Security/Archeology project
- Library dating to pre-2000

How did we start?

- Browsing to Treck's website
- Looking for datasheets, manuals, demos



Treck Demo for Windows

Treck's Windows 32-bit demo application that showcases many of our products including IPv4, DHCPv4, Auto IP (IPv4), IPv6, DHCPv6, Auto IP (IPv6), TCP echo client and server, UDP echo client and server, DNS client, Telnet server, FTP Server with and without SSL, TFTP server, HTTP server, IPsec, and NETSTAT information output.

Freescale: 5208 Demo

Treck's Freescale Demo targeted for the MCF5208EVB which includes DHCP and a web server.

Xilinx Downloads

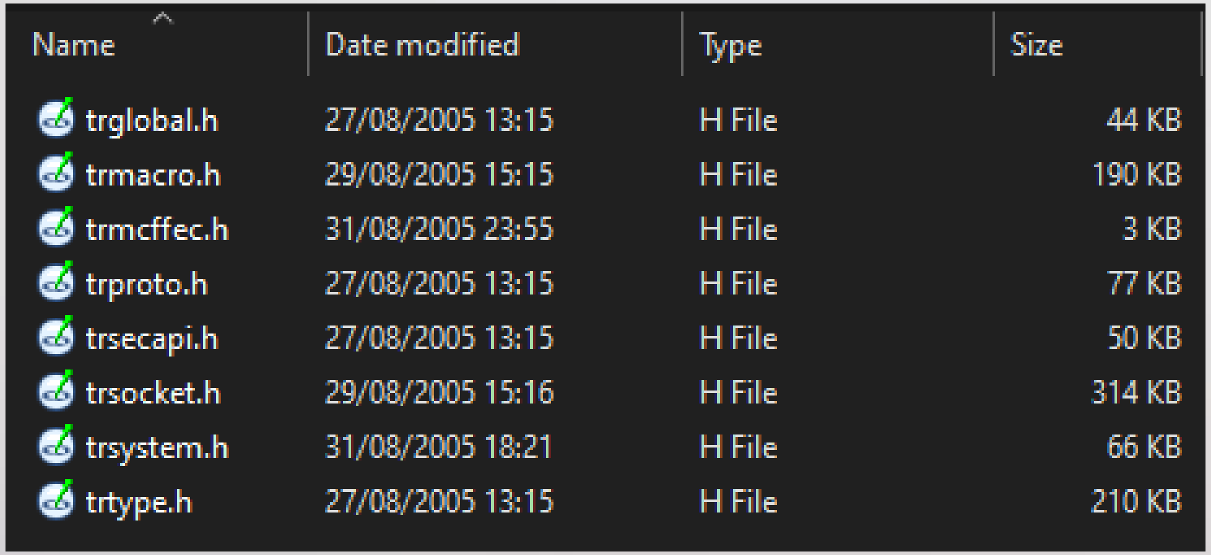
Treck offers several Xilinx Demo applications.

Click here for more information about Treck's Xilinx downloads.

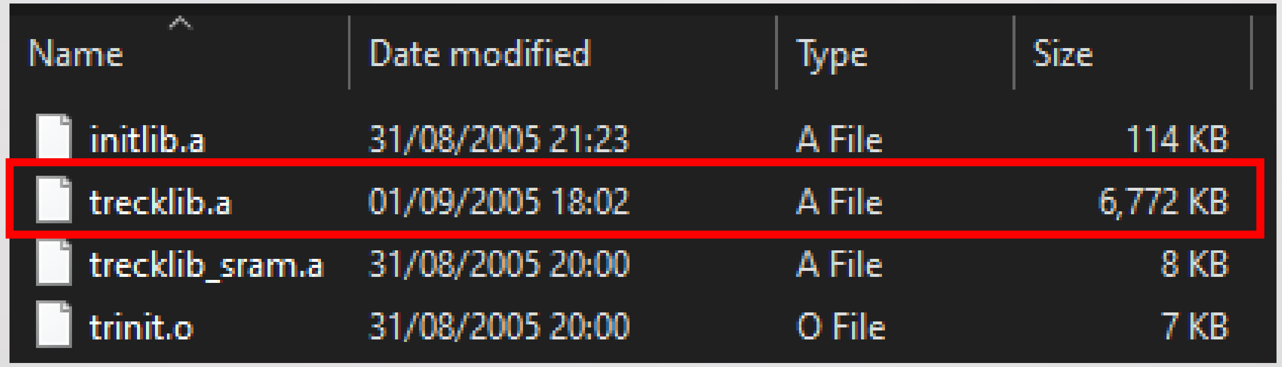
Binary #1 – Freescale demo

Freescale 5280 demo

- Contains headers and static library



Name	Date modified	Type	Size
trglobal.h	27/08/2005 13:15	H File	44 KB
trmacro.h	29/08/2005 15:15	H File	190 KB
trmcffec.h	31/08/2005 23:55	H File	3 KB
trproto.h	27/08/2005 13:15	H File	77 KB
trsecapi.h	27/08/2005 13:15	H File	50 KB
trsocket.h	29/08/2005 15:16	H File	314 KB
trsystem.h	31/08/2005 18:21	H File	66 KB
trtype.h	27/08/2005 13:15	H File	210 KB



Name	Date modified	Type	Size
initlib.a	31/08/2005 21:23	A File	114 KB
trecklib.a	01/09/2005 18:02	A File	6,772 KB
trecklib_sram.a	31/08/2005 20:00	A File	8 KB
trinit.o	31/08/2005 20:00	O File	7 KB

- The headers provide useful comments and structure definitions.

Freescale 5280 demo

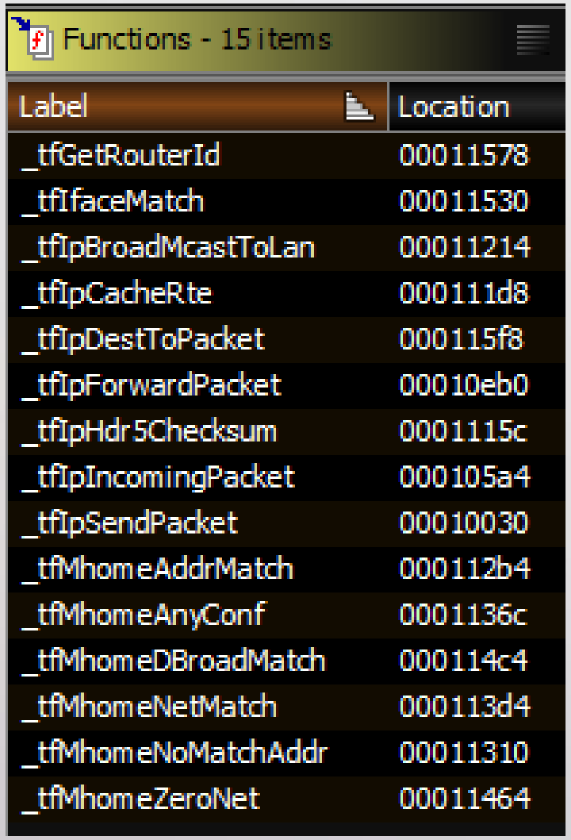
- Static library contains 202 object files:

```
tr8023.o      trdhcp.o      trhttp.o      triptunl.o    trnetid.o
trarp.o       trdialer.o    trhttpd.o     trlist.o      trntstat.o
trarpchk.o    trdsplib.o    tricmp.o      trlock.o      trping.o
trautoip.o    treap.o       trigmp.o      trlog.o       trpop3.o
trbase64.o    trethcom.o    trindrmc.o    trloop.o      trppp.o
trbootp.o     trether.o     trindrv.o     trlqm.o       trramfs.o      …
trbtdhcp.o    trethtag.o    trinscdr.o    trmime.o      trrelay.o
trbuffer.o    trfs.o        trip.o        trmoblip.o    trresolv.o
trcmplib.o    trftp.o       tripfrag.o    trmschap.o    trrip.o
trdevice.o    trftpd.o      triphc.o      trnat.o       trromfs.o
```

- Architecture: Motorola 68030 big-endian

Freescale 5280 demo

- Object files have function names:



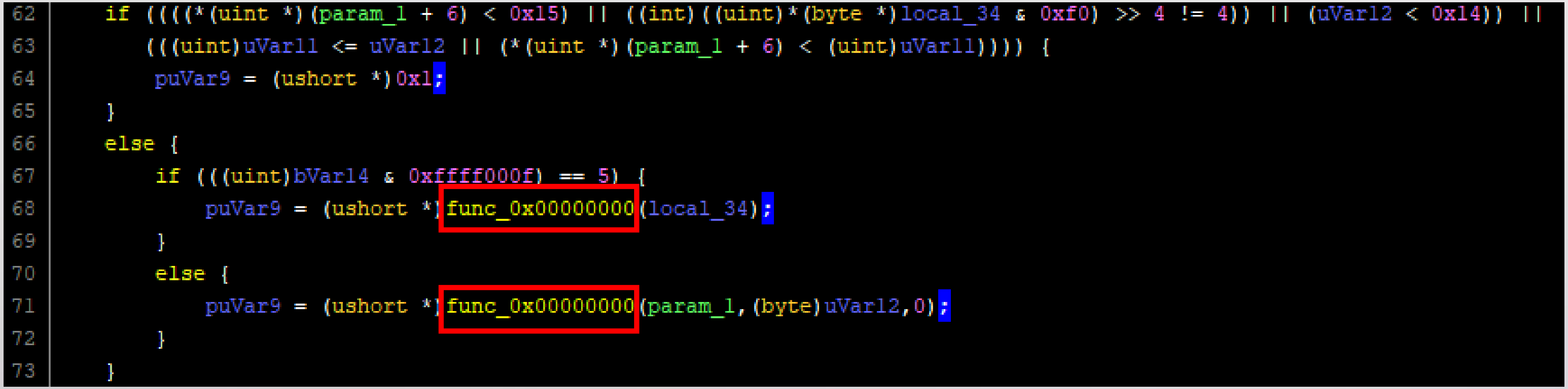
Functions - 15 items

Label	Location
_tfGetRouterId	00011578
_tfIfaceMatch	00011530
_tfIpBroadMcastToLan	00011214
_tfIpCacheRte	000111d8
_tfIpDestToPacket	000115f8
_tfIpForwardPacket	00010eb0
_tfIpHdr5Checksum	0001115c
_tfIpIncomingPacket	000105a4
_tfIpSendPacket	00010030
_tfMhomeAddrMatch	000112b4
_tfMhomeAnyConf	0001136c
_tfMhomeDBroadMatch	000114c4
_tfMhomeNetMatch	000113d4
_tfMhomeNoMatchAddr	00011310
_tfMhomeZeroNet	00011464

trip.o

Freescale 5280 demo

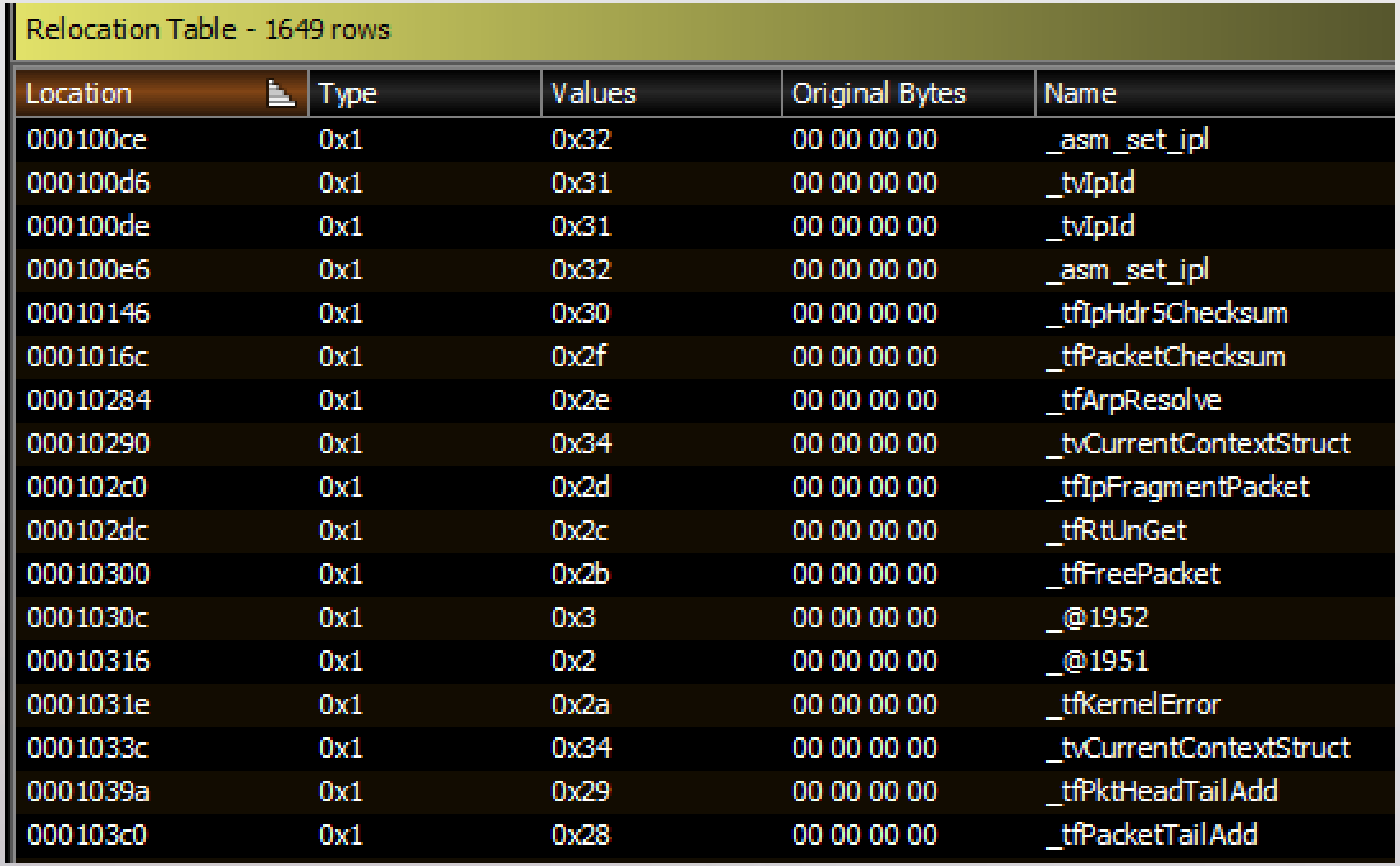
- But non-local function calls are missing:



```
if ((((*(uint *)(param_1 + 6) < 0x15) || ((int)((uint)*(byte *)local_34 & 0xf0) >> 4 != 4)) || (uVar12 < 0x14)) ||
   (((uint)uVar11 <= uVar12 || (*(uint *)(param_1 + 6) < (uint)uVar11)))) {
    puVar9 = (ushort *)0x1;
}
else {
    if (((uint)bVar14 & 0xffff000f) == 5) {
        puVar9 = (ushort *)func_0x00000000(local_34);
    }
    else {
        puVar9 = (ushort *)func_0x00000000(param_1,(byte)uVar12,0);
    }
}
```

Freescale 5280 demo

- Some can be recovered using the relocation table:



Relocation Table - 1649 rows

Location	Type	Values	Original Bytes	Name
000100ce	0x1	0x32	00 00 00 00	_asm_set_ipl
000100d6	0x1	0x31	00 00 00 00	_tvIpId
000100de	0x1	0x31	00 00 00 00	_tvIpId
000100e6	0x1	0x32	00 00 00 00	_asm_set_ipl
00010146	0x1	0x30	00 00 00 00	_tfIpHdr5Checksum
0001016c	0x1	0x2f	00 00 00 00	_tfPacketChecksum
00010284	0x1	0x2e	00 00 00 00	_tfArpResolve
00010290	0x1	0x34	00 00 00 00	_tvCurrentContextStruct
000102c0	0x1	0x2d	00 00 00 00	_tfIpFragmentPacket
000102dc	0x1	0x2c	00 00 00 00	_tfRtUnGet
00010300	0x1	0x2b	00 00 00 00	_tfFreePacket
0001030c	0x1	0x3	00 00 00 00	_@1952
00010316	0x1	0x2	00 00 00 00	_@1951
0001031e	0x1	0x2a	00 00 00 00	_tfKernelError
0001033c	0x1	0x34	00 00 00 00	_tvCurrentContextStruct
0001039a	0x1	0x29	00 00 00 00	_tfPktHeadTailAdd
000103c0	0x1	0x28	00 00 00 00	_tfPacketTailAdd

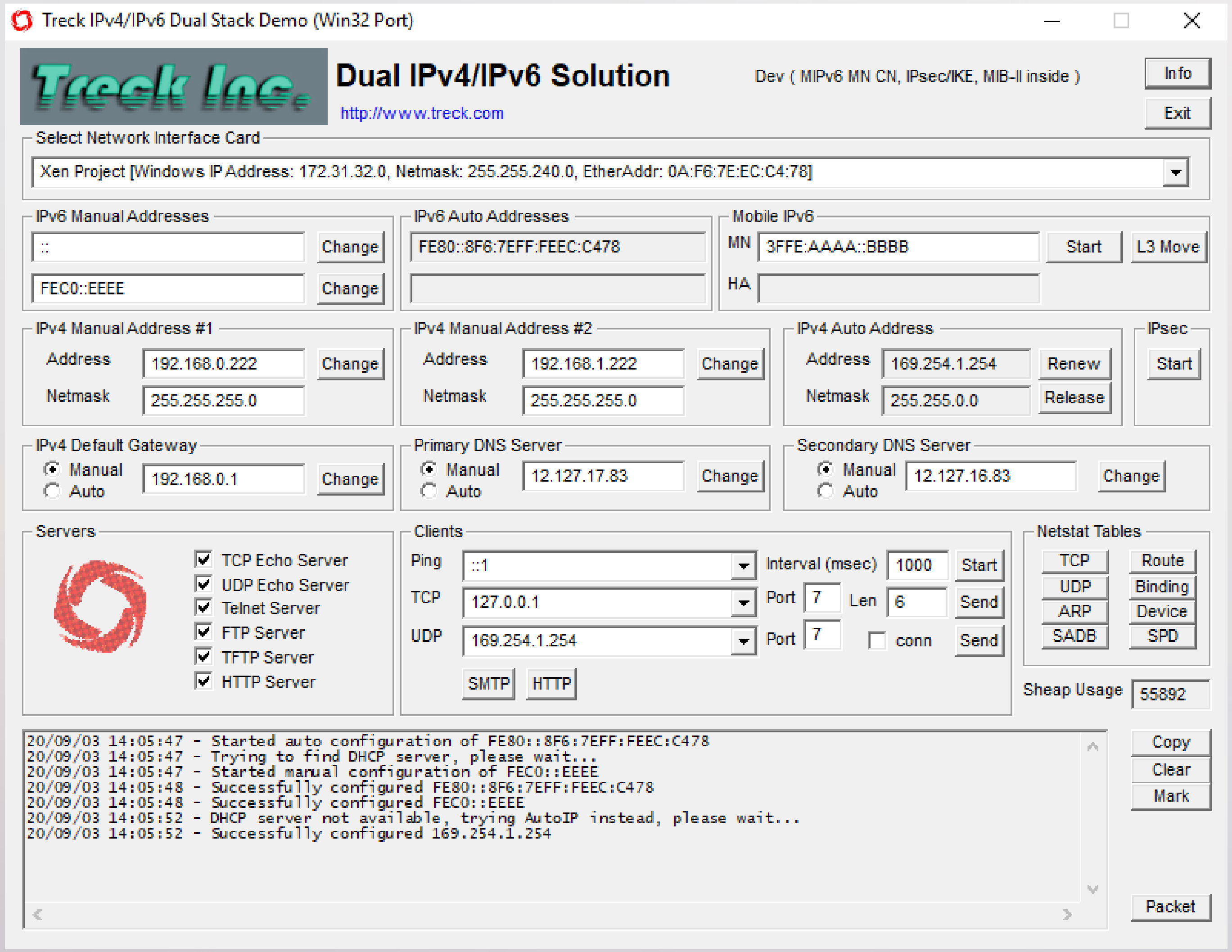
Freescale 5280 demo

- In summary:
  - Useful data point
  - Cannot be debugged easily

Binary #2 – Win32 demo

Win32 Demo

- Treck (used to) offer Windows 32-bit demo app



Treck IPv4/IPv6 Dual Stack Demo (Win32 Port)

Treck Inc. Dual IPv4/IPv6 Solution

Dev (MIPv6 MN CN, IPsec/IKE, MIB-II inside)

http://www.treck.com

Select Network Interface Card: Xen Project [Windows IP Address: 172.31.32.0, Netmask: 255.255.240.0, EtherAddr: 0A:F6:7E:EC:C4:78]

IPv6 Manual Addresses: :: / FEC0::EEEE

IPv6 Auto Addresses: FE80::8F6:7EFF:FEEC:C478

Mobile IPv6: MN 3FFE:AAAA::BBBB; HA

IPv4 Manual Address #1: Address 192.168.0.222; Netmask 255.255.255.0

IPv4 Manual Address #2: Address 192.168.1.222; Netmask 255.255.255.0

IPv4 Auto Address: Address 169.254.1.254; Netmask 255.255.0.0

IPv4 Default Gateway: [✓ Manual] Auto; 192.168.0.1

Primary DNS Server: [✓ Manual] Auto; 12.127.17.83

Secondary DNS Server: [✓ Manual] Auto; 12.127.16.83

Servers: ✓ TCP Echo Server; ✓ UDP Echo Server; ✓ Telnet Server; ✓ FTP Server; ✓ TFTP Server; ✓ HTTP Server

Clients: Ping ::1, Interval (msec) 1000; TCP 127.0.0.1, Port 7, Len 6; UDP 169.254.1.254, Port 7, conn

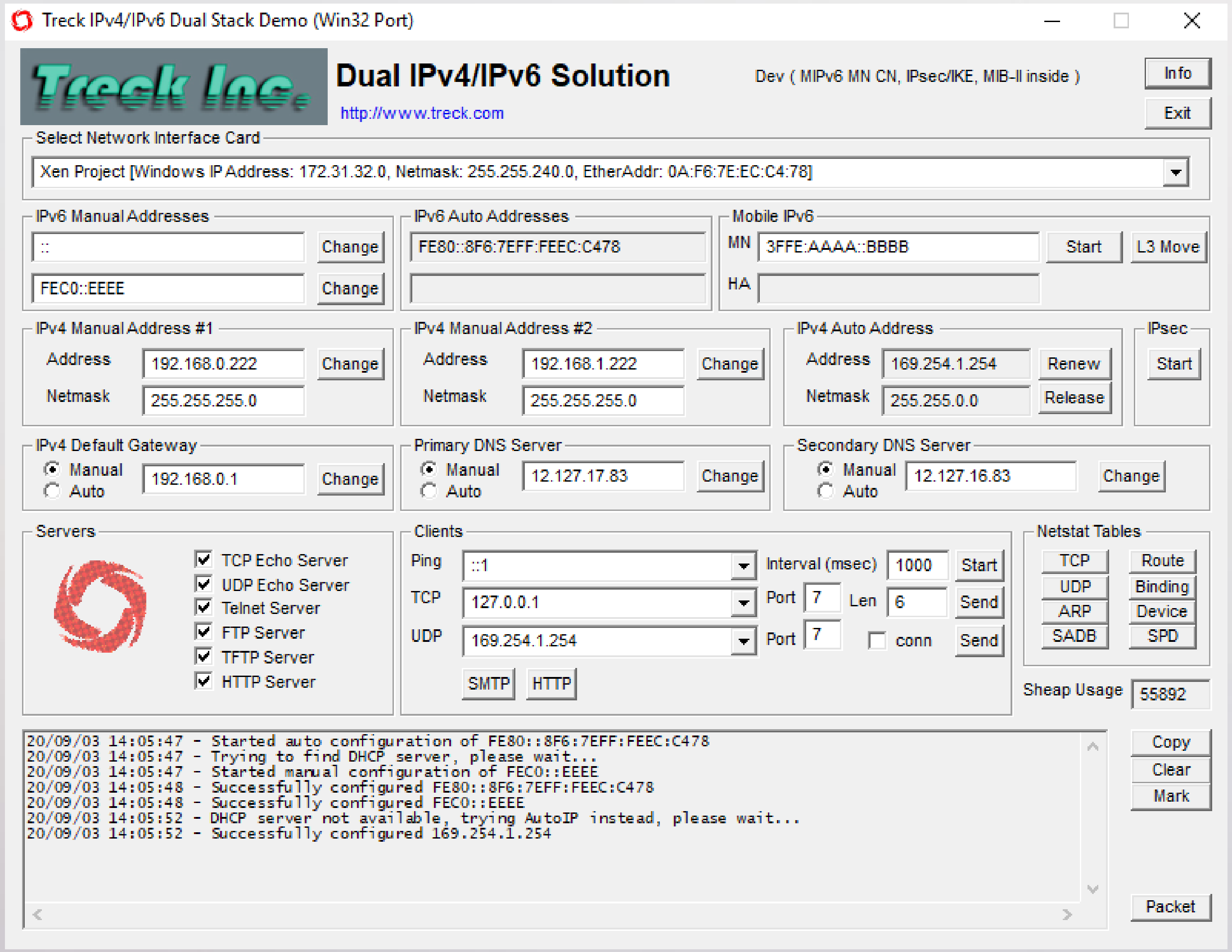
Sheap Usage 55892

```
20/09/03 14:05:47 - Started auto configuration of FE80::8F6:7EFF:FEEC:C478
20/09/03 14:05:47 - Trying to find DHCP server, please wait...
20/09/03 14:05:47 - Started manual configuration of FEC0::EEEE
20/09/03 14:05:48 - Successfully configured FE80::8F6:7EFF:FEEC:C478
20/09/03 14:05:48 - Successfully configured FEC0::EEEE
20/09/03 14:05:52 - DHCP server not available, trying AutoIP instead, please wait...
20/09/03 14:05:52 - Successfully configured 169.254.1.254
```

Win32 Demo

JSOF

- Supports many useful features:
  - IPv4
  - IPv6
  - DHCP client
  - TCP
  - UDP
  - ICMP
  - IPSEC
  - Mobile IPv6



Treck IPv4/IPv6 Dual Stack Demo (Win32 Port)

Treck Inc. **Dual IPv4/IPv6 Solution** Dev (MIPv6 MN CN, IPsec/IKE, MIB-II inside) Info Exit

http://www.treck.com

Select Network Interface Card: Xen Project [Windows IP Address: 172.31.32.0, Netmask: 255.255.240.0, EtherAddr: 0A:F6:7E:EC:C4:78]

IPv6 Manual Addresses	:: ; FEC0::EEEE
IPv6 Auto Addresses	FE80::8F6:7EFF:FEEC:C478
Mobile IPv6	MN 3FFE:AAAA::BBBB ; HA
IPv4 Manual Address #1	Address 192.168.0.222 ; Netmask 255.255.255.0
IPv4 Manual Address #2	Address 192.168.1.222 ; Netmask 255.255.255.0
IPv4 Auto Address	Address 169.254.1.254 ; Netmask 255.255.0.0
IPv4 Default Gateway	[✓ Manual] Auto ; 192.168.0.1
Primary DNS Server	[✓ Manual] Auto ; 12.127.17.83
Secondary DNS Server	[✓ Manual] Auto ; 12.127.16.83

Servers: ✓ TCP Echo Server, ✓ UDP Echo Server, ✓ Telnet Server, ✓ FTP Server, ✓ TFTP Server, ✓ HTTP Server

Clients: Ping ::1 ; Interval (msec) 1000 ; TCP 127.0.0.1 ; Port 7 ; Len 6 ; UDP 169.254.1.254 ; Port 7 ; conn

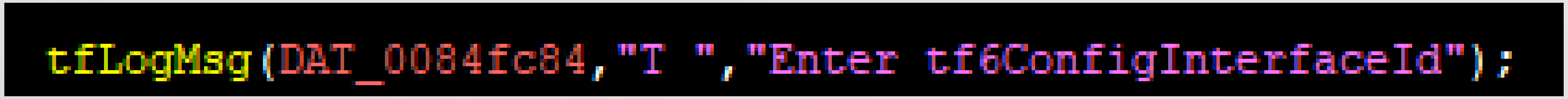
Sheap Usage 55892

```
20/09/03 14:05:47 - Started auto configuration of FE80::8F6:7EFF:FEEC:C478
20/09/03 14:05:47 - Trying to find DHCP server, please wait...
20/09/03 14:05:47 - Started manual configuration of FEC0::EEEE
20/09/03 14:05:48 - Successfully configured FE80::8F6:7EFF:FEEC:C478
20/09/03 14:05:48 - Successfully configured FEC0::EEEE
20/09/03 14:05:52 - DHCP server not available, trying AutoIP instead, please wait...
20/09/03 14:05:52 - Successfully configured 169.254.1.254
```

Win32 Demo: Finding Treck

- No debug symbols.

- Able to recover some function names using debug strings:

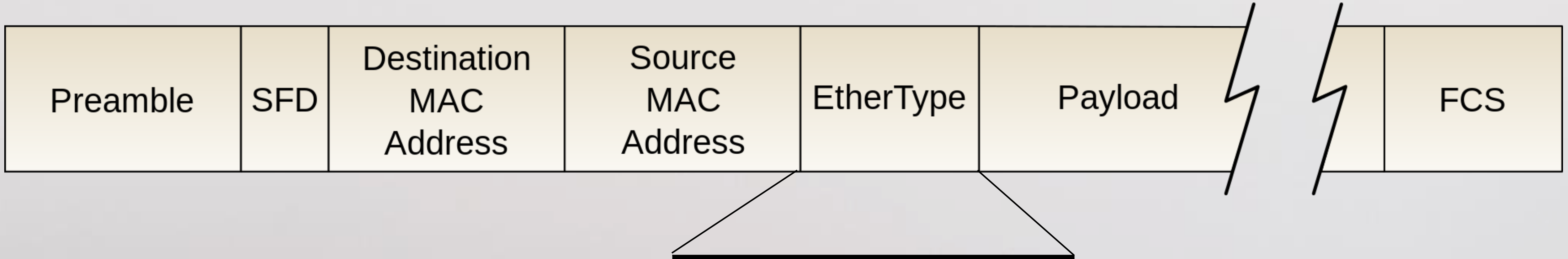


```
tfLogMsg(DAT_0084fc84,"T ","Enter tf6ConfigInterfaceId");
```

- Applies mostly to IPv6 functions ☹

Win32 Demo: Finding Treck

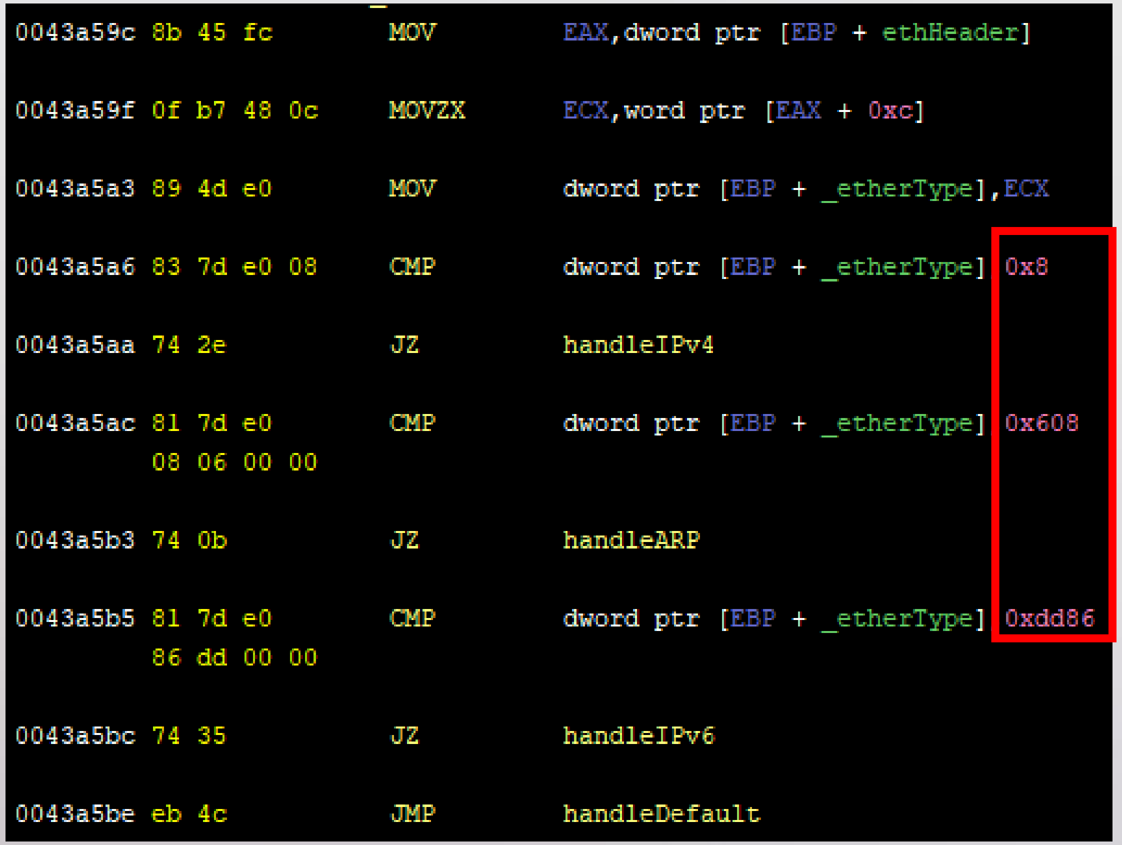
- To locate the IPv4 code base, we searched for EtherType constants in the binary.
- Recall Ethernet packet format:



EtherType	Protocol
0x0800	IPv4
0x0806	ARP
0x86dd	IPv6

Win32 Demo: Finding Treck

- Using this technique we were able to locate `tfEtherRecv`.



```
0043a59c 8b 45 fc        MOV        EAX,dword ptr [EBP + ethHeader]
0043a59f 0f b7 48 0c     MOVZX      ECX,word ptr [EAX + 0xc]
0043a5a3 89 4d e0        MOV        dword ptr [EBP + _etherType],ECX
0043a5a6 83 7d e0 08     CMP        dword ptr [EBP + _etherType] 0x8
0043a5aa 74 2e           JZ         handleIPv4
0043a5ac 81 7d e0        CMP        dword ptr [EBP + _etherType] 0x608
         08 06 00 00
0043a5b3 74 0b           JZ         handleARP
0043a5b5 81 7d e0        CMP        dword ptr [EBP + _etherType] 0xdd86
         86 dd 00 00
0043a5bc 74 35           JZ         handleIPv6
0043a5be eb 4c           JMP        handleDefault
```

x86 is little-endian architecture!

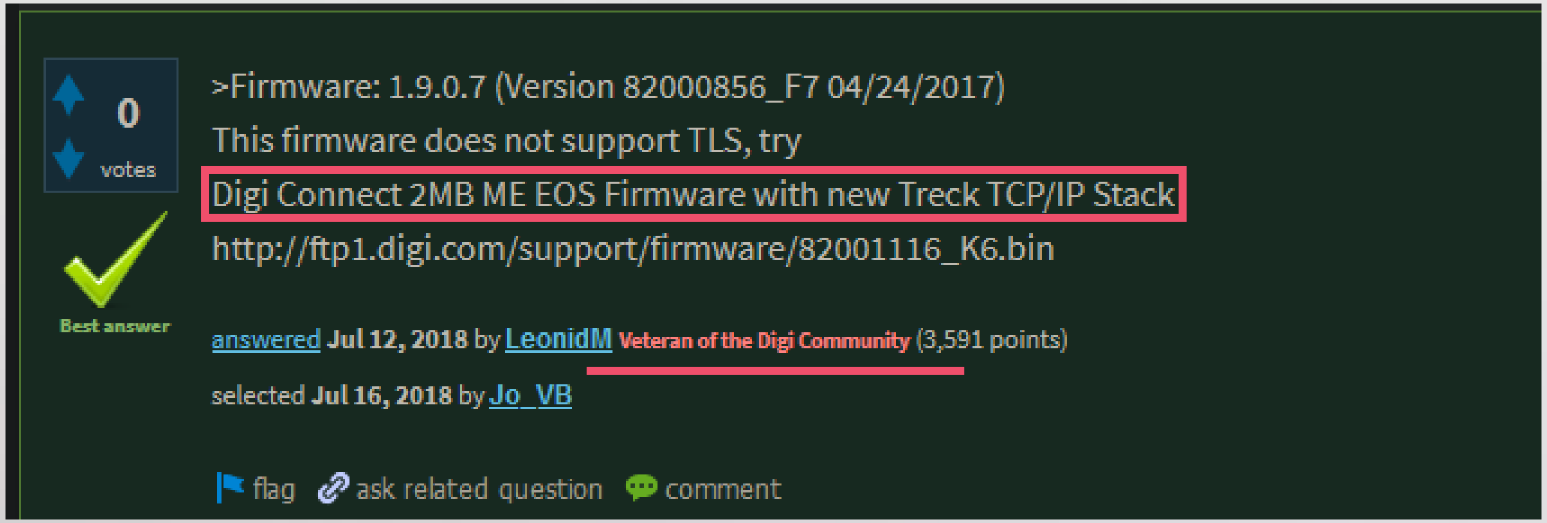
Win32 Demo: Results

- We reverse engineered large parts of the network stack
- We found some vulnerabilities
- We wanted to test if other devices are affected

Binary #3 - Digi dev board

Digi Connect ME 9210

- A "Veteran of the Digi Community" mentioned online that Digi Connect ships with Treck TCP/IP stack in Digi forum:



0 votes

Best answer

>Firmware: 1.9.0.7 (Version 82000856_F7 04/24/2017)
This firmware does not support TLS, try
Digi Connect 2MB ME EOS Firmware with new Treck TCP/IP Stack
http://ftp1.digi.com/support/firmware/82001116_K6.bin

answered Jul 12, 2018 by LeonidM Veteran of the Digi Community (3,591 points)

selected Jul 16, 2018 by Jo_VB

flag ask related question comment

Digi Connect ME 9210

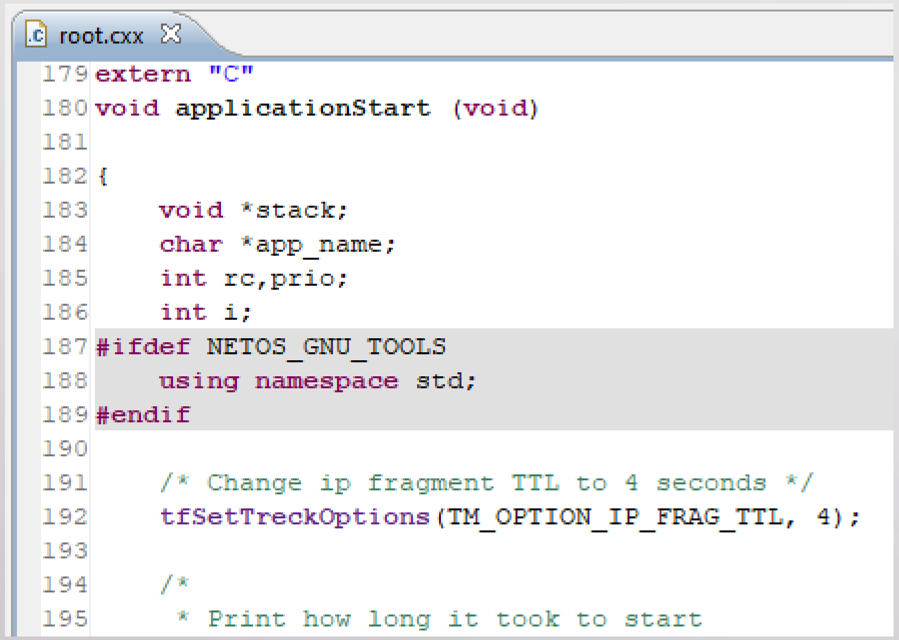
- Digi Connect ME devices come in two flavors:
  - Running embedded Linux
  - Running proprietary NET+OS

- The network stack of NET+OS 7.5 is Treck TCP/IP.

- We bought the Connect ME 9210 development kit.

Digi Connect ME 9210

- Runs Digi's new 32-bit NS9210 processor (ARM9).
- Have debugging capabilities using JTAG.
- Comes with eclipse-based IDE to write software:



```
extern "C"
void applicationStart (void)

{
    void *stack;
    char *app_name;
    int rc,prio;
    int i;
#ifdef NETOS_GNU_TOOLS
    using namespace std;
#endif

    /* Change ip fragment TTL to 4 seconds */
    tfSetTreckOptions(TM_OPTION_IP_FRAG_TTL, 4);

    /*
     * Print how long it took to start
```

Digi Connect ME 9210

- We compiled some basic example and examined the resulting ELF file.
- ELF comes with debug symbols!
- We developed an exploit for CVE-2020-11896 on this device.
- Disadvantage: relatively old Treck version (4.7).

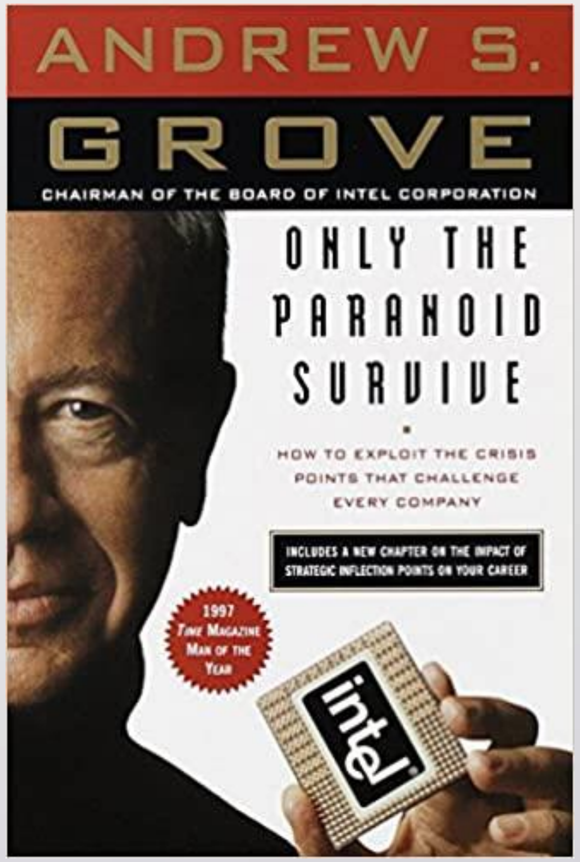
Binary #4 – Intel AMT

Intel ME

- In a quest for newer versions, we looked at Intel ME.
- Treck powers the AMT module.
- We speculated that since Intel is a security-aware company, they must have updated their Treck software.

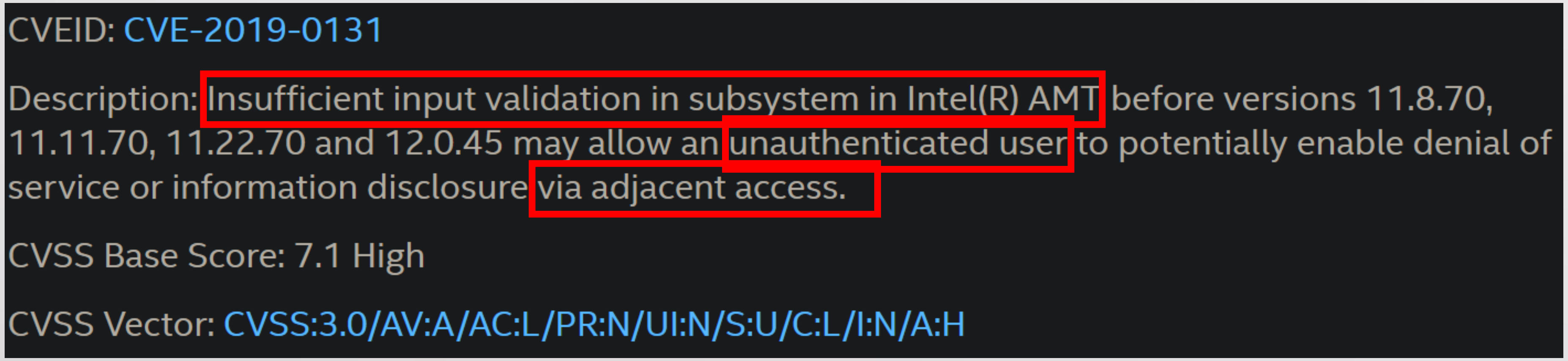
We thought we had 1-days

- Intel binary had some “defensive programming”
- We thought we had **1-days** that still existed in the wild (we were mostly wrong)
- Maybe fixes, maybe `ifdef`, maybe they are paranoid


ANDREW S.
GROVE
CHAIRMAN OF THE BOARD OF INTEL CORPORATION
ONLY THE PARANOID SURVIVE
HOW TO EXPLOIT THE CRISIS POINTS THAT CHALLENGE EVERY COMPANY
INCLUDES A NEW CHAPTER ON THE IMPACT OF STRATEGIC INFLECTION POINTS ON YOUR CAREER
1997 TIME MAGAZINE MAN OF THE YEAR
intel

Intel ME: Patch-diffing

- INTEL-SA-00241 describes a vulnerability that looks related:



CVEID: CVE-2019-0131

Description: Insufficient input validation in subsystem in Intel(R) AMT before versions 11.8.70, 11.11.70, 11.22.70 and 12.0.45 may allow an unauthenticated user to potentially enable denial of service or information disclosure via adjacent access.

CVSS Base Score: 7.1 High

CVSS Vector: CVSS:3.0/AV:A/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:H

- We wanted to patch-diff AMT versions to find it.

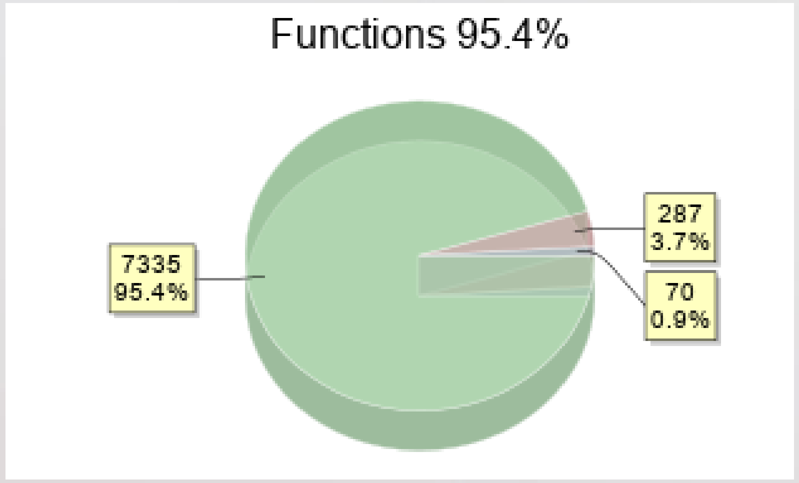
Intel ME: Patch-diffing

- We obtained two ME firmware versions:
  - Intel ME **12.0.32**.1421 Corporate/5MB
  - Intel ME **12.0.55**.1521 Corporate/5MB

- Used the ME Analyzer tool to unpack the firmware and extract the AMT module.

Intel ME: Patch-diffing

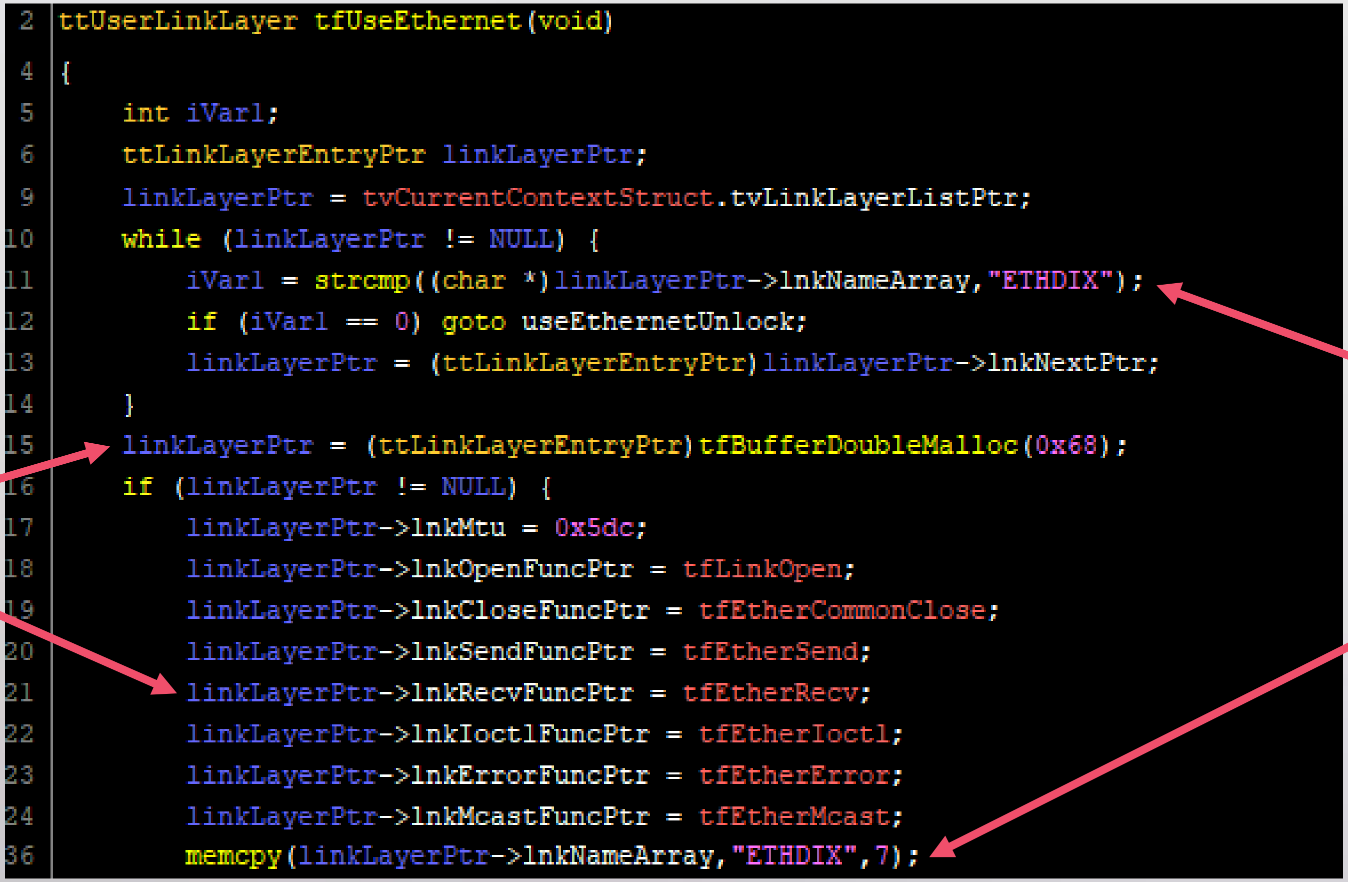
- We used BinDiff as our patch-diffing tool.



- Challenge:
  - diff is large.
  - We want to focus on Treck-related code only.

Intel ME: Finding Treck

- `tfUseEthernet` initializes the Ethernet link layer.



```
ttUserLinkLayer tfUseEthernet(void)
{
    int iVar1;
    ttLinkLayerEntryPtr linkLayerPtr;
    linkLayerPtr = tvCurrentContextStruct.tvLinkLayerListPtr;
    while (linkLayerPtr != NULL) {
        iVar1 = strcmp((char *)linkLayerPtr->lnkNameArray,"ETHDIX");
        if (iVar1 == 0) goto useEthernetUnlock;
        linkLayerPtr = (ttLinkLayerEntryPtr)linkLayerPtr->lnkNextPtr;
    }
    linkLayerPtr = (ttLinkLayerEntryPtr)tfBufferDoubleMalloc(0x68);
    if (linkLayerPtr != NULL) {
        linkLayerPtr->lnkMtu = 0x5dc;
        linkLayerPtr->lnkOpenFuncPtr = tfLinkOpen;
        linkLayerPtr->lnkCloseFuncPtr = tfEtherCommonClose;
        linkLayerPtr->lnkSendFuncPtr = tfEtherSend;
        linkLayerPtr->lnkRecvFuncPtr = tfEtherRecv;
        linkLayerPtr->lnkIoctlFuncPtr = tfEtherIoctl;
        linkLayerPtr->lnkErrorFuncPtr = tfEtherError;
        linkLayerPtr->lnkMcastFuncPtr = tfEtherMcast;
        memcpy(linkLayerPtr->lnkNameArray,"ETHDIX",7);
```

Initializes a struct with function pointers – `tfEtherRecv` among them.

Two references for the string "`ETHDIX`".

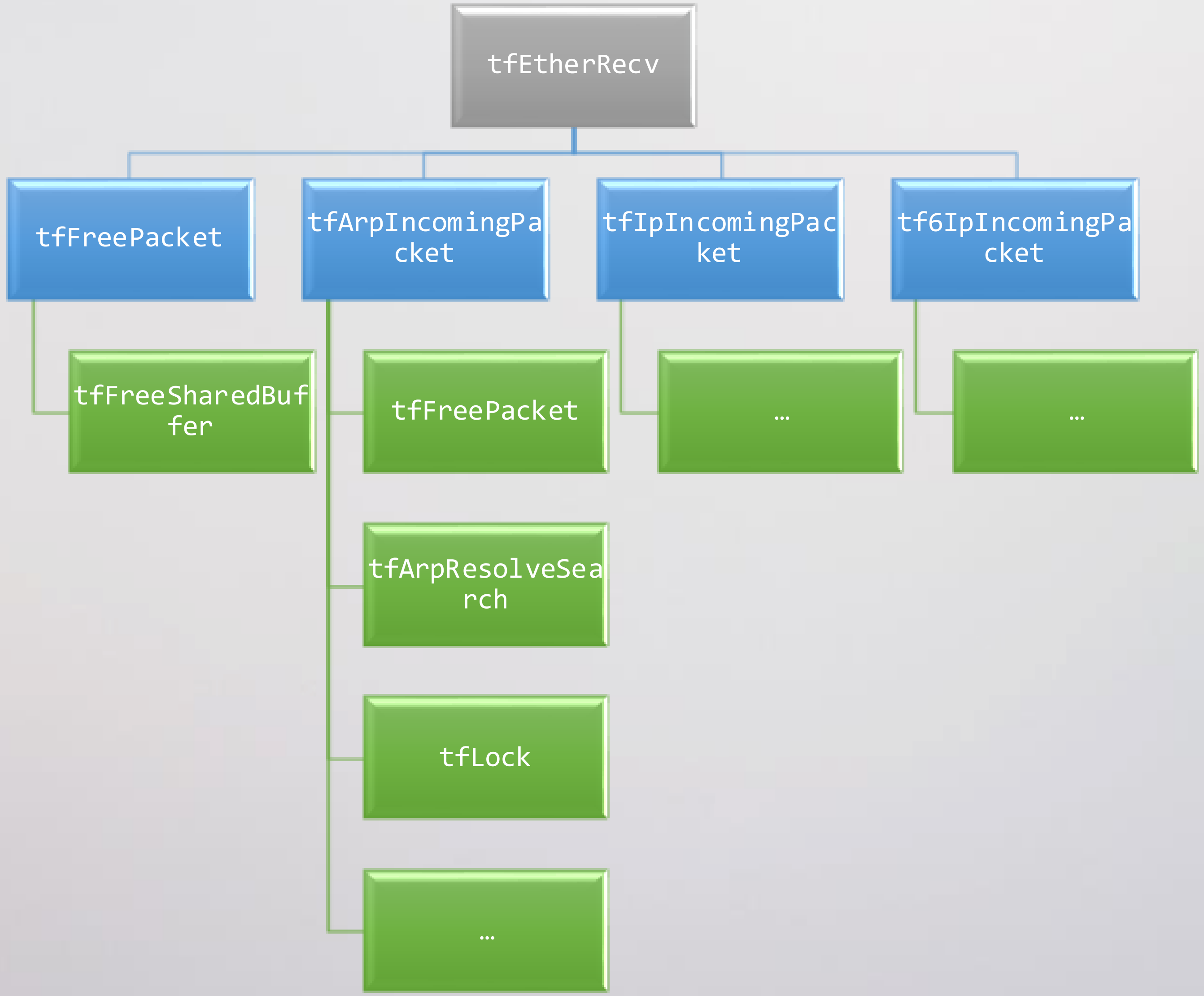
*Decompiled code taken from the Digi Connect device

Intel ME: Finding Treck

- We signed the `tfUseEthernet` function structure.
- Using the “`ETHDIX`” string we found the image base address.
- We developed a Ghidra script to mark Treck-related code, then extracted Ghidra symbols to IDA for diffing.

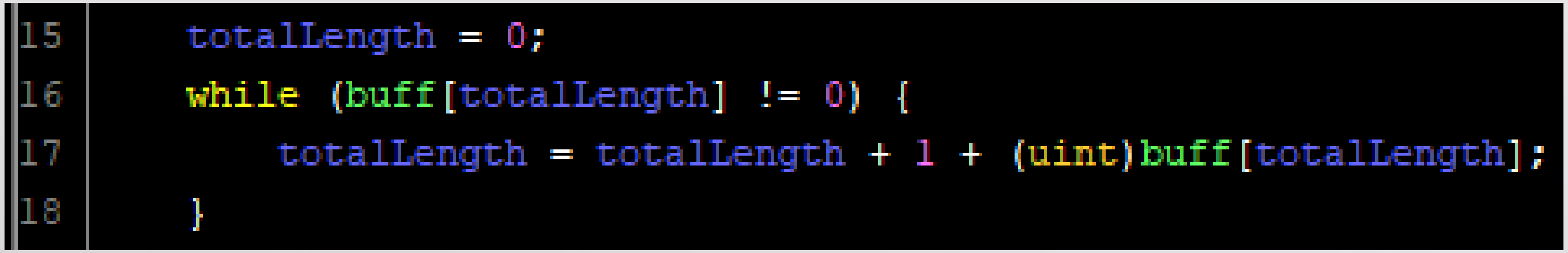
Marking Treck-related code

- Traverse call-graph from known-Treck entry points.
- Function pointers in `tfUseEthernet` as entry points.
- Luckily, library functions reside in separate module(s).
- To gain more coverage we considered parents of functions with many xrefs (e.g. `tfLock`).



Intel ME: A vulnerability

- A fixed bug was found in the DHCPv6 client:
  - During option 24 processing in `tf6DhcpSaveReplyInfo`.
  - Function accepts single argument (`buff`) and computes the total label length.

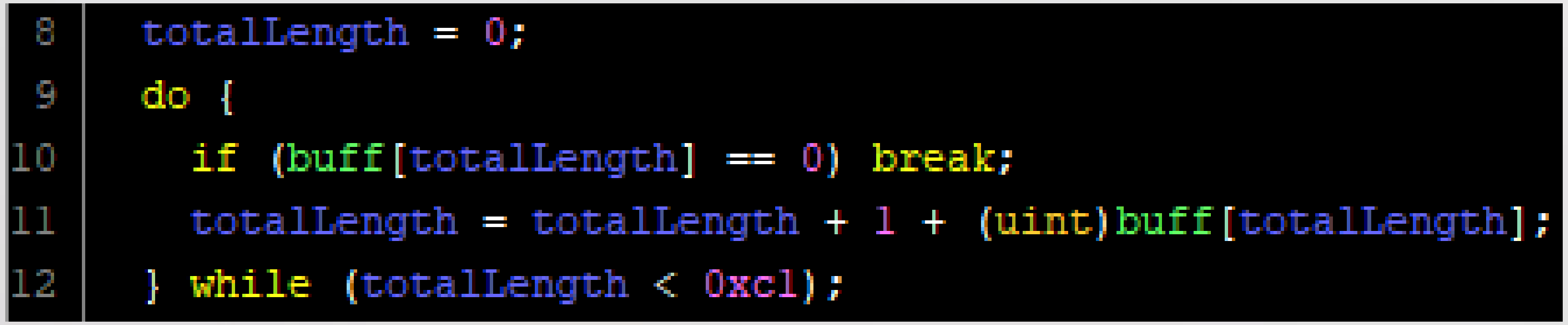


```
    totalLength = 0;
    while (buff[totalLength] != 0) {
        totalLength = totalLength + 1 + (uint)buff[totalLength];
    }
```

- Matches the description of CVE-2019-0131 shown earlier:
  - Adjacent access
  - Infoleak/DoS

Intel ME: A vulnerability

- We also found that Treck got the fix wrong:



```
totalLength = 0;
do {
  if (buff[totalLength] == 0) break;
  totalLength = totalLength + 1 + (uint)buff[totalLength];
} while (totalLength < 0xc1);
```

- Still OOB access.

- We reported the issue to Treck. This is CVE-2020-11905.

1 days, 0 days, Any-days

- Some of the vulnerabilities fixed **only** in Intel. Also, Intel has exploit mitigations.
- Digi had old code; Intel had new code. Intel had some code (no DNS)
- Until disclosure, we thought some bugs were **1-days** and Intel was most updated.
- Treck told us they are **0-days**. The story of AMT is unclear.

1 days, 0 days, Any-days

- Few types of Treck supply-chain vulnerabilities:
  - True 0-days
  - 0-days only fixed in AMT code (to our knowledge)
  - N-days that exist in the wild and fixed upstream - **Any-days**
    - Never publicly reported as far as we know
    - We don’t know if considered security fix previously

- Support package $\rightarrow$ updates
  - No support $\rightarrow$ no security

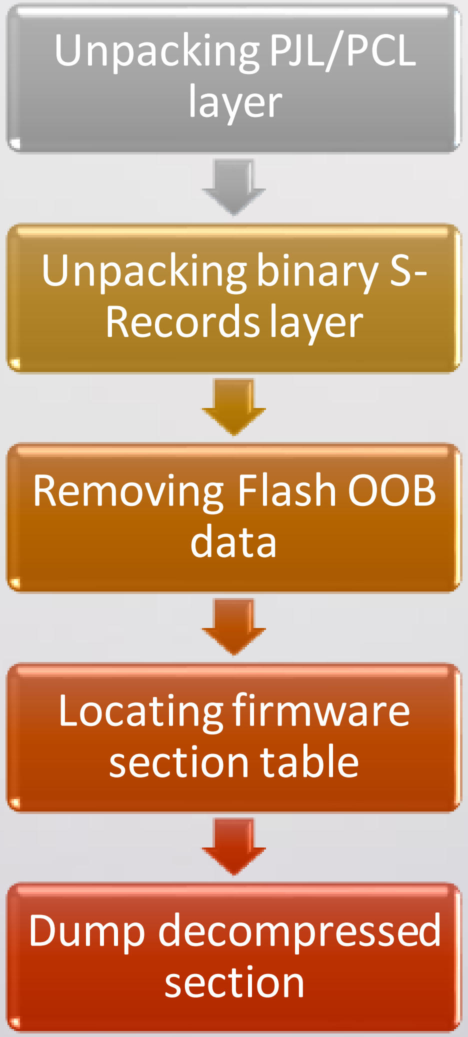
Binary #5 – HP printer

HP OfficeJet Pro 8720

- Searching for some common Treck function names on Google yields interesting results.
- We found that some HP printers run Treck.
- We wanted to check if they are affected by the vulnerabilities.

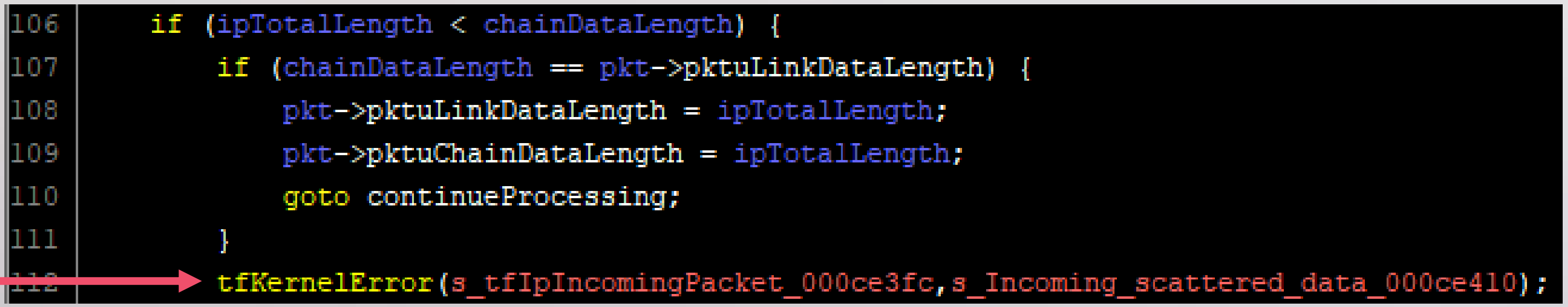
HP firmware unpacking

- RFU (remote firmware update) file obtained from HP's public FTP server.
- Multi–stage unpacking.
- Bizarre file formats.
- Whole process is described in our blog.


Unpacking PJL/PCL layer
Unpacking binary S-Records layer
Removing Flash OOB data
Locating firmware section table
Dump decompressed section

HP OfficeJet Pro 8720: A crash

- We found that CVE-2020-11896 crashes the printer.
- Apparently, there is an `ifdef` that disallows fragmented data over an IP-in-IP tunnel.
- However, sending those packets cause `tfKernelError` to run.
- Vulnerability variant.



```
    if (ipTotalLength < chainDataLength) {
        if (chainDataLength == pkt->pktuLinkDataLength) {
            pkt->pktuLinkDataLength = ipTotalLength;
            pkt->pktuChainDataLength = ipTotalLength;
            goto continueProcessing;
        }
        tfKernelError(s_tfIpIncomingPacket_000ce3fc,s_Incoming_scattered_data_000ce410);
```

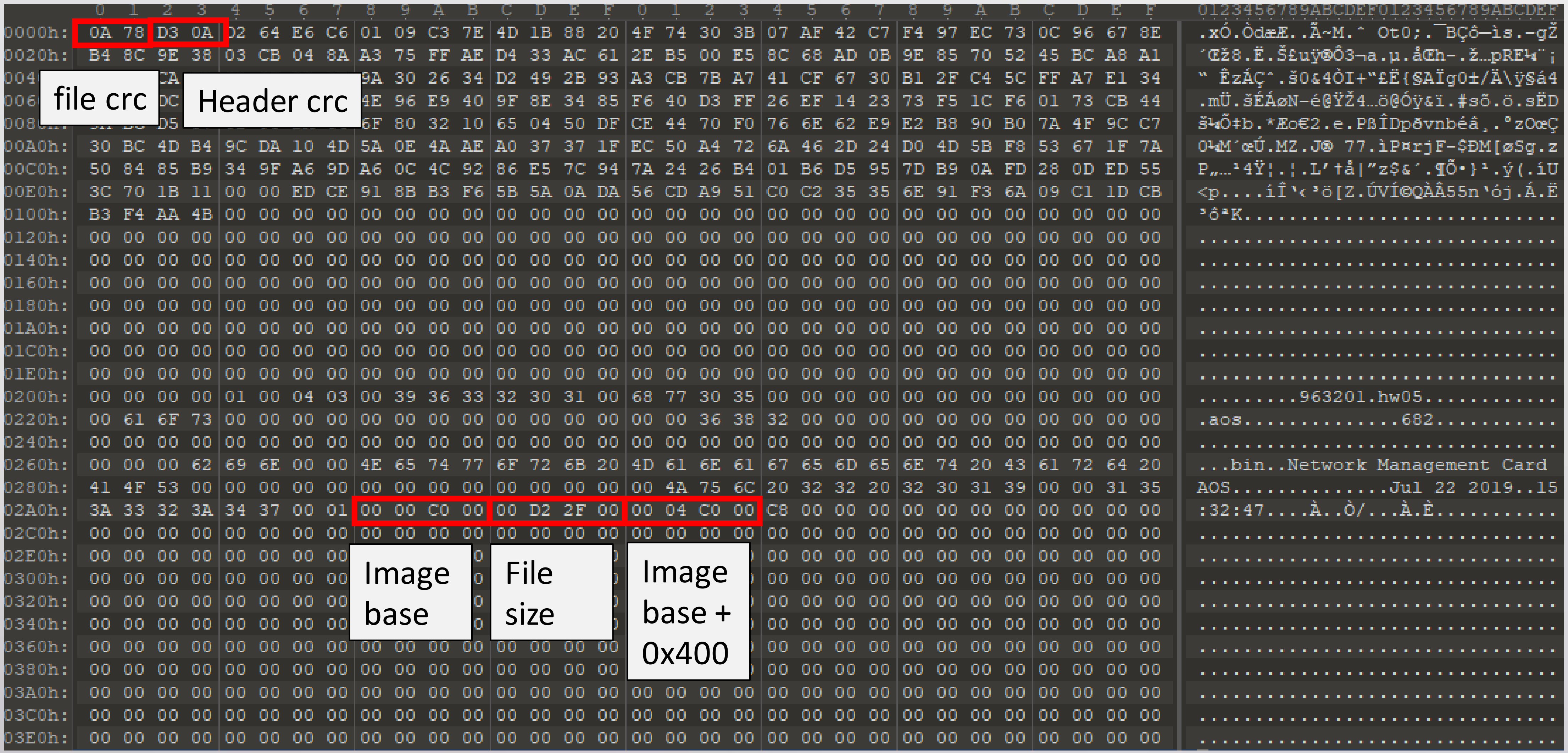
Crash!

Binary #6 – APC UPS

Schneider Electric APC UPS

- Downloaded firmware update files from APC website.
- Not encrypted/compressed.
- Reverse engineered some parts of the file format:
  - Image base address
  - CRC16 fields

The AOS binary



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF0123456789ABCDEF
0000h: 0A 78 D3 0A D2 64 E6 C6 01 09 C3 7E 4D 1B 88 20 4F 74 30 3B 07 AF 42 C7 F4 97 EC 73 0C 96 67 8E   .xÓ.ÒdæÆ..Ã~M.ˆ Ot0;.¯BÇô—ìs.–gŽ
0020h: B4 8C 9E 38 03 CB 04 8A A3 75 FF AE D4 33 AC 61 2E B5 00 E5 8C 68 AD 0B 9E 85 70 52 45 BC A8 A1   ´Œž8.Ë.Š£uÿ®Ô3¬a.µ.åŒh-.ž…pRE¼¨¡
0040h:       CA                9A 30 26 34 D2 49 2B 93 A3 CB 7B A7 41 CF 67 30 B1 2F C4 5C FF A7 E1 34   “ ÊzÁÇˆ.š0&4ÒI+“£Ë{§AÏg0±/Ä\ÿ§á4
0060h:       DC                4E 96 E9 40 9F 8E 34 85 F6 40 D3 FF 26 EF 14 23 73 F5 1C F6 01 73 CB 44   .mÜ.šÉÁøN–é@ŸŽ4…ö@Óÿ&ï.#sõ.ö.sËD
0080h:       D5                6F 80 32 10 65 04 50 DF CE 44 70 F0 76 6E 62 E9 E2 B8 90 B0 7A 4F 9C C7   š¼Õ‡b.*Æo€2.e.PßÎDpðvnbéâ¸.°zOœÇ
00A0h: 30 BC 4D B4 9C DA 10 4D 5A 0E 4A AE A0 37 37 1F EC 50 A4 72 6A 46 2D 24 D0 4D 5B F8 53 67 1F 7A   0¼M´œÚ.MZ.J® 77.ìP¤rjF-$ÐM[øSg.z
00C0h: 50 84 85 B9 34 9F A6 9D A6 0C 4C 92 86 E5 7C 94 7A 24 26 B4 01 B6 D5 95 7D B9 0A FD 28 0D ED 55   P„…¹4Ÿ¦.¦.L’†å|”z$&´.¶Õ•}¹.ý(.íU
00E0h: 3C 70 1B 11 00 00 ED CE 91 8B B3 F6 5B 5A 0A DA 56 CD A9 51 C0 C2 35 35 6E 91 F3 6A 09 C1 1D CB   <p....íÎ‘‹³ö[Z.ÚVÍ©QÀÂ55n‘ój.Á.Ë
0100h: B3 F4 AA 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ³ôªK............................
0120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
0140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
0160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
0180h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
01A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
01C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
01E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
0200h: 00 00 00 00 01 00 04 03 00 39 36 33 32 30 31 00 68 77 30 35 00 00 00 00 00 00 00 00 00 00 00 00   .........963201.hw05............
0220h: 00 61 6F 73 00 00 00 00 00 00 00 00 00 00 00 00 00 00 36 38 32 00 00 00 00 00 00 00 00 00 00 00   .aos..............682...........
0240h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
0260h: 00 00 00 62 69 6E 00 00 4E 65 74 77 6F 72 6B 20 4D 61 6E 61 67 65 6D 65 6E 74 20 43 61 72 64 20   ...bin..Network Management Card 
0280h: 41 4F 53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 4A 75 6C 20 32 32 20 32 30 31 39 00 00 31 35   AOS..............Jul 22 2019..15
02A0h: 3A 33 32 3A 34 37 00 01 00 00 C0 00 00 D2 2F 00 00 04 C0 00 C8 00 00 00 00 00 00 00 00 00 00 00   :32:47....À..Ò/...À.È...........
02C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
02E0h: 00 00 00 00 00 00 00 00                         00 00 00 00 00 00 00 00 00 00 00 00   ................................
0300h: 00 00 00 00 00 00 00 00                         00 00 00 00 00 00 00 00 00 00 00 00   ................................
0320h: 00 00 00 00 00 00 00 00                         00 00 00 00 00 00 00 00 00 00 00 00   ................................
0340h: 00 00 00 00 00 00 00 00                         00 00 00 00 00 00 00 00 00 00 00 00   ................................
0360h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00             00 00 00 00 00 00 00 00 00 00 00 00   ................................
0380h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00             00 00 00 00 00 00 00 00 00 00 00 00   ................................
03A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
03C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
03E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................................
```

The AOS binary: Which processor?

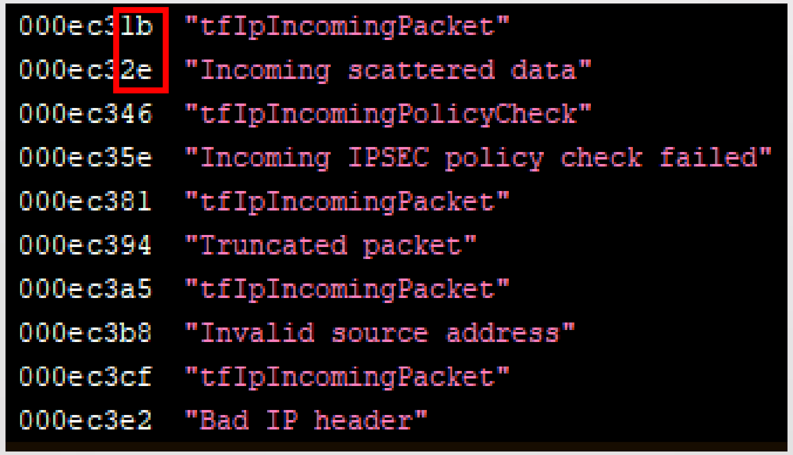
- Loaded into Ghidra.
- Choosing 16-bit x86 protected mode “kind of” works.
- Disassembler cannot resolve far-calls.
- Obscure architecture.

The AOS binary: Which processor?

- Strange memory addressing.
- Can’t be protected mode - too many segments, no GDT.
- Can’t be real mode - shifting by 4 does not work.
- We opened the old books to find olden x86 witchcraft – no luck.
  - Do you have a moment to learn about **unreal mode?**

The AOS binary: Which processor?

- A pattern emerges when looking at the strings/function calls (but mostly strings).
- LSB of a pushed string corresponds to LSB of the offset of the string within the binary.
- Shifting the segment word by 8 does the trick. We saw that we always land on a function this way.



```
000ec31b  "tfIpIncomingPacket"
000ec32e  "Incoming scattered data"
000ec346  "tfIpIncomingPolicyCheck"
000ec35e  "Incoming IPSEC policy check failed"
000ec381  "tfIpIncomingPacket"
000ec394  "Truncated packet"
000ec3a5  "tfIpIncomingPacket"
000ec3b8  "Invalid source address"
000ec3cf  "tfIpIncomingPacket"
000ec3e2  "Bad IP header"
```

```
push cs
push 0xf2e
push cs
push 0xf1b
call 0xc466:0x382  <- tfKernelError
```

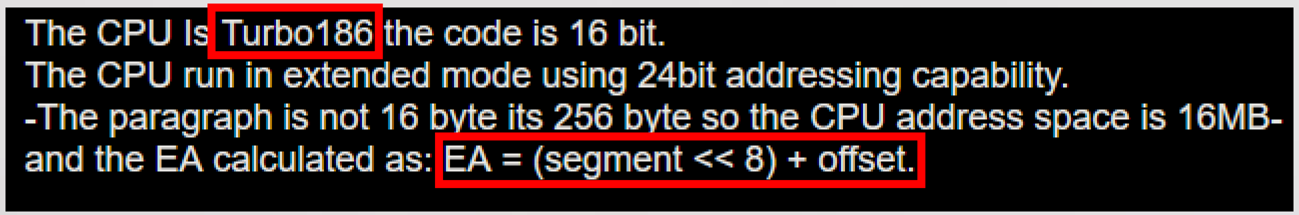
```
LinearAddress = (segment << 8) + offset
```

The AOS binary: Loading into RE tool

- We must fix the far-call issue to reverse engineer the firmware.
- We tried to change Ghidra’s processor module, recompile it.
- Only partial success, no strings.
- We tried to specify the segment granularity on radare2 – better, still lacks strings.

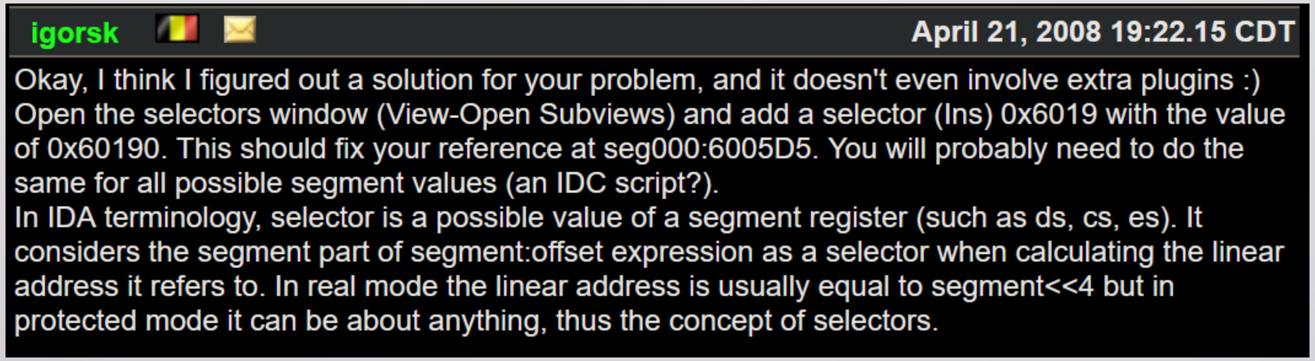
The AOS binary: Loading into RE tool

- We found someone who faced the same issue on http://www.openrce.org/forums/posts/753.
- Mystery solved: Turbo186!



The CPU Is Turbo186 the code is 16 bit.
The CPU run in extended mode using 24bit addressing capability.
-The paragraph is not 16 byte its 256 byte so the CPU address space is 16MB-
and the EA calculated as: EA = (segment << 8) + offset.

- Solution: use IDA's segment selectors.
- Thanks igor skochinsky.



igorsk April 21, 2008 19:22.15 CDT

Okay, I think I figured out a solution for your problem, and it doesn't even involve extra plugins :)
Open the selectors window (View-Open Subviews) and add a selector (Ins) 0x6019 with the value of 0x60190. This should fix your reference at seg000:6005D5. You will probably need to do the same for all possible segment values (an IDC script?).
In IDA terminology, selector is a possible value of a segment register (such as ds, cs, es). It considers the segment part of segment:offset expression as a selector when calculating the linear address it refers to. In real mode the linear address is usually equal to segment<<4 but in protected mode it can be about anything, thus the concept of selectors.

The AOS binary: Loading into IDA

- We wrote an IDA python script to create segment selectors which emulate the “shifting by 8”.
- Now we have strings, switch statements, far-calls working.
- We can start reverse engineering.
- No decompiler for 16-bit x86.

The AOS binary: Heap Functions

- Even after segment fixing, many far-calls point to non-mapped regions
- Comparing with Digi firmware, we concluded these are far-calls to heap utility functions
  - *`malloc()`*, *`free()`*, etc.
- The binary contains debug strings with the function names
  - But without references…
- Because of the 8-bit segment shift, we can search for undefined push instructions
- Found and re-mapped these functions to their proper dynamic location

New Vulnerability

- Found newer Treck version than Digi’s.
- Also new vulnerability (CVE-2020-11901: Bad RDLENGTH).
- Bad fix for a previously found vulnerability.
- Didn’t exist in AMT because they don’t use this feature.

Binary #7 – GE MDS

Bonus Binary: GE MDS

- General Electric communication device for utilities (water/power).
- Used Google search + n-gram slices to find the architecture
  - `cpu_rec` works too!
- Runs Blackfin processor and uses Treck.
- Didn’t use extensively, didn’t teach us anything new.

Wrap-up

Take-aways

- Supply chain is complicated
- Obscurity doesn’t work
  - Well, mostly.
- Know your upstream, patch your upstream
- Deeper in the supply-chain $\rightarrow$ higher impact

Take-aways

- Software providers security SLA
  - Report security issues?
  - Timeline?
  - Product support vs security support?
  - Two way? What if user finds a vulnerability?

- Proprietary vs. OSS?

Inconsistent patching

- One vendor patches and another doesn’t.

- Patch-gapping on steroids!

Conclusions

- Complex reverse engineering process
- Forks in software library can unveil more vulnerabilities
- Supply chain makes security difficult
- Proprietary update process is obscure

JSOF

Thanks for listening!

info@jsof-tech.com