NAVSEA WARFARE CENTERS Port Hueneme

Intro To Bug Hunting

From Manual Reverse Engineering to Intelligent Automation

NAVAL SURFACE WARFARE CENTER
PORT HUENEME DIVISION

-October 11, 2019-

Socrates Frangis
NSWC Port Hueneme Cybersecurity Technical Lead

DISTRIBUTION A. Approved for public release: distribution unlimited.

Disclaimer

The views and opinions expressed in this talk are my own and NOT necessarily those of the United States Navy

¯_(ツ)_/¯

DISTRIBUTION A. Approved for public release: distribution unlimited.

General Outline

- Who Am I?
- Where Do I Work?
- What is Bug Hunting?
- Static vs Dynamic Analysis
- Free Tools…
- Capture The Flag
- DARPA Cyber Grand Challenge
- Defense Innovation Unit (DIU) Project Voltron Pilot Effort
- Questions

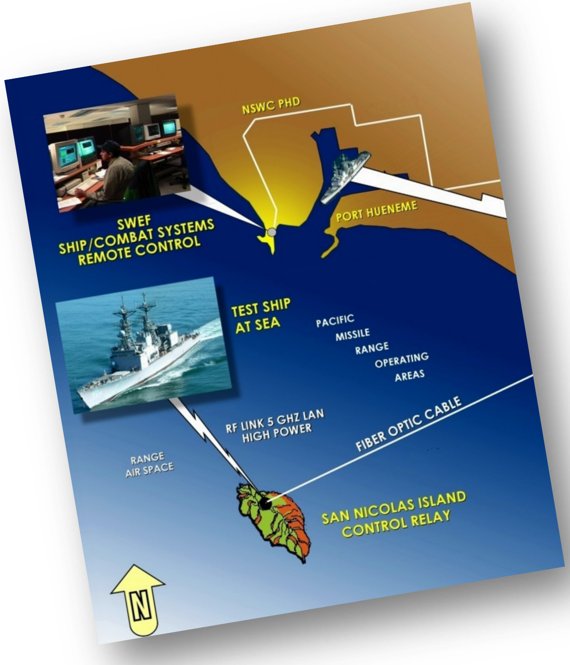
DISTRIBUTION A. Approved for public release: distribution unlimited.

Who Am I?

- **Naval Sea Systems Command – NSWC Port Hueneme Division**
    - Cybersecurity Technical Lead
    - 11 Years with USN
- **Security Researcher – Reverse Engineering Software & Embedded Systems**
    - Vulnerability Assessments, Bug Hunting, Red Team Support
- **Academia**
    - BS Applied Mathematics & Computer Science (UC Santa Barbara)
    - MS Systems Engineering (Naval Postgraduate School)
    - PhD* Systems Engineering w/ Focus in Cybersecurity (Naval Postgraduate School)

NAVSEA WARFARE CENTERS
Port Hueneme

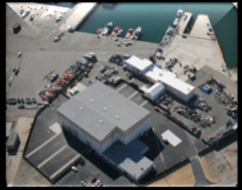
Where Do I Work?



Deep Water Port

SDTS

MPSF



UNREP Site

SWEF

Desert Ship

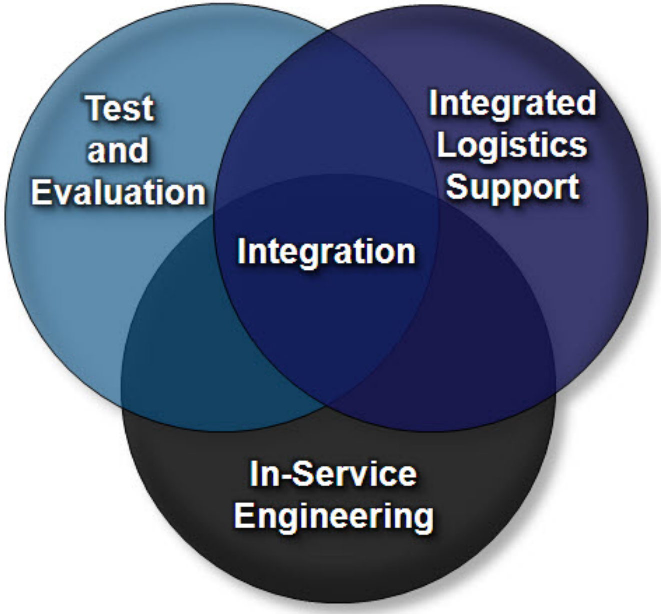
Radar Lab

DISTRIBUTION A. Approved for public release: distribution unlimited.

NAVSEA WARFARE CENTERS Port Hueneme

Where Do I Work?

- AEGIS Combat System
- Air and Missile Defense Radar
- Ballistic Missile Defense (BMD)
- Battle Force Interoperability
- Cooperative Engagement Capability (CEC)
- DDG-1000 Combat Systems
- Dual Band Radar
- Evolved NATO Seasparrow Missile (ESSM)
- Gun Fire Control Systems
- HARPOON Weapon System
- HE Laser Weapon System
- LCS Combat Systems
- Missile Launching Systems
- Mission Modules



Mission

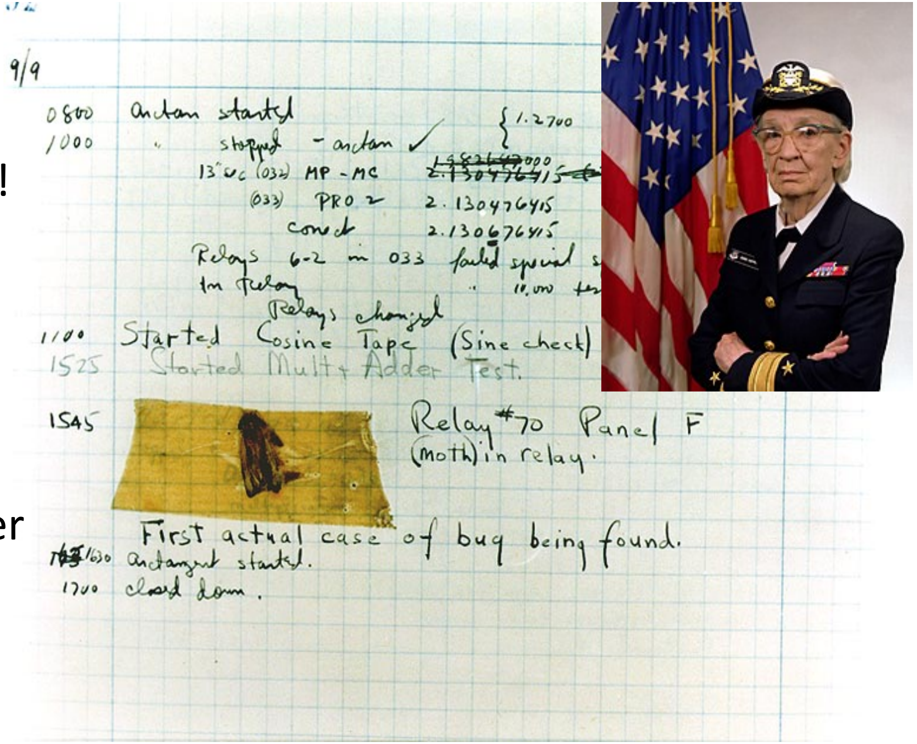
- Naval Fires Control System
- Naval Integrated Fire Control-Counter Air
- NATO Seasparrow Missile System
- Rolling Airframe Missile System
- Search Radars
- Ship Self Defense System (SSDS)
- Standard Missile
- Supporting Arms Coordination Center- Automated
- TOMAHAWK Weapon Control System (All Variants) and TOMAHAWK All Up Round
- Underway Replenishment Systems

DISTRIBUTION A. Approved for public release: distribution unlimited.

NAVSEA WARFARE CENTERS Port Hueneme

History of the Term Bug

- **Term "bug" had existed for years**
    - Edison even used it (1873)
    - 9/9/47 ~ One was actually found!
- **MK II Aiken Relay Calculator**
    - Built at Harvard (1947)
    - Financed by Navy
- **RDML Grace Hopper other firsts…**
    - General Purpose Digital Computer
    - Compilers & Linkers
    - Programming Language (COBOL)
    - USN Female Admiral



9/9

0800 Antan started
1000 " stopped - antan ✓ {1.2700
13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415~~
(033) PRO 2 2.130476415
const 2.130676415
Relays 6-2 in 033 failed special s
In Relay " 11,000 tes
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~163~~ 1630 Antangent started.
1700 closed down.

DISTRIBUTION A. Approved for public release: distribution unlimited.

Bug Hunting

- **What is Bug Hunting?**
  - Looking for previously “unknown” software defects
  - Hopefully ones that are security vulnerabilities
- **How?**
  - Through static and dynamic analysis
  - Responsibly disclosing to the developer so they can fix the problem
  - Usually through a trusted third party
- **Programs**

HACK THE PENTAGON

hackerone

bugcrowd

DISTRIBUTION A. Approved for public release: distribution unlimited.

Static vs. Dynamic Analysis

- **Static Analysis**
  - Non runtime environment
  - Typically inspects source code or disassembled binary
  - Time consuming (manual)
  - Quick results if automated
  - Can’t find vulnerabilities in runtime environment
  - Akin to dissection

- **Dynamic Analysis**
  - Runtime environment
  - Inspects compiled binary while it is running (no code required)
  - Time consuming (automated)
  - No quick results
  - Can find vulnerabilities in a runtime environment
  - Akin to vivisection

DISTRIBUTION A. Approved for public release: distribution unlimited.

Example ~ Hello World! (Code)

```
#include <stdio.h>
int main(int argc, char **argv) {              //example hello.c
   char name[16];                              //setup a variable to hold input
   char date[32]="date";                       //setup a variable to call date
   printf("Enter your name: ");                //print a message
   gets(name);                                 //get the users name as input
   printf("\nHello %s! The date is ", name);  //print a message
   system(date);                               //make a system call for datetime
}
```

```
(PROGRAM RUN)
n00b@lappy:~$ hello
Enter your name: Socrates
Hello Socrates! The date is Tue Jul 30 15:33:33 UTC 2019
```

This seems okay…right?

NAVSEA WARFARE CENTERS Port Hueneme

Example ~ Hello World! (Code)

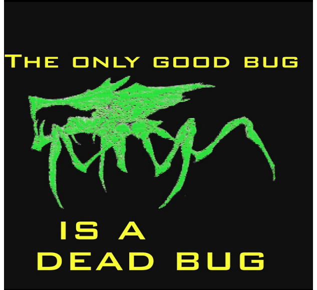
```
#include <stdio.h>
int main(int argc, char **argv) {
   char name[16];
   char date[32]="date";
   printf("Enter your name: ");
   gets(name);
   printf("\nHello %s! The date is ", name);
   system(date);
}
```

```
hax0r@lappy:~$ hello
Enter your name: xxxxxxxxxxxxxxxxcat /etc/shadow
Hello xxxxxxxxxxxxxxxx The date is:
root:$6$Ke02nYgo.9v0SF4p$hjztYvo/M4buqO4oBX8KZTftjCn6fE4cV5o/I95QPekeQpITwFTRbDUBYBLI
Ux2mhorQoj9bLN8v.w6btE9xy1:16431:0:99999:7:::
mysql:!!:16550:::::: ...continued...
```

DISTRIBUTION A. Approved for public release: distribution unlimited.

Free Tools

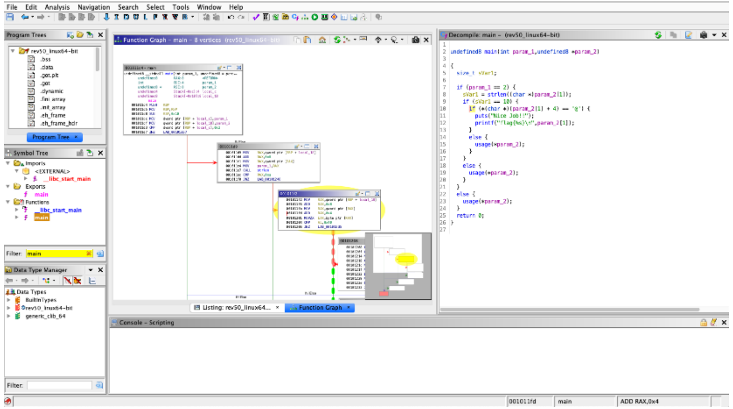
- **So how would we find these kinds of bugs?**
  - Static Analysis Tools
    - Text Editor + Source Code + Eyeball(s)
    - Ghidra + Compiled Binary + Eyeball(s)
  - Dynamic Analysis Tools
    - American Fuzzy Lop + Source Code
    - Google Open Source Software Fuzz + Source Code
    - American Fuzzy Lop + Unicorn Engine + Compiled Binary
    - ANGR + Compiled Binary



DISTRIBUTION A. Approved for public release: distribution unlimited.

Free Tools - Ghidra

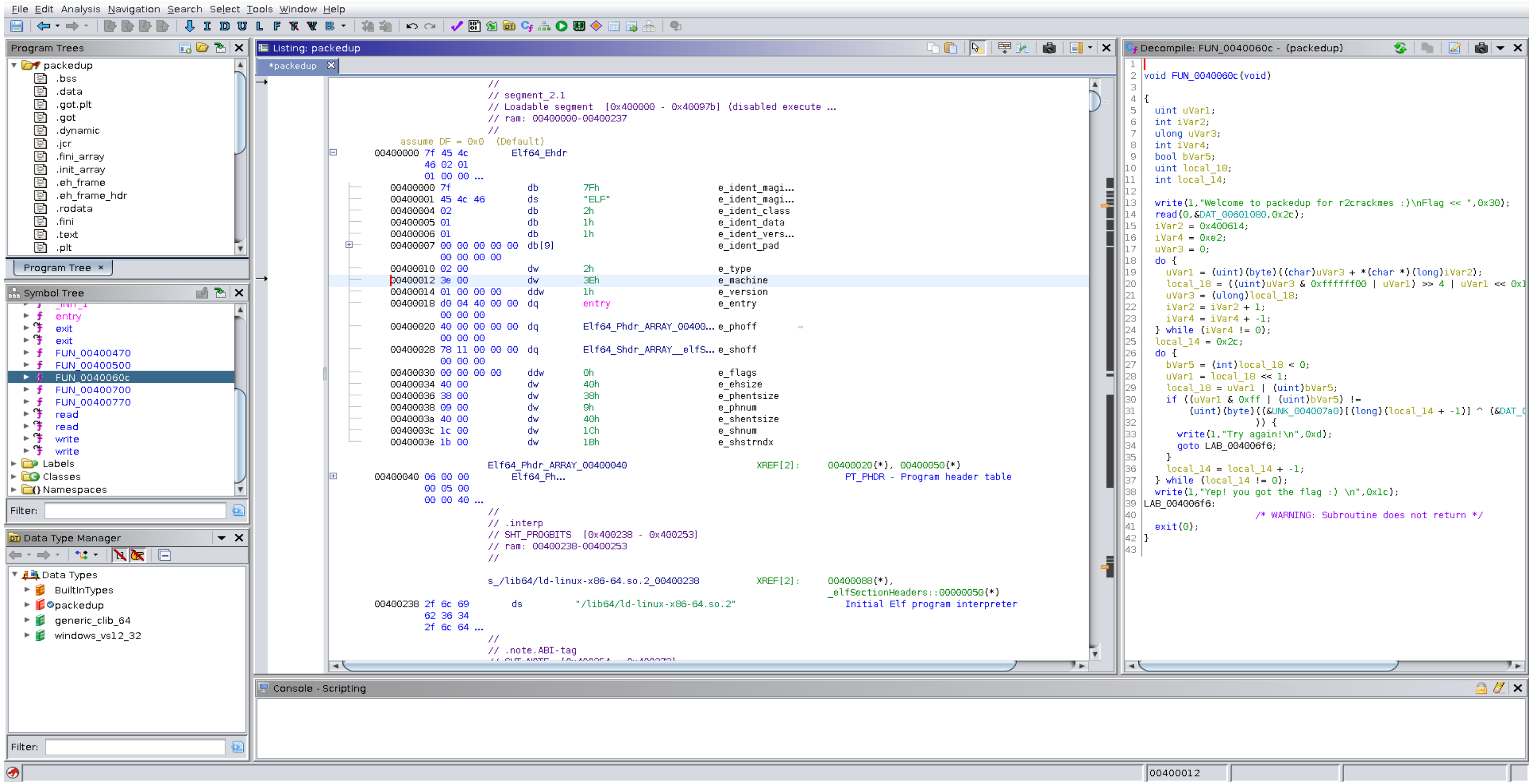
- **NSA reverse engineering tool released as free open source**
  - RSA Conference 2019 from Rob Joyce
- **Allows you to statically analyze a compiled binary**
  - Shows disassembled machine code
  - Shows decompiled “pseudo” source code
  - Shows an applications control flow graph
- **Supports many computing architectures**
  - x86 16, 32, and 64 bit
  - ARM, AVR, AARCH64, PowerPC
  - MIPS, JAVA, PIC, SPARC



```

undefined8 main(int param_1,undefined8 *param_2)

{
  size_t sVar1;

  if (param_1 == 2) {
    sVar1 = strlen((char *)param_2[1]);
    if (sVar1 == 10) {
      if (*(char *)(param_2[1] + 4) == '@') {
        puts("Nice Job!!");
        printf("flag{%s}\n",param_2[1]);
      }
      else {
        usage(*param_2);
      }
    }
    else {
      usage(*param_2);
    }
  }
  else {
    usage(*param_2);
  }
  return 0;
}

```

DISTRIBUTION A. Approved for public release: distribution unlimited.

Free Tools - Ghidra



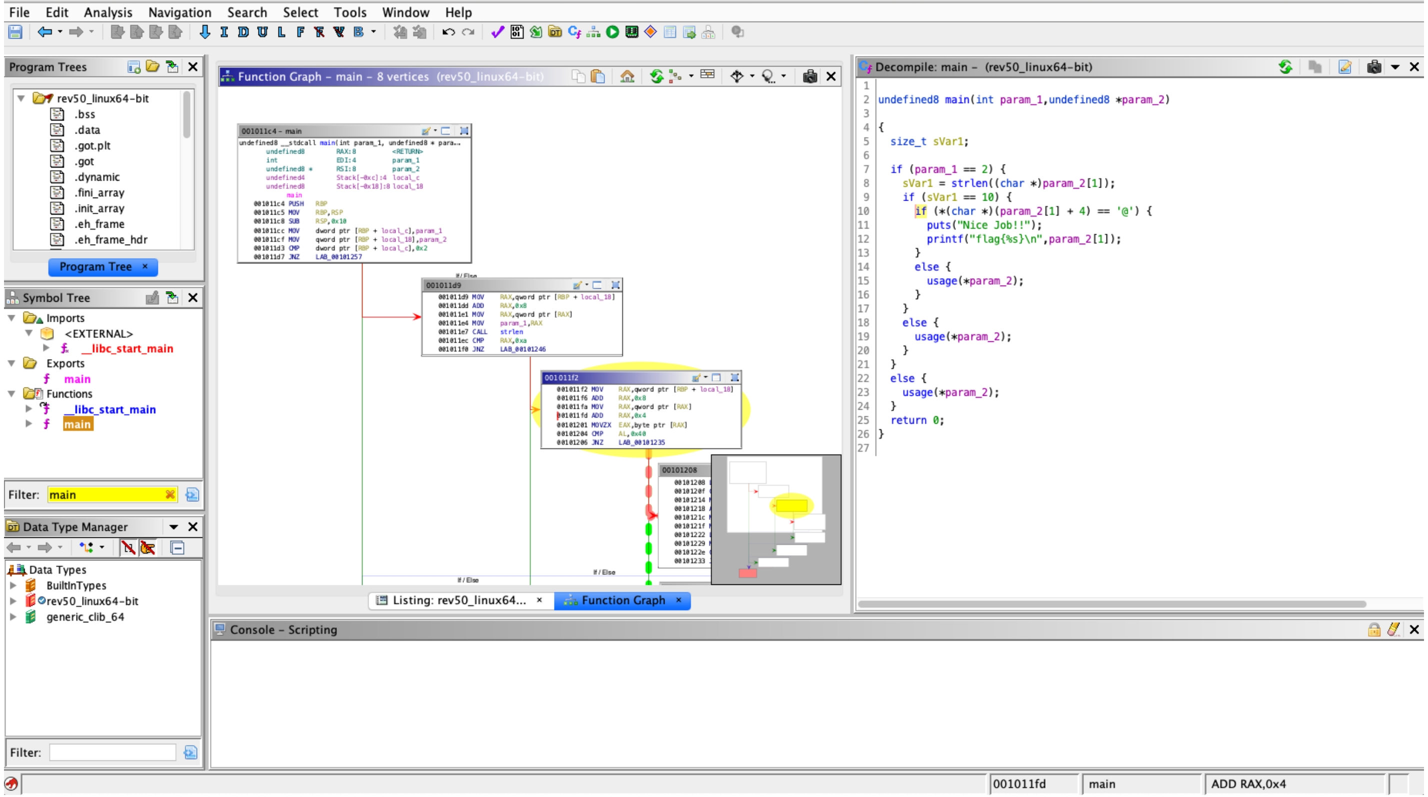
```
                       //
                       // segment_2.1
                       // Loadable segment  [0x400000 - 0x40097b] (disabled execute ...
                       // ram: 00400000-00400237
                       //
             assume DF = 0x0  (Default)
        00400000 7f 45 4c        Elf64_Ehdr
                 46 02 01
                 01 00 00 ...
           00400000 7f              db        7Fh                    e_ident_magi...
           00400001 45 4c 46        ds        "ELF"                  e_ident_magi...
           00400004 02              db        2h                     e_ident_class
           00400005 01              db        1h                     e_ident_data
           00400006 01              db        1h                     e_ident_vers...
           00400007 00 00 00 00 00  db[9]                            e_ident_pad
                    00 00 00 00
           00400010 02 00           dw        2h                     e_type
           00400012 3e 00           dw        3Eh                    e_machine
           00400014 01 00 00 00     ddw       1h                     e_version
           00400018 d0 04 40 00 00  dq        entry                  e_entry
                    00 00 00
           00400020 40 00 00 00 00  dq        Elf64_Phdr_ARRAY_00400... e_phoff
                    00 00 00
           00400028 78 11 00 00 00  dq        Elf64_Shdr_ARRAY__elfS... e_shoff
                    00 00 00
           00400030 00 00 00 00     ddw       0h                     e_flags
           00400034 40 00           dw        40h                    e_ehsize
           00400036 38 00           dw        38h                    e_phentsize
           00400038 09 00           dw        9h                     e_phnum
           0040003a 40 00           dw        40h                    e_shentsize
           0040003c 1c 00           dw        1Ch                    e_shnum
           0040003e 1b 00           dw        1Bh                    e_shstrndx

                       Elf64_Phdr_ARRAY_00400040           XREF[2]:     00400020(*), 00400050(*)
        00400040 06 00 00          Elf64_Ph...                            PT_PHDR - Program header table
                 00 05 00
                 00 00 40 ...
                       //
                       // .interp
                       // SHT_PROGBITS  [0x400238 - 0x400253]
                       // ram: 00400238-00400253
                       //

                       s_/lib64/ld-linux-x86-64.so.2_00400238         XREF[2]:     00400088(*),
                                                                                   _elfSectionHeaders::00000050(*)
        00400238 2f 6c 69        ds         "/lib64/ld-linux-x86-64.so.2"            Initial Elf program interpreter
                 62 36 34
                 2f 6c 64 ...
                       //
                       // .note.ABI-tag
```

```c

void FUN_0040060c(void)

{
  uint uVar1;
  int iVar2;
  ulong uVar3;
  int iVar4;
  bool bVar5;
  uint local_18;
  int local_14;

  write(1,"Welcome to packedup for r2crackmes :)\nFlag << ",0x30);
  read(0,&DAT_00601080,0x2c);
  iVar2 = 0x400614;
  iVar4 = 0xe2;
  uVar3 = 0;
  do {
    uVar1 = (uint)(byte)((char)uVar3 + *(char *)(long)iVar2);
    local_18 = ((uint)uVar3 & 0xffffff00 | uVar1) >> 4 | uVar1 << 0x1
    uVar3 = (ulong)local_18;
    iVar2 = iVar2 + 1;
    iVar4 = iVar4 + -1;
  } while (iVar4 != 0);
  local_14 = 0x2c;
  do {
    bVar5 = (int)local_18 < 0;
    uVar1 = local_18 << 1;
    local_18 = uVar1 | (uint)bVar5;
    if ((uVar1 & 0xff | (uint)bVar5) !=
        (uint)(byte)((&UNK_004007a0)[(long)(local_14 + -1)] ^ (&DAT_0
                    )) {
      write(1,"Try again!\n",0xd);
      goto LAB_004006f6;
    }
    local_14 = local_14 + -1;
  } while (local_14 != 0);
  write(1,"Yep! you got the flag :) \n",0x1c);
LAB_004006f6:
                    /* WARNING: Subroutine does not return */
  exit(0);
}

```

DISTRIBUTION A. Approved for public release: distribution unlimited.

NAVSEA WARFARE CENTERS
Port Hueneme

Free Tools - Ghidra



```
001011c4 - main
undefined8 __stdcall main(int param_1, undefined8 * para...
     undefined8        RAX:8            <RETURN>
     int               EDI:4            param_1
     undefined8 *      RSI:8            param_2
     undefined4        Stack[-0xc]:4    local_c
     undefined8        Stack[-0x18]:8   local_18
               main
   001011c4 PUSH     RBP
   001011c5 MOV      RBP,RSP
   001011c8 SUB      RSP,0x10
   001011cc MOV      dword ptr [RBP + local_c],param_1
   001011cf MOV      qword ptr [RBP + local_18],param_2
   001011d3 CMP      dword ptr [RBP + local_c],0x2
   001011d7 JNZ      LAB_00101257
```

```
001011d9
   001011d9 MOV      RAX,qword ptr [RBP + local_18]
   001011dd ADD      RAX,0x8
   001011e1 MOV      RAX,qword ptr [RAX]
   001011e4 MOV      param_1,RAX
   001011e7 CALL     strlen
   001011ec CMP      RAX,0xa
   001011f0 JNZ      LAB_00101246
```

```
001011f2
   001011f2 MOV      RAX,qword ptr [RBP + local_18]
   001011f6 ADD      RAX,0x8
   001011fa MOV      RAX,qword ptr [RAX]
   001011fd ADD      RAX,0x4
   00101201 MOVZX    EAX,byte ptr [RAX]
   00101204 CMP      AL,0x40
   00101206 JNZ      LAB_00101235
```

```
00101208
   00101208
   0010120f
   00101214
   00101218
   0010121c
   0010121f
   00101222
   00101229
   0010122e
   00101233
```

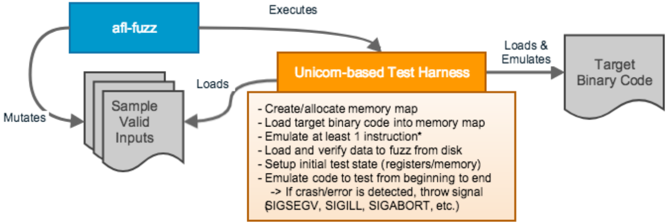
```

undefined8 main(int param_1,undefined8 *param_2)

{
  size_t sVar1;

  if (param_1 == 2) {
    sVar1 = strlen((char *)param_2[1]);
    if (sVar1 == 10) {
      if (*(char *)(param_2[1] + 4) == '@') {
        puts("Nice Job!!");
        printf("flag{%s}\n",param_2[1]);
      }
      else {
        usage(*param_2);
      }
    }
    else {
      usage(*param_2);
    }
  }
  else {
    usage(*param_2);
  }
  return 0;
}

```

DISTRIBUTION A. Approved for public release: distribution unlimited.

NAVSEA WARFARE CENTERS Port Hueneme

Free Tools – American Fuzzy Lop

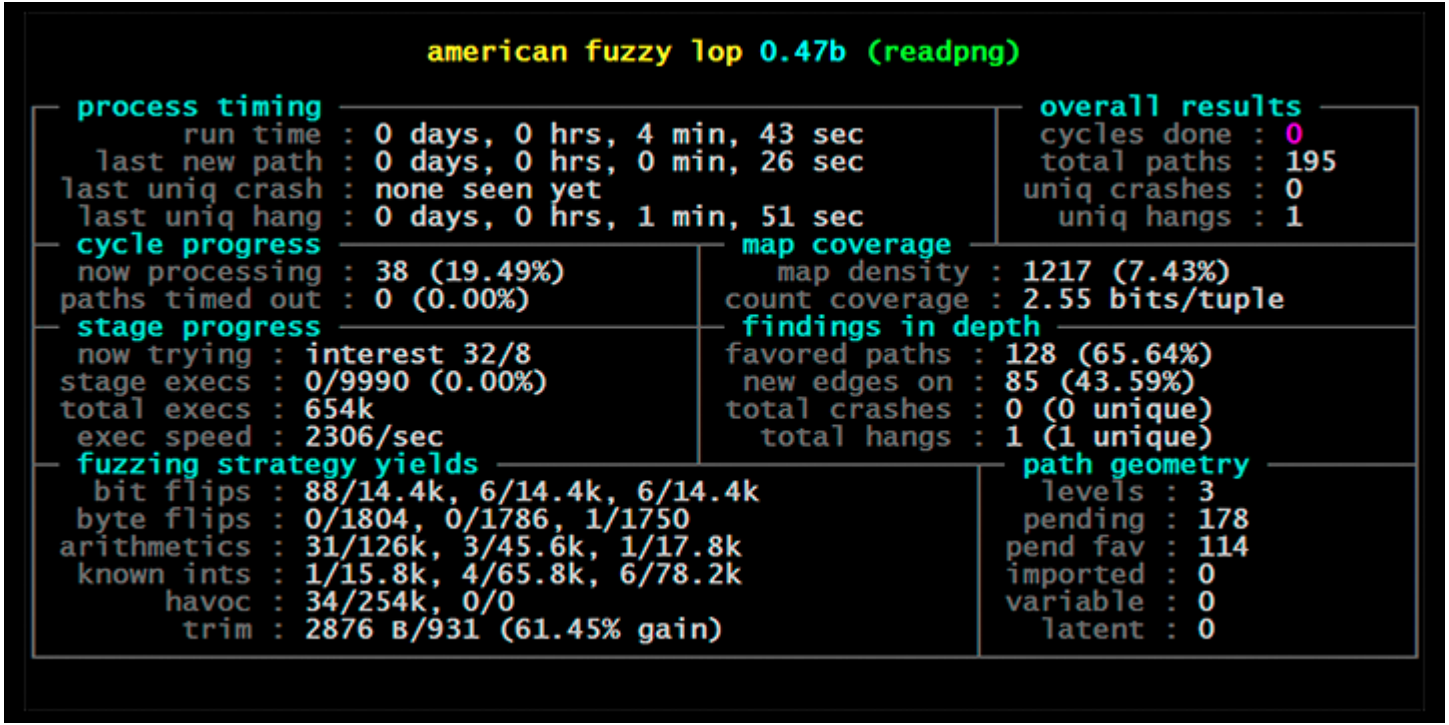
- **Automated fuzzing tool released as free open source**
  - Fuzzing – a QA technique to discover bugs via injecting random data to input fields
- **Allows you to dynamically analyze a binary**
  - By compiling source code with AFL
    - Then analysis with afl-fuzz engine
  - By injecting a pre-compiled binary
    - Then analysis with AFL-unicorn engine



- Create/allocate memory map
- Load target binary code into memory map
- Emulate at least 1 instruction*
- Load and verify data to fuzz from disk
- Setup initial test state (registers/memory)
- Emulate code to test from beginning to end
  -> If crash/error is detected, throw signal (SIGSEGV, SIGILL, SIGABORT, etc.)

DISTRIBUTION A. Approved for public release: distribution unlimited.

Free Tools – American Fuzzy Lop



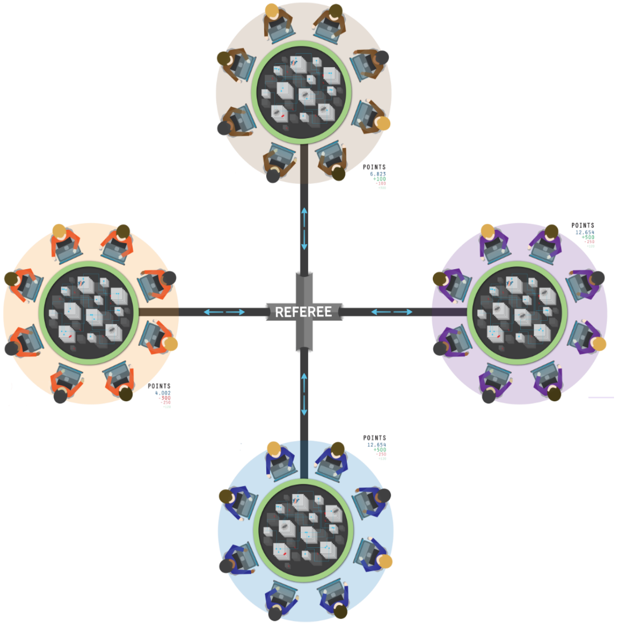
```
                    american fuzzy lop 0.47b (readpng)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 4 min, 43 sec       │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 0 min, 26 sec       │  total paths : 195    │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 0 days, 0 hrs, 1 min, 51 sec       │   uniq hangs : 1      │
├─ cycle progress ──────────────────┬─ map coverage ───┴───────────────────────┤
│  now processing : 38 (19.49%)     │    map density : 1217 (7.43%)           │
│ paths timed out : 0 (0.00%)       │ count coverage : 2.55 bits/tuple        │
├─ stage progress ──────────────────┼─ findings in depth ─────────────────────┤
│  now trying : interest 32/8       │ favored paths : 128 (65.64%)            │
│ stage execs : 0/9990 (0.00%)      │  new edges on : 85 (43.59%)             │
│ total execs : 654k                │ total crashes : 0 (0 unique)            │
│  exec speed : 2306/sec            │   total hangs : 1 (1 unique)            │
├─ fuzzing strategy yields ─────────┴─────────────┬─ path geometry ───────────┤
│   bit flips : 88/14.4k, 6/14.4k, 6/14.4k         │    levels : 3             │
│  byte flips : 0/1804, 0/1786, 1/1750             │   pending : 178           │
│ arithmetics : 31/126k, 3/45.6k, 1/17.8k          │  pend fav : 114           │
│  known ints : 1/15.8k, 4/65.8k, 6/78.2k          │  imported : 0             │
│       havoc : 34/254k, 0/0                       │  variable : 0             │
│        trim : 2876 B/931 (61.45% gain)           │    latent : 0             │
└──────────────────────────────────────────────────┴───────────────────────────┘
```

DISTRIBUTION A. Approved for public release: distribution unlimited.

Capture The Flag

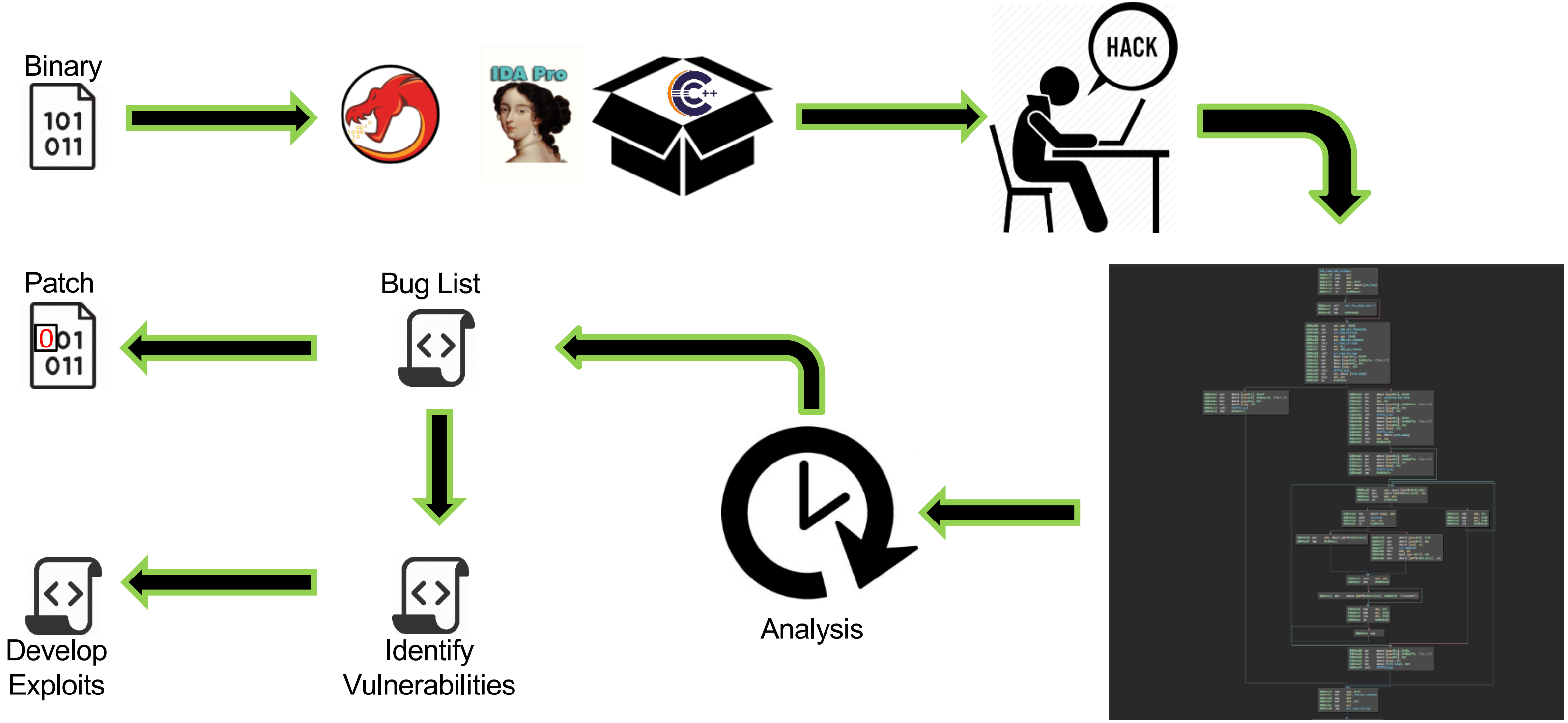
- A Game of Bug Hunting + Exploitation
- Each team analyzes their application binaries
  - Discover vulnerabilities however they want
  - Patch vulnerabilities to protect themselves from attacks (defensive)
  - Exploit vulnerabilities against other teams servers (offensive)
- Points are earned from successful completion of aforementioned actions
- DEF CON CTF is the championship for humans

DEFCON



DISTRIBUTION A. Approved for public release: distribution unlimited.

Capture The Flag



Binary

Patch

Bug List

Analysis

Identify Vulnerabilities

Develop Exploits

DISTRIBUTION A. Approved for public release: distribution unlimited.

NAVSEA WARFARE CENTERS Port Hueneme

DARPA Cyber Grand Challenge

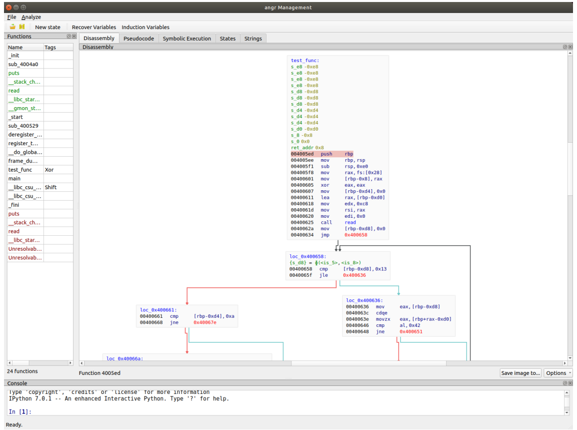
- Leverage Artificial Intelligence (AI) to bug hunt and play CTF.
- Compete against humans (unofficially in the DEF CON CTF)
- Used a test operating system
- 1st Place For All Secure with MAYHEM
- 3rd Place UCSB Shellphish with ANGR

WORLD'S FIRST MACHINE-ONLY HACKING COMPETITION

DISTRIBUTION A. Approved for public release: distribution unlimited.

Free Tools – ANGR

- **Automated intelligent code analysis and fuzzing tool released as free open source**
    - A Framework for analyzing binaries
    - Meant to have components change out over time
    - Developed by UC Santa Barbara for the DARPA Cyber Grand Challenge…
- **Allows you to concolically (statically + dynamically) analyze a binary**
    - No source code required
    - Can disassemble, decompile, and fuzz software
    - Can generate patches (to fix) and exploits (to break)
    - Purely command line but a GUI is being developed



Name	Tags
_init	
sub_4004a0	
puts	
__stack_ch...	
read	
__libc_star...	
__gmon_st...	
_start	
sub_400529	
deregister_...	
register_t...	
__do_globa...	
frame_du...	
test_func	Xor
main	
__libc_csu_...	Shift
__libc_csu_...	
_fini	
puts	
__stack_ch...	
read	
__libc_star...	
Unresolvab...	
Unresolvab...	

```
test_func:
s_e8 -0xe8
s_e8 -0xe8
s_e8 -0xe8
s_e8 -0xe8
s_d8 -0xd8
s_d8 -0xd8
s_d8 -0xd8
s_d4 -0xd4
s_d4 -0xd4
s_d4 -0xd4
s_d0 -0xd0
s_8 -0x8
s_0 0x0
ret_addr 0x8
004005ed  push   rbp
004005ee  mov    rbp, rsp
004005f1  sub    rsp, 0xe0
004005f8  mov    rax, fs:[0x28]
00400601  mov    [rbp-0x8], rax
00400605  xor    eax, eax
00400607  mov    [rbp-0xd4], 0x0
00400611  lea    rax, [rbp-0xd0]
00400618  mov    edx, 0xc8
0040061d  mov    rsi, rax
00400620  mov    edi, 0x0
00400625  call   read
0040062a  mov    [rbp-0xd8], 0x0
00400634  jmp    0x400658
```

```
loc_0x400658:
{s_d8} = ϕ(<is_5>, <is_8>)
00400658  cmp    [rbp-0xd8], 0x13
0040065f  jle    0x400636
```

```
loc_0x400661:
00400661  cmp    [rbp-0xd4], 0xa
00400668  jne    0x40067e
```

```
loc_0x400636:
00400636  mov    eax, [rbp-0xd8]
0040063c  cdqe
0040063e  movzx  eax, [rbp+rax-0xd0]
00400646  cmp    al, 0x42
00400648  jne    0x400651
```

```
loc_0x40066a:
```

24 functions

Function 4005ed

```
Type 'copyright', 'credits' or 'license' for more information
IPython 7.0.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

Ready.

DISTRIBUTION A. Approved for public release: distribution unlimited.

DIU Project Voltron Pilot

- **Defense Innovation Unit (DIU) Project Voltron**
  - Mature the Cyber Reasoning Systems from CGC
  - Work with Government & Industry to pilot
  - Automate bug hunting outside of academics & lab work

DISTRIBUTION A. Approved for public release: distribution unlimited.

Questions?

DISTRIBUTION A. Approved for public release: distribution unlimited.

NAVSEA WARFARE CENTERS Port Hueneme

Backup - Abstract

- Finding software defects and identifying security vulnerabilities in binaries is just as important as developing the software itself. This talk will cover the breadth of bug hunting, reverse engineering tools, DARPA Cyber Grand Challenge (CGC), and briefly address efforts in piloting a tool from the Defense Innovation Unit (DIU) Project Voltron - the follow on to the CGC for intelligently automating the detection and remediation of software bugs.

Backup - Bio

- Socrates Frangis is the Cybersecurity Technical Lead, as a direct report to the NAVSEA Naval Surface Warfare Center Port Hueneme Division Technical Director. His duties at the command have spanned security research, security engineering, penetration testing, software maintenance, and naval combat systems engineering. Academically, he has an undergraduate degree in Computer Science & Applied Mathematics from UCSB, Graduate degree from Naval Postgraduate School in Systems Engineering with a focus on directed energy weapons, and is currently a PhD Student at Naval Postgraduate School in Systems Engineering with a focus on cybersecurity.

DISTRIBUTION A. Approved for public release: distribution unlimited.