

Axel Munoz: CSDN, BAH DarkLabs

Practical Applications of Reverse Engineering using NSA's GHIDRA

README

- \$whoami
- Cybersecurity Landscape
- Reverse Engineering
- Offensive RE
- Defensive RE
- GHIDRA
- Reverse Engineering Walkthrough

\$whoami



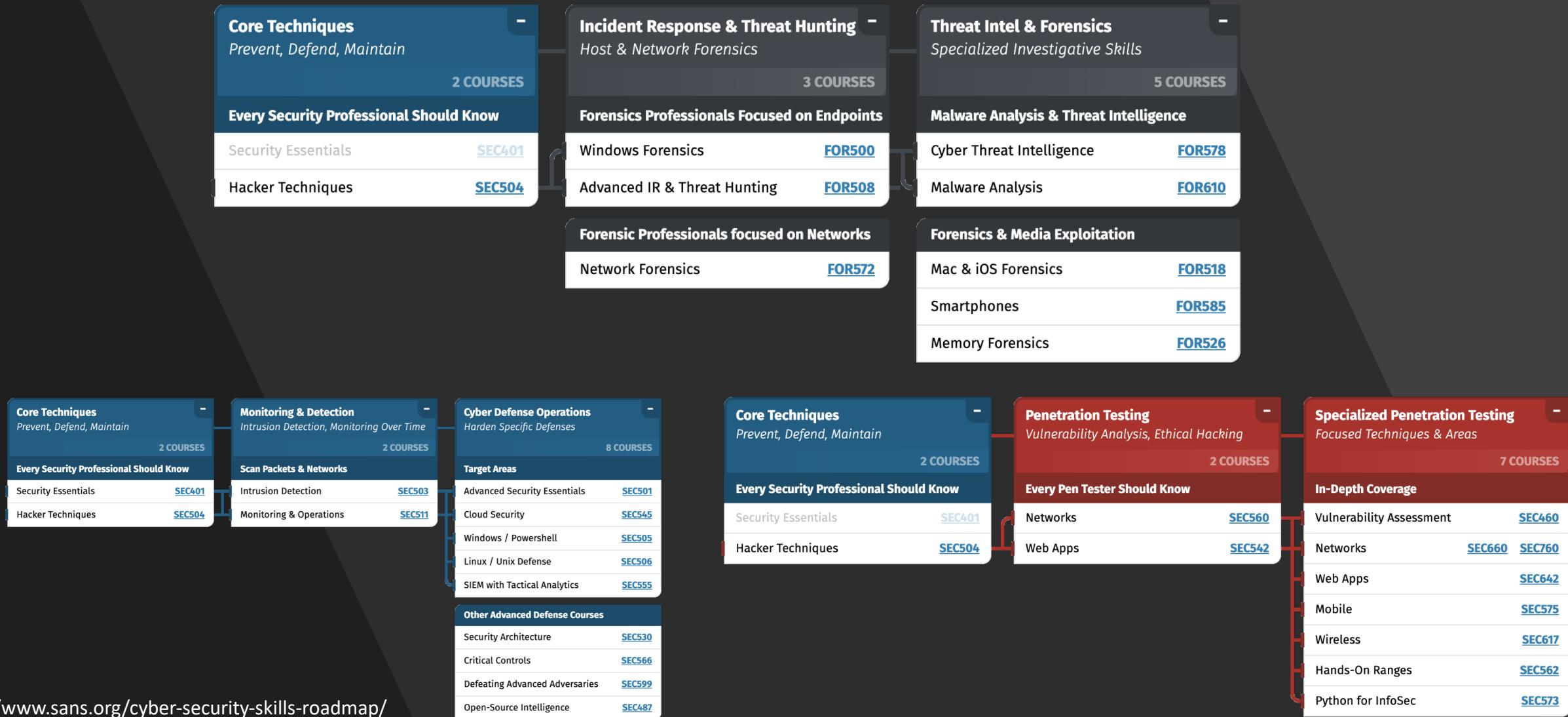
Custom-Designed Major:

- Cryptology and Intelligence Analysis
- Enables students to pursue individualized course of studies not available through existing majors or combination of majors and minors.
- Designed for highly motivated students who have interdisciplinary curiosity and career ambitions.
- The Major:
 - Requires an intensive, but guided, application process
 - Emphasizes self-directed learning
 - Incorporates early and intensive research
 - Culminates in a faculty mentored capstone project

Booz Allen Hamilton:

- Senior Threat Hunter
- DarkLabs
 - Client Hunting
 - Research & Development
- Developing novel Hunt methods

Cybersecurity Landscape



Reverse Engineering

- Machine Language are the 1's and 0's your computer understands
- Even the most trivial code gets compiled into complex, hard-to-understand assembly in order to run
- Malware and security software that doesn't want to be analyzed can fight back
 - Assembly can change during the analysis process
 - Assembly could be "packed" or encrypted
- Malware doesn't want to be analyzed so you can't detect it in the future
- Security software doesn't want to be analyzed so attackers can't find vulnerabilities

Reverse Engineering

Hello, Nerd Night! -> Machine Code

- Python

```
print("Hello, Nerd Night!")
```

- C

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    puts("Hello World!");
    return EXIT_SUCCESS;
}
```

- Javascript

```
console.log("Hello, Nerd Night!");
```

- Fortran

```
PROGRAM HELLO
WRITE (*,100)
STOP
100 FORMAT ('Hello, Nerd Night!' /)
END
```

```
→ RE Examples cat hello_nerd_night.c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    puts("Hello, Nerd Night!");
    return EXIT_SUCCESS;
}
→ RE Examples gcc -Wall hello_nerd_night.c -o hello_nerd_night.o
→ RE Examples ./hello_nerd_night.o
Hello, Nerd Night!
→ RE Examples r2 hello_nerd_night.o
r_config_set: variable 'asm.cmtright' not found
-- Analyze socket connections with the socket plugin: 'radare2 socket://www.foo.com:80'. Use 'w' to send data
[0x100000f50]> aa
[x] Analyze all flags starting with sym. and entry0 (aa)
[0x100000f50]> pdf@main
    ;-- main:
    ;-- section.0.__TEXT.__text:
    ;-- func.100000f50:
    ;-- rip:
(fcn) entry0 40
entry0 (int arg4);
    ; var int local_8h @ rbp-0x8
    ; var int local_4h @ rbp-0x4
    ; arg int arg4 @ rcx
0x100000f50 55      push rbp          ; [00] -r-x section size 40 named 0.__TEXT.__text
0x100000f51 4889e5    rbp = rsp
0x100000f54 4883c10   rsp -= 0x10
0x100000f58 c745fc000000. dword [local_4h] = 0
0x100000f5f 488d3d340000. rdi = str.Hello_Nerd_Night ; section.3.__TEXT.__cstring ; 0x100000f9a ; "Hello, Nerd Night!"
0x100000f66 e80d000000  sym.imp.puts ()           ; int puts(const char *s)
0x100000f6b 31c9      ecx = 0
0x100000f6d 8945f8    dword [local_8h] = eax
0x100000f70 89c8      eax = ecx
0x100000f72 4883c410  rsp += 0x10
0x100000f76 5d
0x100000f77 c3        return
[0x100000f50]> █
```

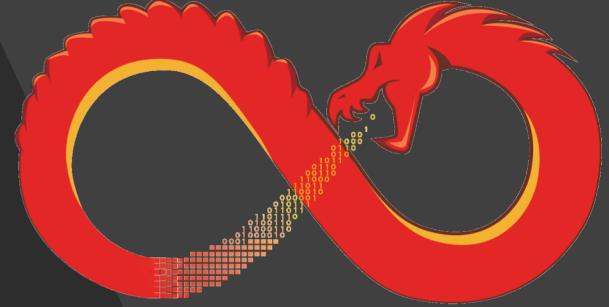
Offensive RE

- Just because it's offensive, doesn't mean it's illegal or even unethical
- Company pay analysts to RE software to discover vulnerabilities
- Analysts RE software looking for bugs for bug bounty (\$\$!)
- Requires background in exploitation development
 - How can I abuse this bug in order to take control?

Defensive RE

- Very different than Offensive RE
- Look for unique signatures for future detection
- 5 W's+H:
 - **How** did this malware arrive in our environment?
 - **Who** sent this malware?
 - **What** is the objective for the malware? (Exfiltration, Botnet, C2 connection, System damage)
 - **When** did the malware arrive? **When** was it made?
 - **Where** did this malware come from? (Infrastructure, country, etc.)
 - **Why** was this malware targeted towards us? (Financial gain, Intellectual Property, Hacktivism)

GHIDRA



GHIDRA

- National Security Agency created a disassembler and decompiler
 - Makes it easier to read and understand the code
 - Still impossible to retrieve original code, so still difficult
- Completely free and Open Source
 - Anyone can look at the source code for the program and make improvements
- The only (decent) free disassembler for Mac, Linux, and Windows
 - Supports 25+ instruction sets (32bit, 64bit, ARM, CHIP-8, etc.)

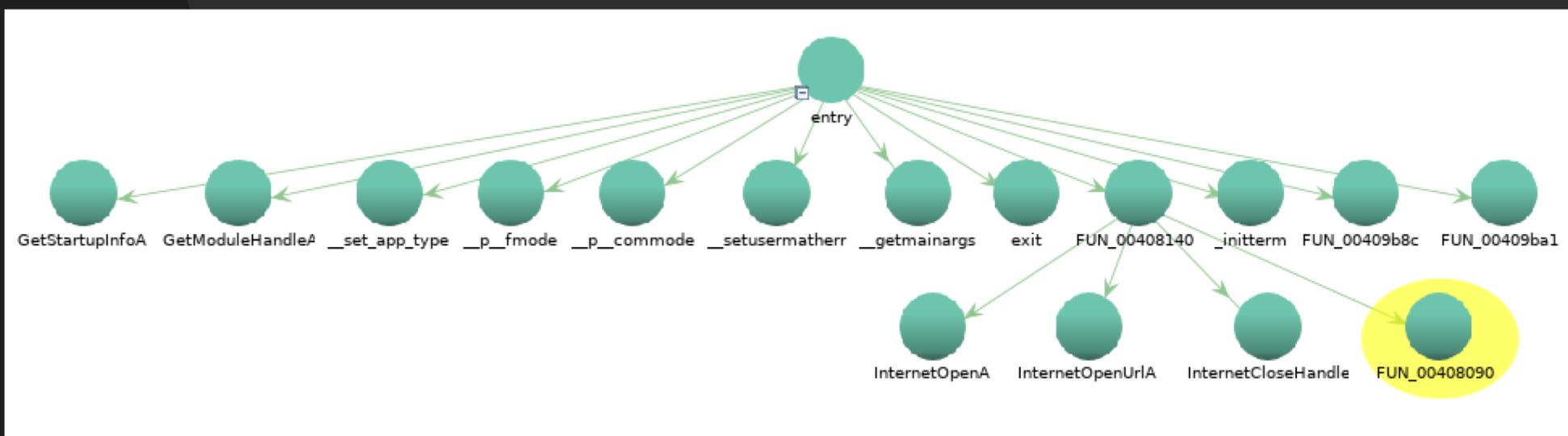
Reverse Engineering Walkthrough w/ GHIDRA

- WannaCry
 - Ransomware attack ~ May 2017
 - 200k computers across 150 countries
 - Billions of dollars in damage
 - Attributed to DPRK actors
 - Used EternalBlue Exploit developed by NSA (ironic)
- Infamous kill switch
 - Hardcoded domain to check if it should proceed
 - Smart move, bad execution
 - Marcus Hutchins first discovered it
 - Let's find it!



Walkthrough (cont.)

- Function Call Graphs (a feature of GHIDRA) shows the parent-child relationship of function calls
- Highlighted function starts the actual malicious code (encryption of files)
- **How** does the code decide whether or not to call this function?



Walkthrough (cont.)

- Interesting that there are functions referring to the internet so early in the code execution... Let's look into that.
- Section One shows a URL
- Section Two shows the program connecting to that URL
- Section Three shows that the program will quit without malicious behavior if the connection fails
- But wait... That isn't a valid URL...
 - s_http://www.iuqerfsodp9ifjaposdfj_004313d0

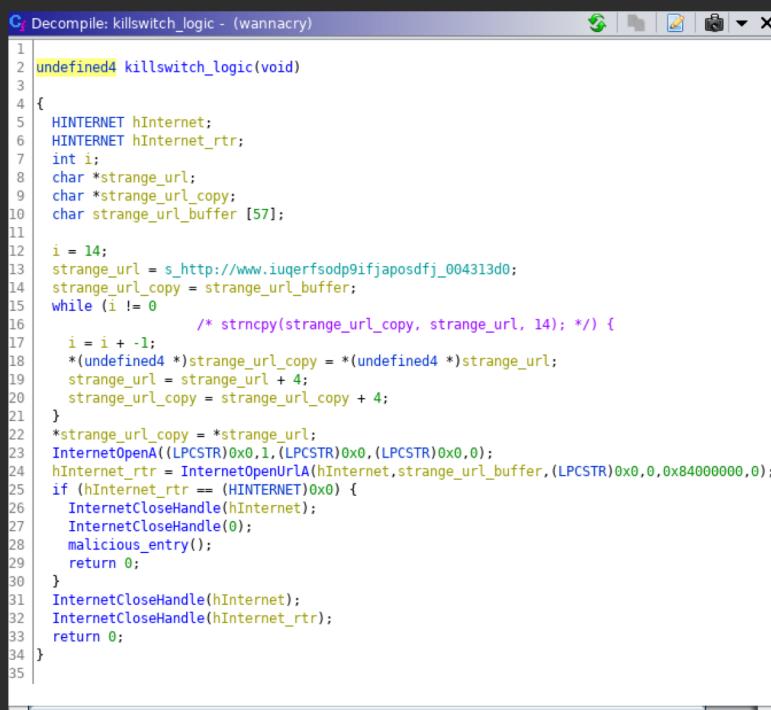
[http://www.\[.\]iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea\[.\]com](http://www.[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea[.]com)

```
undefined FUN_00408140()
AL:1 <RETURN>
Stack[-0x1]:1 local_1
Stack[-0x3]:2 local_3
Stack[-0x7]:4 local_7
Stack[-0xb]:4 local_b
Stack[-0xf]:4 local_f
Stack[-0x13]:4 local_l3
Stack[-0x17]:4 local_17
Stack[-0x50]:1 local_50
XREF[1]: 00408177(W)
XREF[1]: 0040816c(W)
XREF[1]: 00408168(W)
XREF[1]: 00408164(W)
XREF[1]: 00408160(W)
XREF[1]: 0040815c(W)
XREF[1]: 00408158(W)
XREF[1]: 0040814f(*)
XREF[1]: entry:00409b45(c)

00408140 83 ec 50 SUB ESP,0x50
00408143 56 PUSH ESI
00408144 57 PUSH EDI
00408145 b9 0e 00 MOV ECX,0xe
0040814a be d0 13 MOV ESI,s_http://www.iuqerfsodp9ifjaposdfj_004313d0 = "http://www.iuqerfsodp9ifjapos...
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.iuqerfsodp9ifjaposdfj... = http://www.iuqerfsodp9ifjapos...
00408157 a4 MOVSB ES:EDI,ESI=>s_http://www.iuqerfsodp9ifjaposdfj... = http://www.iuqerfsodp9ifjapos...
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
0040817b ff 15 34 CALL dword ptr [->WININET.DLL::InternetOpenA]
al 40 00
00408181 6a 00 PUSH 0x0
00408183 68 00 00 PUSH 0x84000000
00408188 6a 00 PUSH 0x0
0040818a 8d 4c 24 14 LEA ECX,[ESP + 0x14]
0040818e 8b f0 MOV ESI,EAX
00408190 6a 00 PUSH 0x0
00408192 51 PUSH ECX
00408193 56 PUSH ESI
00408194 ff 15 38 CALL dword ptr [->WININET.DLL::InternetOpenUrlA]
al 40 00
0040819a 8b f8 MOV EDI,EAX
0040819c 56 PUSH ESI
0040819d 8b 35 3c MOV ESI,dword ptr [->WININET.DLL::InternetCloseHandle... = 0000a7b2
al 40 00
004081a3 85 ff TEST EDI,EDI
004081a5 75 15 JNZ LAB_004081bc
004081a7 ff d6 CALL ESI=>WININET.DLL::InternetCloseHandle
004081a9 6a 00 PUSH 0x0
004081ab ff d6 CALL ESI=>WININET.DLL::InternetCloseHandle
004081ad e8 de fe CALL FUN_00408090
ff ff
004081b2 5f POP EDI
004081b3 33 c0 XOR EAX,EAX
004081b5 5e POP ESI
004081b6 83 c4 50 ADD ESP,0x50
004081b9 c2 10 00 RET 0x10
undefined FUN_00408090()
```

Walkthrough (cont.)

- GHIDRA has a built-in decompiler which has a very naïve code converter
- It's still difficult to understand
- Unlike other software, GHIDRA makes it easy to decode



The screenshot shows the GHIDRA decompiler interface with two panes. The left pane displays the decompiled assembly code for the `killswitch_logic` function. The right pane shows the original raw assembly code for the `FUN_00408140` function.

```
C:\Decompile: killswitch_logic - (wannacry)
1 undefined4 killswitch_logic(void)
2
3 {
4     HINTERNET hInternet;
5     HINTERNET hInternet_rtr;
6     int i;
7     char *strange_url;
8     char *strange_url_copy;
9     char strange_url_buffer [57];
10
11    i = 14;
12    strange_url = s_http://www.iuqerfsodp9ifjaposdfj_004313d0;
13    strange_url_copy = strange_url_buffer;
14    while (i != 0
15        /* strncpy(strange_url_copy, strange_url, 14); */ {
16        i = i + -1;
17        *(undefined4 *)strange_url_copy = *(undefined4 *)strange_url;
18        strange_url = strange_url + 4;
19        strange_url_copy = strange_url_copy + 4;
20    }
21    *strange_url_copy = *strange_url;
22    InternetOpenA((LPCSTR)0x0,1,(LPCSTR)0x0,(LPCSTR)0x0,0);
23    hInternet_rtr = InternetOpenUrlA(hInternet,strange_url_buffer,(LPCSTR)0x0,0,0x84000000,0);
24    if (hInternet_rtr == (HINTERNET)0x0) {
25        InternetCloseHandle(hInternet);
26        InternetCloseHandle(0);
27        malicious_entry();
28        return 0;
29    }
30    InternetCloseHandle(hInternet);
31    InternetCloseHandle(hInternet_rtr);
32    return 0;
33}
34}

C:\Decompile: FUN_00408140 - (wannacry)
1 undefined4 FUN_00408140(void)
2
3 {
4     undefined4 uVar1;
5     int iVar2;
6     undefined4 *puVar3;
7     undefined4 *puVar4;
8     undefined4 uStack100;
9     undefined4 uStack96;
10    undefined4 uStack92;
11    undefined4 local_50 [14];
12    undefined4 local_17;
13    undefined4 local_13;
14    undefined4 local_f;
15    undefined4 local_b;
16    undefined4 local_7;
17    undefined2 local_3;
18    undefined local_1;
19
20    iVar2 = 0xe;
21    puVar3 = (undefined *)s_http://www.iuqerfsodp9ifjaposdfj_004313d0;
22    puVar4 = local_50;
23    while (iVar2 != 0) {
24        iVar2 = iVar2 + -1;
25        *puVar4 = *puVar3;
26        puVar3 = puVar3 + 1;
27        puVar4 = puVar4 + 1;
28    }
29    *(undefined *)puVar4 = *(undefined *)puVar3;
30    local_17 = 0;
31    local_13 = 0;
32    local_f = 0;
33    local_b = 0;
34    local_7 = 0;
35    local_3 = 0;
36    uStack92 = 0;
37    uStack96 = 0;
38    uStack100 = 0;
39    local_1 = 0;
40    uVar1 = InternetOpenA(0,1);
41    iVar2 = InternetOpenUrlA(uVar1,&uStack100,0,0,0x84000000,0);
42    if (iVar2 == 0) {
43        InternetCloseHandle(uVar1);
44        InternetCloseHandle(0);
45        FUN_00408090();
46        return 0;
47    }
48    InternetCloseHandle(uVar1);
49    InternetCloseHandle(iVar2);
50    return 0;
51}
52}
```



The screenshot shows the GHIDRA decompiler interface with two panes. The left pane displays the decompiled assembly code for the `FUN_00408140` function. The right pane shows the original raw assembly code for the `FUN_00408140` function.

```
C:\Decompile: FUN_00408140 - (wannacry)
1 undefined4 FUN_00408140(void)
2
3 {
4     undefined4 uVar1;
5     int iVar2;
6     undefined4 *puVar3;
7     undefined4 *puVar4;
8     undefined4 uStack100;
9     undefined4 uStack96;
10    undefined4 uStack92;
11    undefined4 local_50 [14];
12    undefined4 local_17;
13    undefined4 local_13;
14    undefined4 local_f;
15    undefined4 local_b;
16    undefined4 local_7;
17    undefined2 local_3;
18    undefined local_1;
19
20    iVar2 = 0xe;
21    puVar3 = (undefined *)s_http://www.iuqerfsodp9ifjaposdfj_004313d0;
22    puVar4 = local_50;
23    while (iVar2 != 0) {
24        iVar2 = iVar2 + -1;
25        *puVar4 = *puVar3;
26        puVar3 = puVar3 + 1;
27        puVar4 = puVar4 + 1;
28    }
29    *(undefined *)puVar4 = *(undefined *)puVar3;
30    local_17 = 0;
31    local_13 = 0;
32    local_f = 0;
33    local_b = 0;
34    local_7 = 0;
35    local_3 = 0;
36    uStack92 = 0;
37    uStack96 = 0;
38    uStack100 = 0;
39    local_1 = 0;
40    uVar1 = InternetOpenA(0,1);
41    iVar2 = InternetOpenUrlA(uVar1,&uStack100,0,0,0x84000000,0);
42    if (iVar2 == 0) {
43        InternetCloseHandle(uVar1);
44        InternetCloseHandle(0);
45        FUN_00408090();
46        return 0;
47    }
48    InternetCloseHandle(uVar1);
49    InternetCloseHandle(iVar2);
50    return 0;
51}
52}
```

Walkthrough (cont.)

- Now we know there is a killswitch:
 - If the software can reach that odd domain it won't encrypt the files
- Next steps are to:
 - Contact authorities to register domain
 - Identify any hosts in your own network who may have reached out to that domain
 - If there are any, quarantine them

How to Get Started

- Computer Science degree
 - CST/IT/IS
- Tier-One Analyst
- Knowledge of Threat Intelligence
 - Understand the geo-political landscape of who might target your organization/clients
- Exploitation Development
 - How can I abuse this software to gain control of the computer?



Questions?