

RSA® Conference 2019

San Francisco | March 4–8 | Moscone Center



BETTER.

SESSION ID: PNG-T09

GET YOUR FREE NSA REVERSE ENGINEERING TOOL



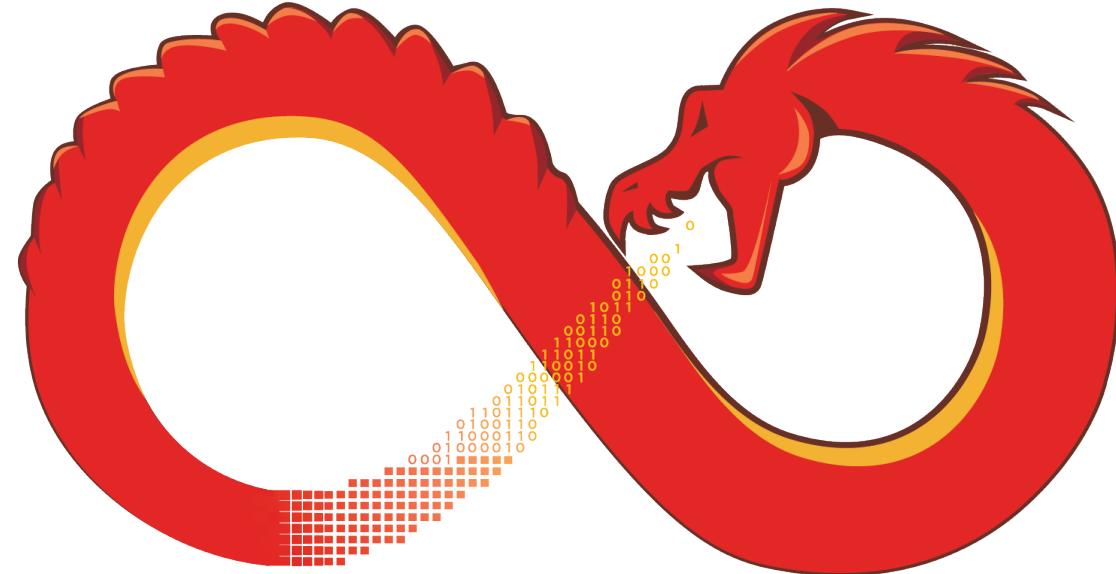
Rob Joyce

Senior Advisor for Cybersecurity
National Security Agency



#RSAC

Introducing:

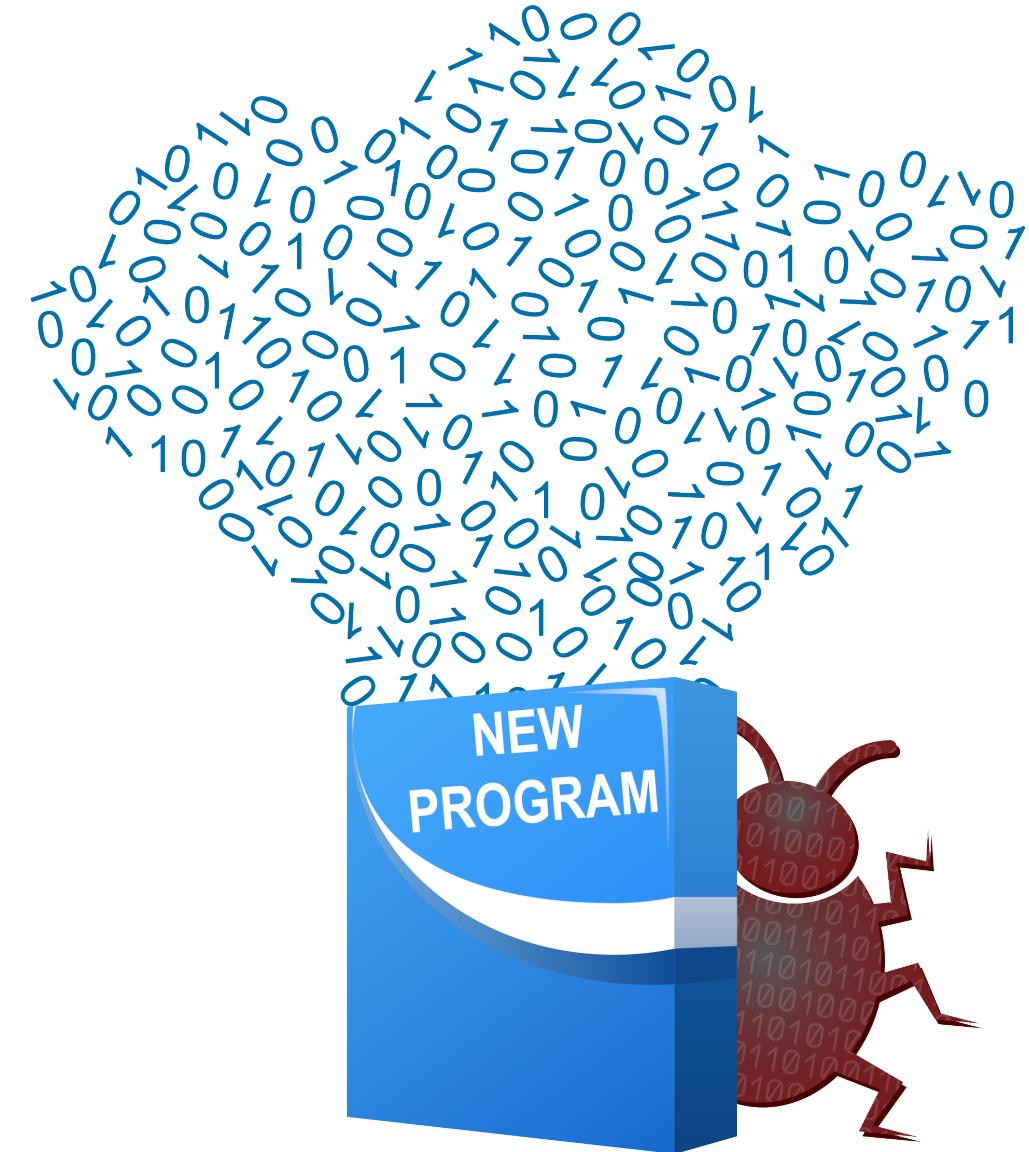


GHIDRA

**SOFTWARE REVERSE
ENGINEERING TOOL SUITE**

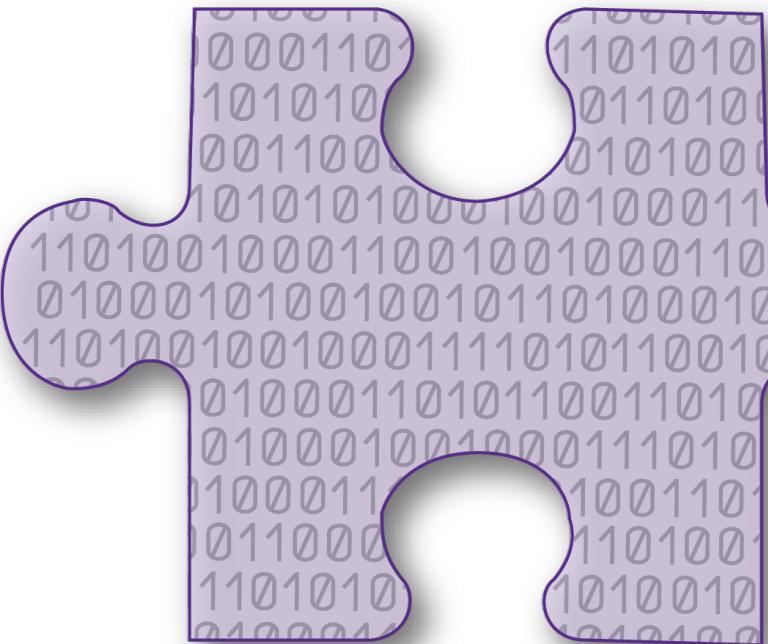


Ghidra Purpose - What's in Your Binary?



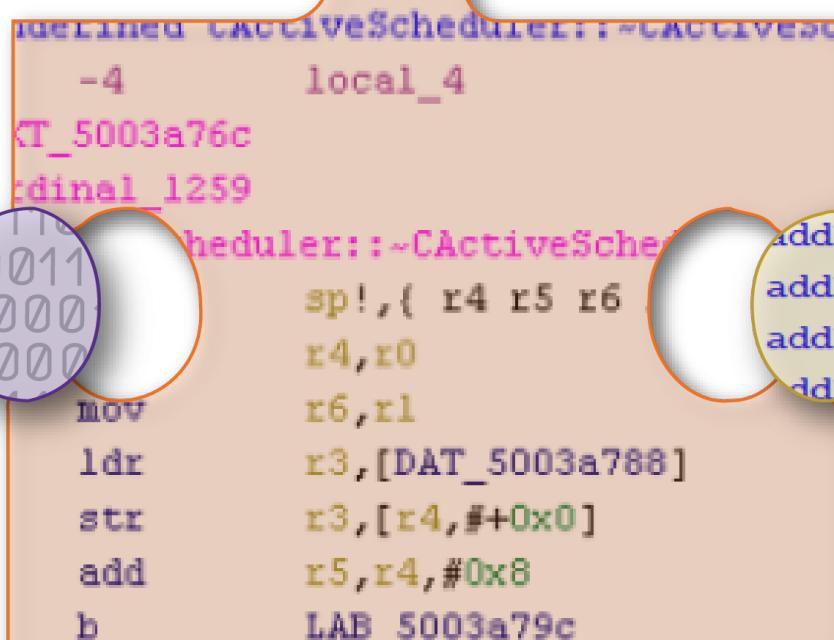
Assembling the Puzzle

RAW BINARY



01001100
00011010
10101010
00110000
01010000
1010101000100011
110100100011001001000110100011
010001001001011010001001000
1101001000111101011001001000
0100011010110011010
010001001000111010
010001101000111010
011000110100011010
11010100
10101010
01000010

ANNOTATED ASSEMBLY



```
mainline CActiveScheduler::~CActiveScheduler()
{
    -4           local_4
    KT_5003a76c
    cardinal_1259
    CActiveScheduler::~CActiveScheduler()
    {
        sp!, { r4 r5 r6
        r4, r0
        r6, r1
        mov    r3, [DAT_5003a788]
        str   r3, [r4, #+0x0]
        add   r5, r4, #0x8
        b     LAB_5003a79c
    }
}
```

5006C070

C CODE



```
list = new CActiveScheduler();
list->addPerson("Lord Quartermain");
list->addPerson("Lady Tottington");
list->addPerson("Were Rabbit");
list->addPerson("Rabbit");
list->addPerson("Gromit");
list->addPerson("Wallace");
```



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**





Why Did We Release Ghidra?

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

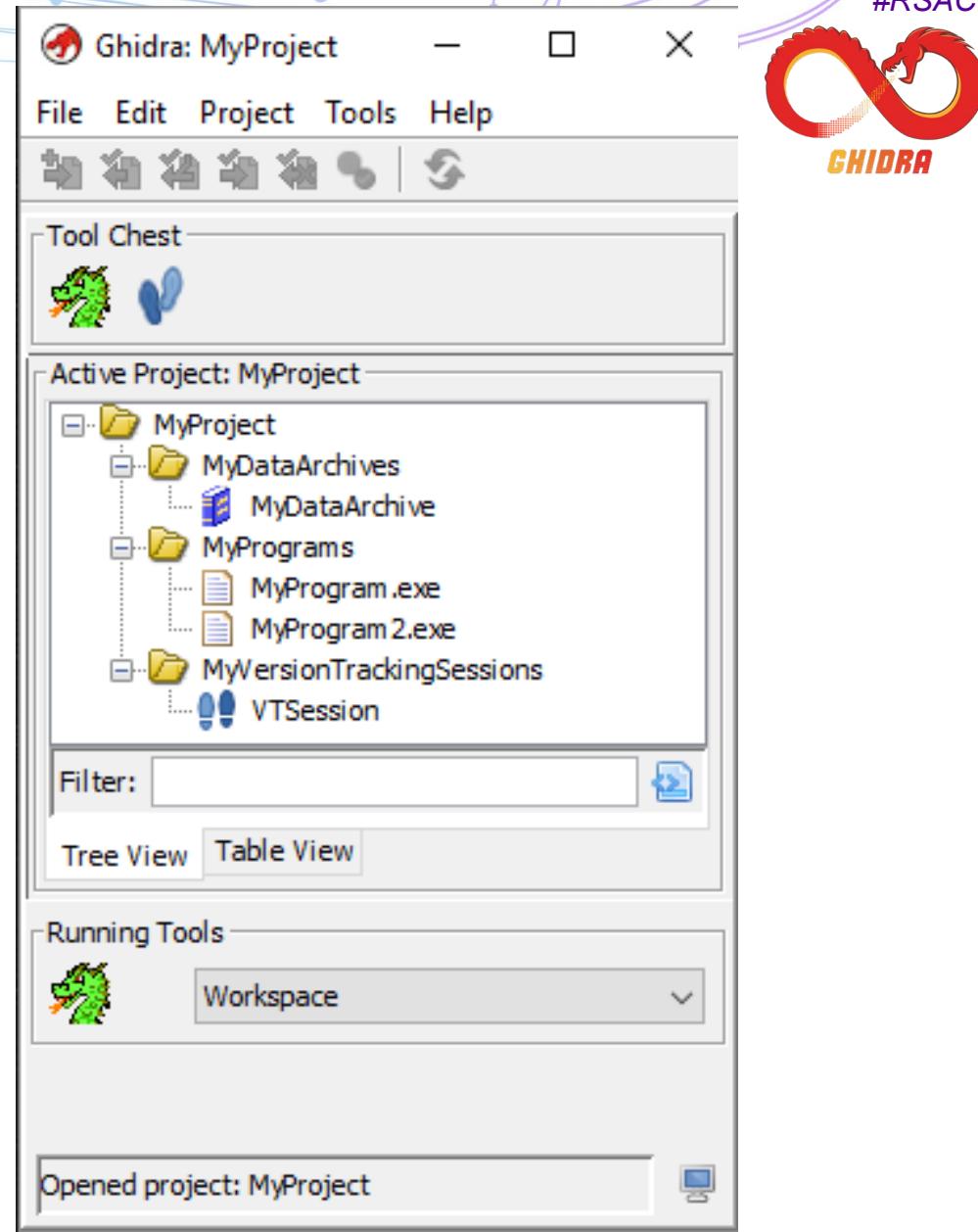


A product of NSA's Research Organization



Get Started in the Project Manager

- Open a new project
- Drag your binary into the project





Configurable Environment

The screenshot displays the CodeBrowser interface with several windows open:

- Program Trees**: Shows the project structure with files like MyProgram.exe, Headers, .textbss, .text, and .rdata.
- Listing: MyProgram.exe - (6 addresses selected)**: Displays assembly code for the selected file. The assembly code for the function at address 00411d04 is shown:

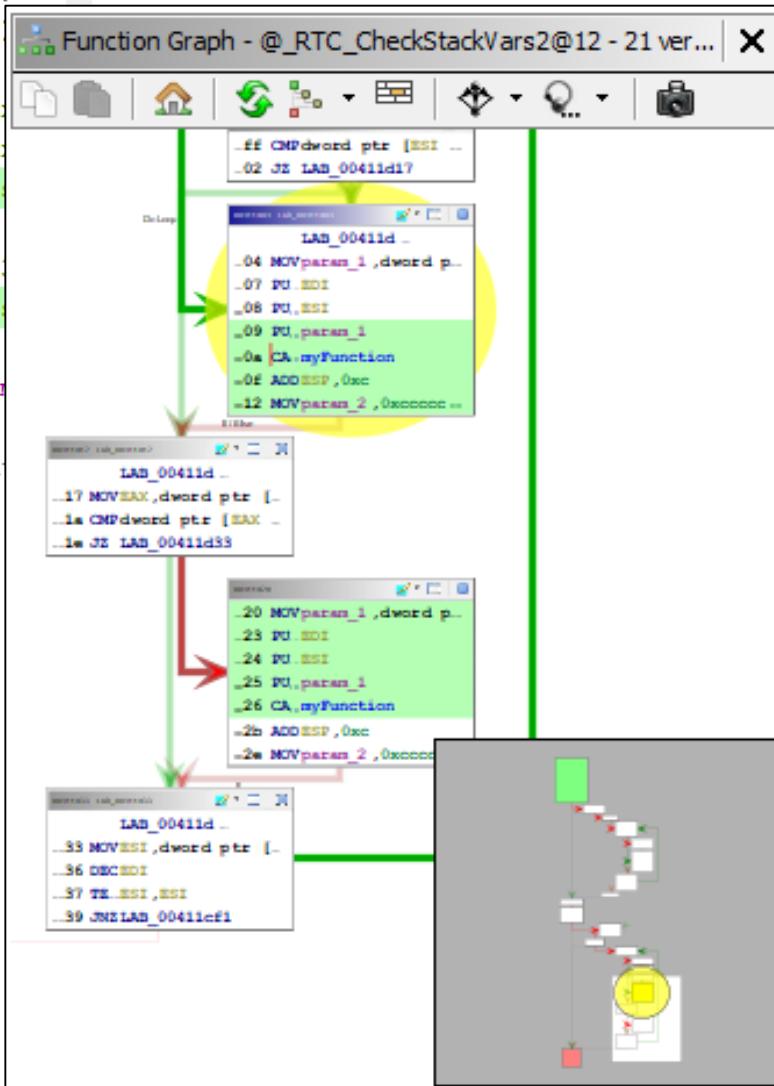
```
00411d04    MOV     param_1
00411d07    PUSH    EDI
00411d08    PUSH    ESI
00411d09    PUSH    param_2
00411d0a    CALL    myFunction
00411d0f    ADD    ESP, 0x4
00411d12    MOV     param_3
...
LAB_00411d17
00411d17    MOV     EAX
00411d1a    CMP    dword ptr [ESI + 0x4]
00411d1e    JZ     LAB_00411d33
00411d20    MOV     param_1
00411d23    PUSH    EDI
00411d24    PUSH    ESI
00411d25    PUSH    param_2
00411d26    CALL    myFunction
```

- Function Graph - @_RTC_CheckStackVars2@12 - 21 vert.**: A graph visualization of the function's control flow. It shows nodes representing assembly code blocks and their connections. A yellow oval highlights a specific node, and a green arrow points from it to another node.
- Decompile: @_RTC_CheckStackVars2@12**: Shows the decompiled C-like pseudocode for the function. The code iterates through a stack of pointers, calling a function for each one until it finds a null pointer.



Many Views All in Sync

```
Cf Decompile: myFunction2 - (MyPro... □ | □ | □ | □ | X
32     iVar3 = iVar3 + 1;
33 } while (piVar2 != (int *)0x0
34 if (param_3 != (int *)0x0
35 do {
36     if (((*param_3 != -0
37         (param_3[6] != -0
38     myFunction(local_re
39 }
40     if (*int *) (param_3[
41     myFunction(local_re
42 }
43     param_3 = (int *)param
44     iVar3 = iVar3 + -1;
45 } while (param_3 != (in
46 }
47 }
48 return;
49 }
```



Bytes: MyProgram.exe □ | □ | □ | X

Addresses	Hex
00411ca0	0b 8b 44 38 04 03 c1 39 14 18 74 19 8b 4e 04 8b
00411cb0	54 0f 08 8b 45 04 52 50 e8 29 f4 ff ff 83 c4 08
00411cc0	ba cc cc cc 8b 45 fc 40 83 c7 0c 89 45 fc 3b
00411cd0	06 7c bd eb 05 ba cc cc cc 8b 75 08 33 ff 8b
00411ce0	c6 85 f6 74 56 8b 40 04 47 85 c0 75 f8 85 f6 74
00411cf0	4a 39 16 75 0f 39 56 14 75 0a 39 56 18 75 05 39
00411d00	56 1c 74 13 8b 4d 04 57 56 51 8 ea f4 ff ff 83
00411d10	c4 0c ba cc cc cc 8b 46 0c 39 54 30 fc 74 13
00411d20	51 e8 ce f4 ff ff 83 c4 0c ba cc
00411d30	04 4f 85 f6 75 b6 5f 5e 5b 8b e5
00411d40	cc
00411d50	cc
00411d60	cc cc cc cc cc ff 25 dc 92 41 00
00411d70	00 ff 25 d4 92 41 00 ff 25 d0 92
8e	Offset: 00000000 Insertion: 00411d00

Listing: MyProgram.exe - (25 addres... □ | □ | X

*MyProgram.exe x

```
PUSH ESI
PUSH param_1
CALL myFunction
ADD ESP, 0xc
MOV param_2, 0xffffffff

LAB_00411d17
MOV EAX, dword ptr [ESI + 0xc]
CMP dword ptr [EAX + ESI*0x1
JZ LAB_00411d33
MOV param_1, dword ptr [EBP +
PUSH EDI
PUSH ESI
PUSH param_1
CALL myFunction
ADD ESP, 0xc
```



Multi-User Analysis and Collaboration

- Shared Repository
- Reverse Engineering
- Version Control
- Fine Grained Merge
- Speeds up analysis
- Share Knowledge





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

ldr 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 (((lVar1 != 0) && ((* (byte *) (lVar1 + 0
(* (void **) (lVar1 + 8)) != (void *) 0x0)))
arrayDispose (* (void **) (lVar1 + 8));
* (undefined8 *) (lVar1 + 8) = * (undefined
}

DAT_006df630 = * (undefined8 *) (puParm1 - 0
DAT_006df638 = * (undefined8 *) (puParm1 - 0
DAT_006dc7c8 = puParm1[0x10];
DAT_006dde7c = puParm1[0x11];
```



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->pipestatus;
}
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
?0

```

ldr w9, [x8] => _structDef

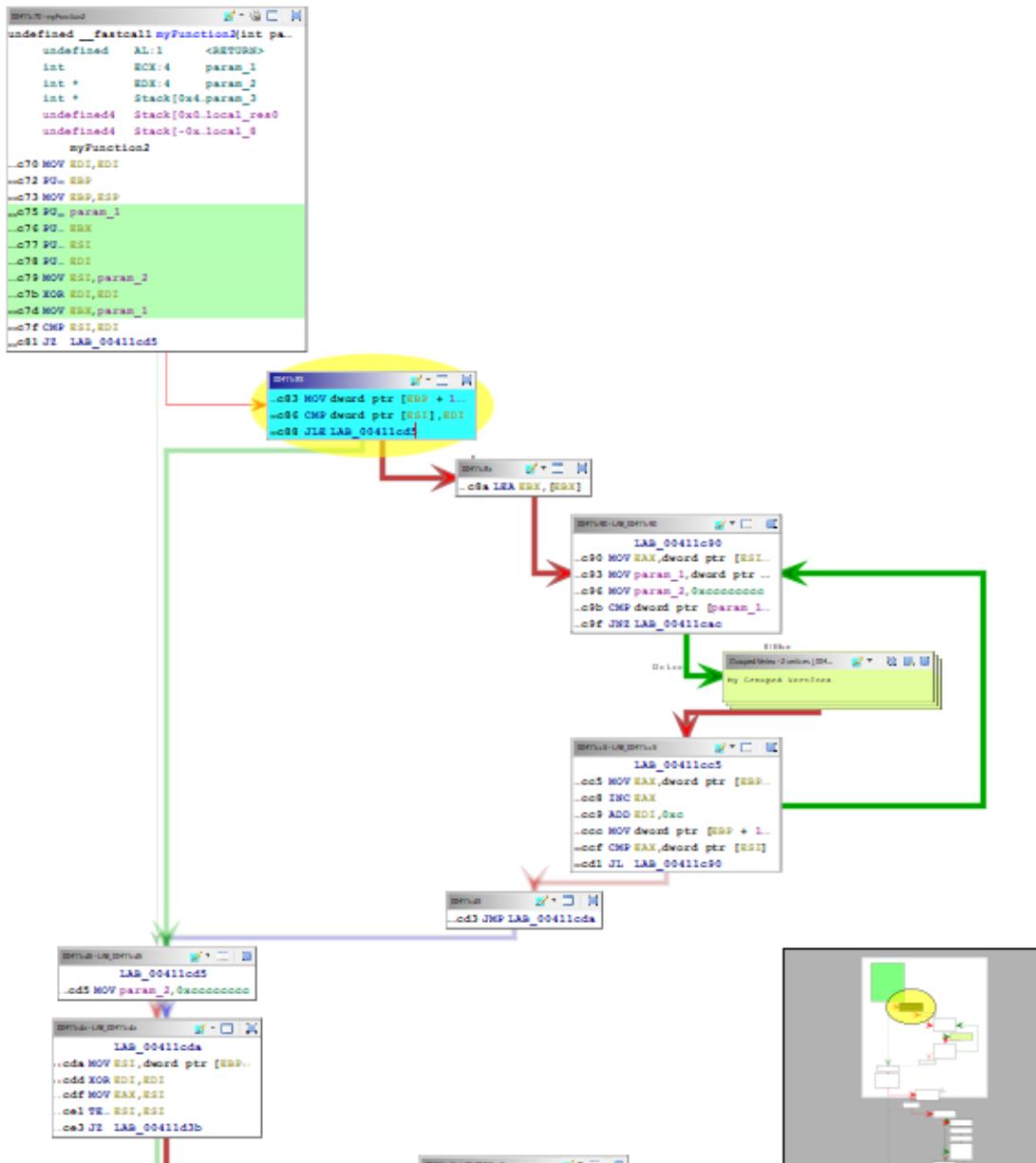
```

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
?0

```



Function Graphs

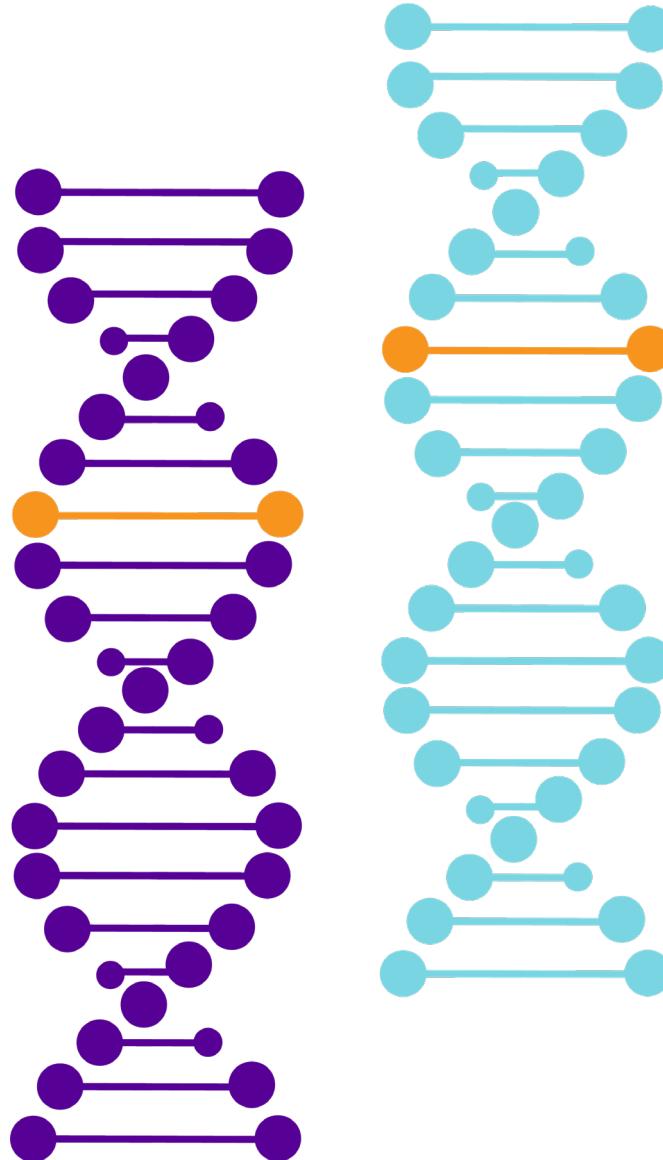


Annotated Differences

0041185f	??	CCh	0041185f	??	CCh
		***** * FUNCTION *****			***** * This function adds a person *****
undefined	_cdecl	FUN_00411	void	_cdecl	addPerson(Person
undefined	AL:1	<RETURN>	Person **	<VOID>	<RETURN>
void * *	Stack[0x4..param_1		char *	Stack[0x8..name	
char *	Stack[0x8..param_2		Person *	EAX:4	person
undefined4	Stack[-0x...local_c		undefined4	Stack[-0x...local_c	
undefined4	Stack[-0x...local_d8		Person *	Stack[-0x...local_d8	
undefined1	Stack[-0x...local_dc		undefined1	Stack[-0x...local_dc	
		FUN_00411860			addPerson
00411860	PUSH	EBP	00411860	PUSH	EBP
00411861	MOV	EBP,ESP	00411861	MOV	EBP,ESP

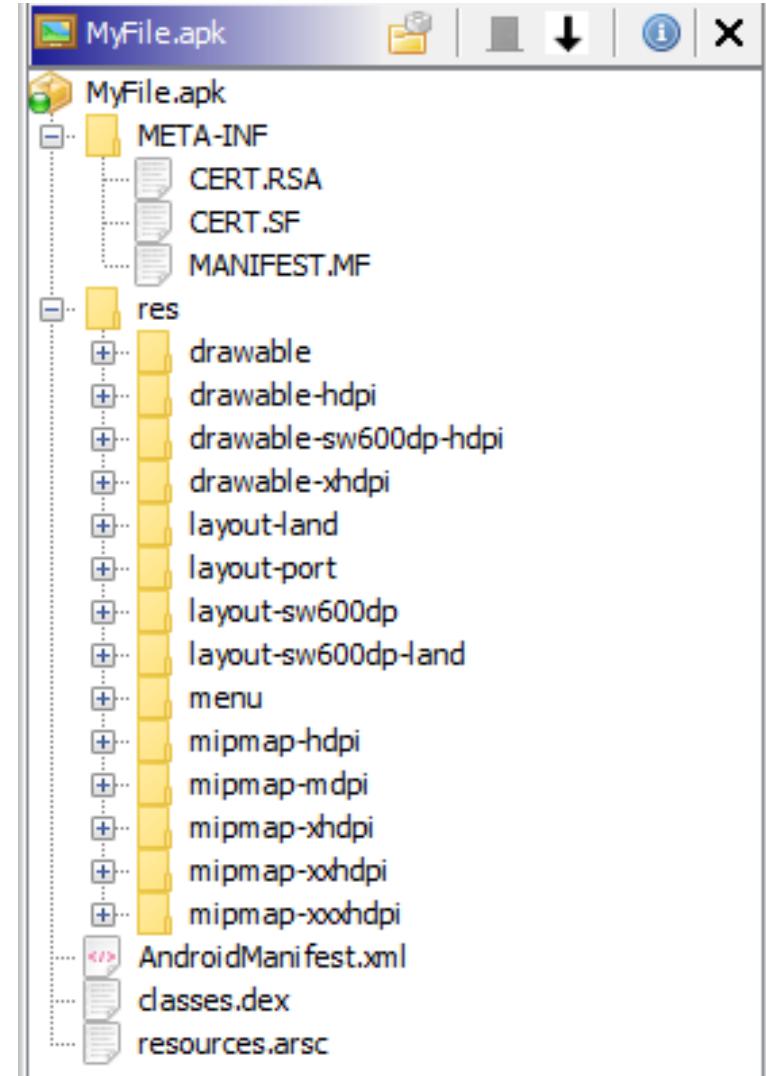
Version Tracking

- Matches functions and data from one version to another
- Multiple algorithms for finding matches
- Easily port annotations and analysis from one version to another



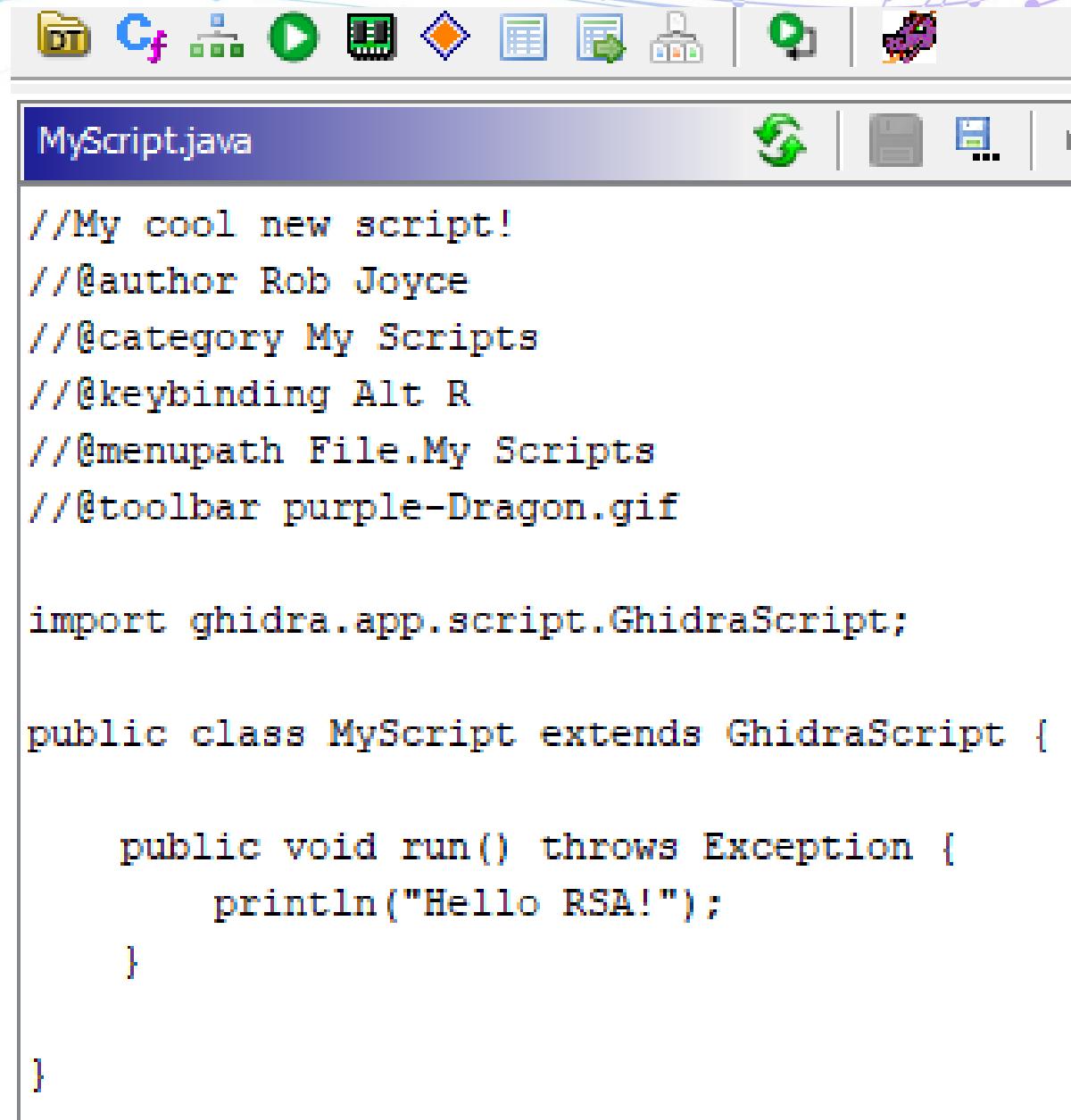
File System

- Viewing/extracting/importing nested components
- Support wide range for formats: tar, zip, gzip, iso9660, apk, etc.



Powerful Scripting

- Extends Ghidra
- Tightly integrated



```
//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 {

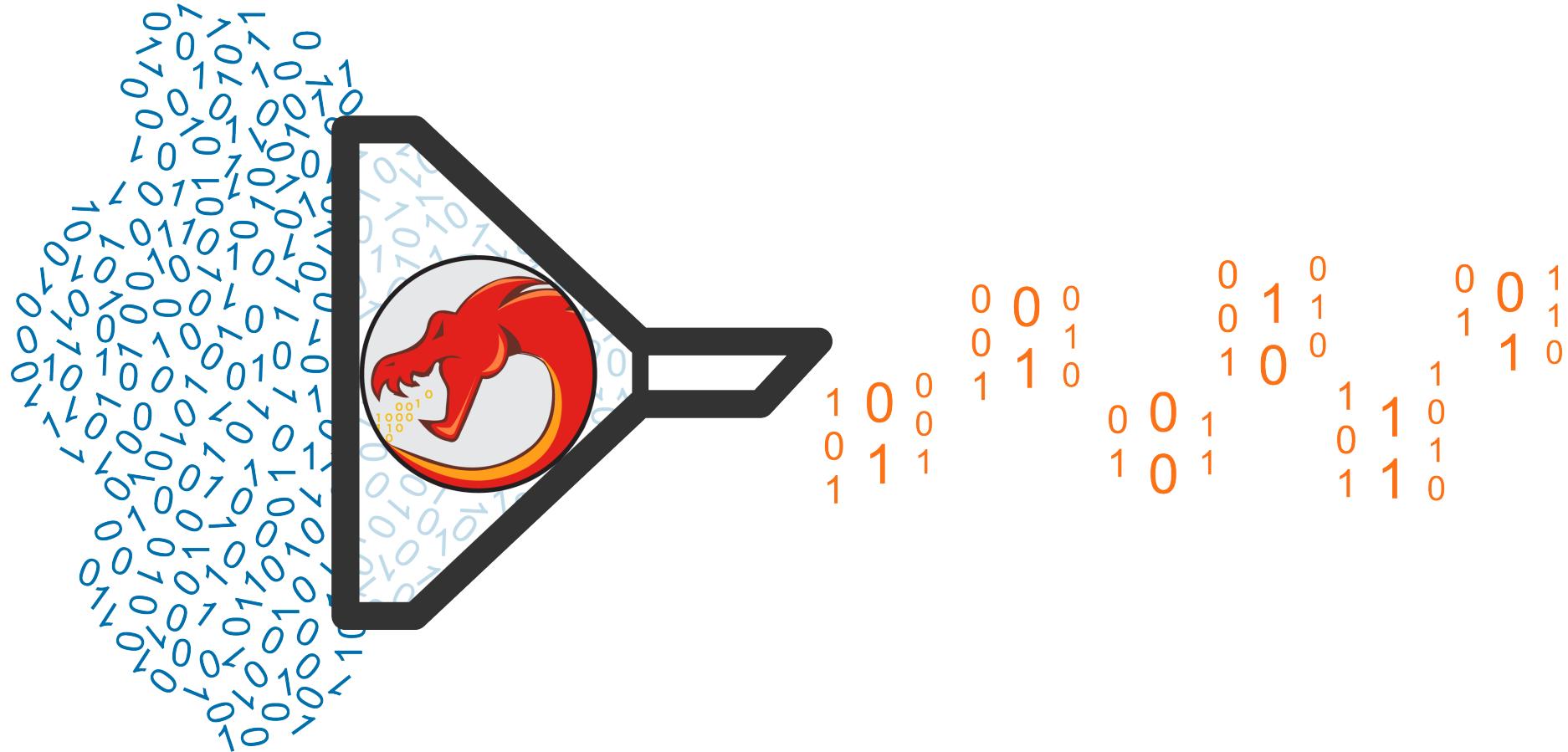
    public void run() throws Exception {
        println("Hello RSA!");
    }

}
```



Automating Analysis

- Batch run Ghidra scripts without the GUI





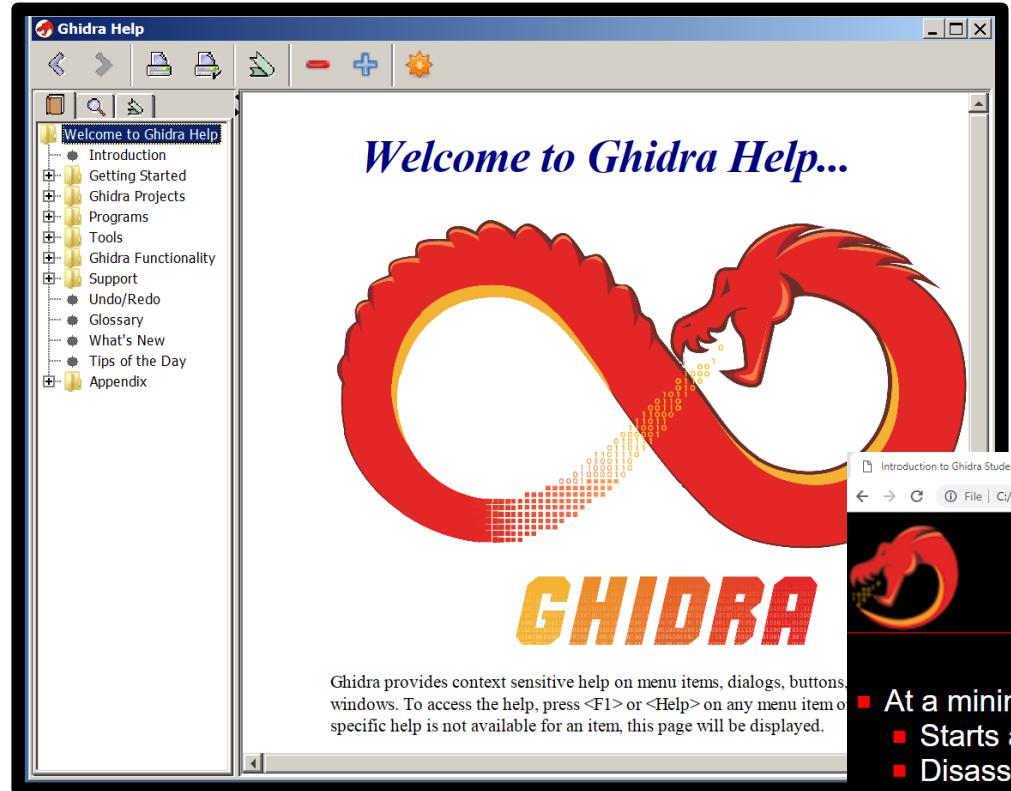
And More Features Including:

Entropy Overview
Data Type Editor
Navigation
Byte Snapshots
Checksums
Memory Viewer
Data Type Archives
Python Support
Symbol Tree
Symbol Table
Extensive Searching
Comments
Searching Bytes
Searching Strings
Filtering





Learning Ghidra



What happens during Auto-Analysis?

- At a minimum:
 - Starts at Entry Points
 - Disassembles by following flows
 - Creates functions at called locations
 - Creates Cross References
- Depending on what options you have selected, Auto-Analysis will do other things as well

Auto-Analysis

- Useful to take a layered approach by running analysis with basic options first and run one-shots later
- Focus on options that will get function starts and data first
- Turn off stack analysis if you have a large binary, can run it individually by function
- Learn how to recognize and fix problems
- Many useful scripts for this



What's Next For Us?

Emulator

Improved
Analysis

Integrated Debugger



What's Next for You?



- Get the software:

www.nsa.gov/ghidra

- Talk to experts at RSA:

NSA Booth, 1753 South Hall



What's in Your Binary?



www.nsa.gov/ghidra

