RSAConference2019
San Francisco | March 4–8 | Moscone Center

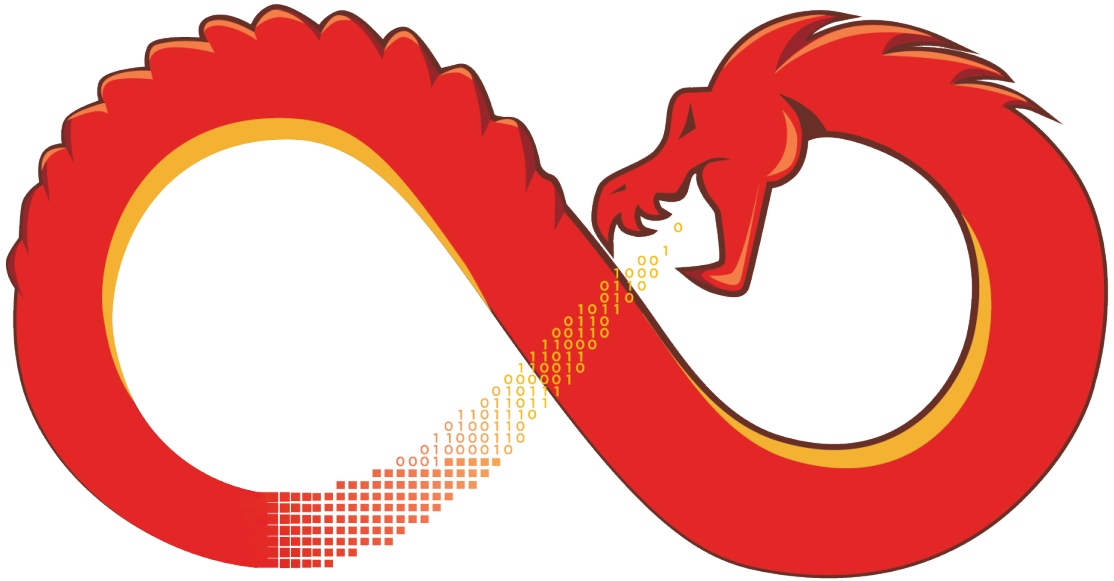
BETTER.

SESSION ID: PNG-T09

GET YOUR FREE NSA REVERSE ENGINEERING TOOL

Rob Joyce
Senior Advisor for Cybersecurity
National Security Agency

Introducing:

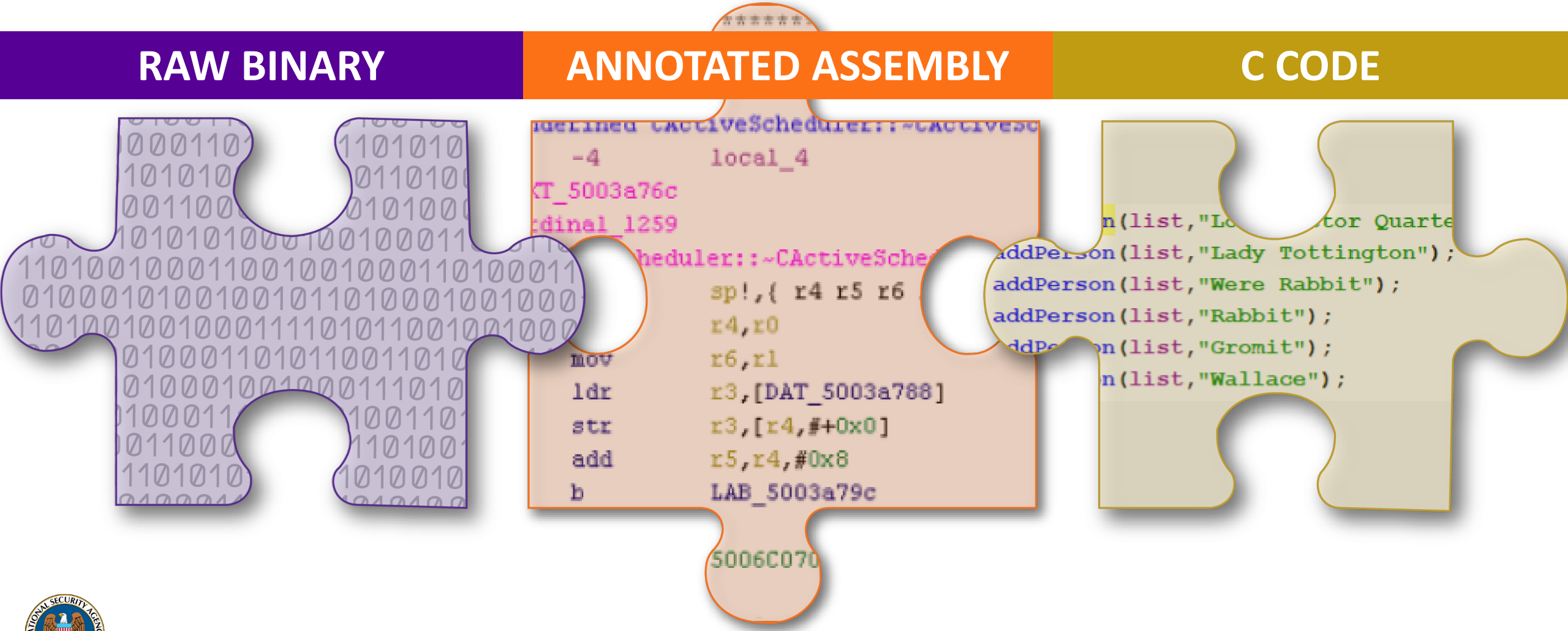


GHIDRA

SOFTWARE REVERSE ENGINEERING TOOL SUITE

Ghidra Purpose - What's in Your Binary?

Assembling the Puzzle



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**

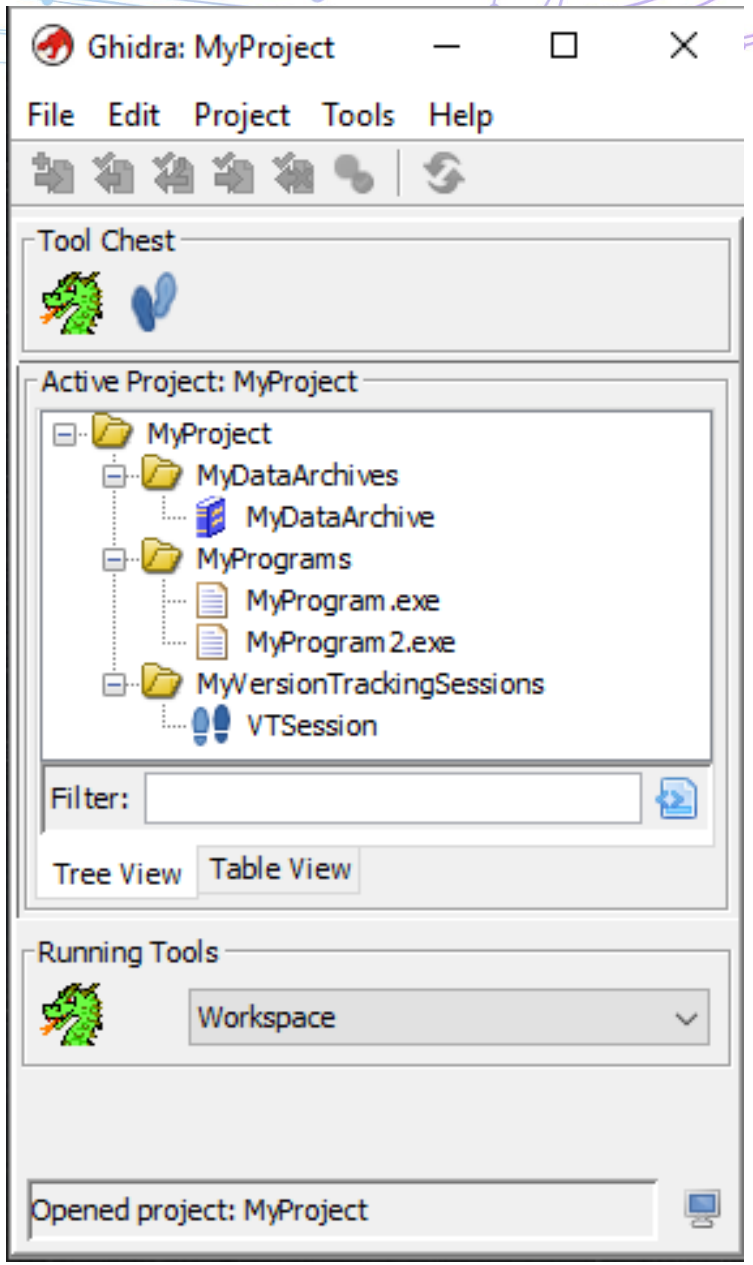
Why Did We Release Ghidra?

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

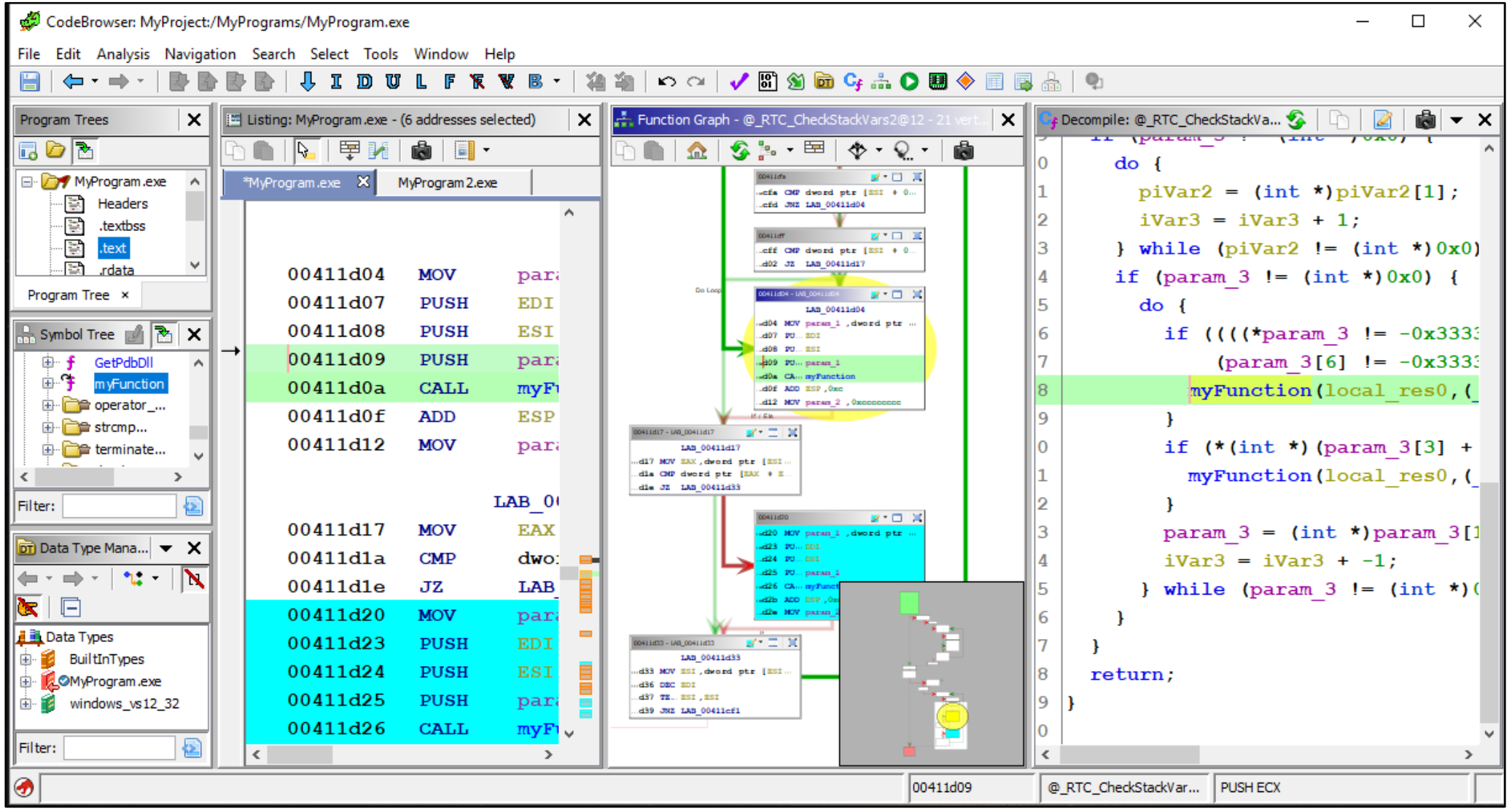
A product of NSA's Research Organization

Get Started in the Project Manager

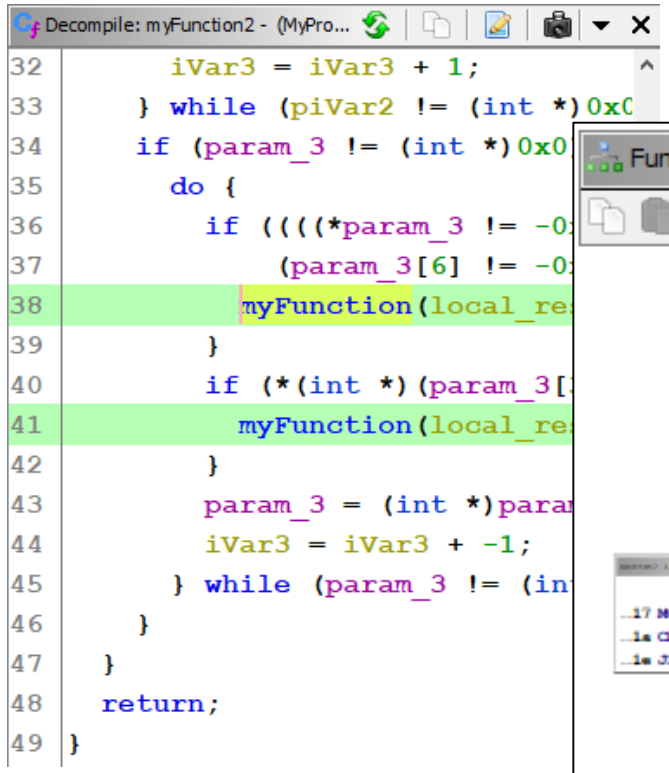
- Open a new project
- Drag your binary into the project

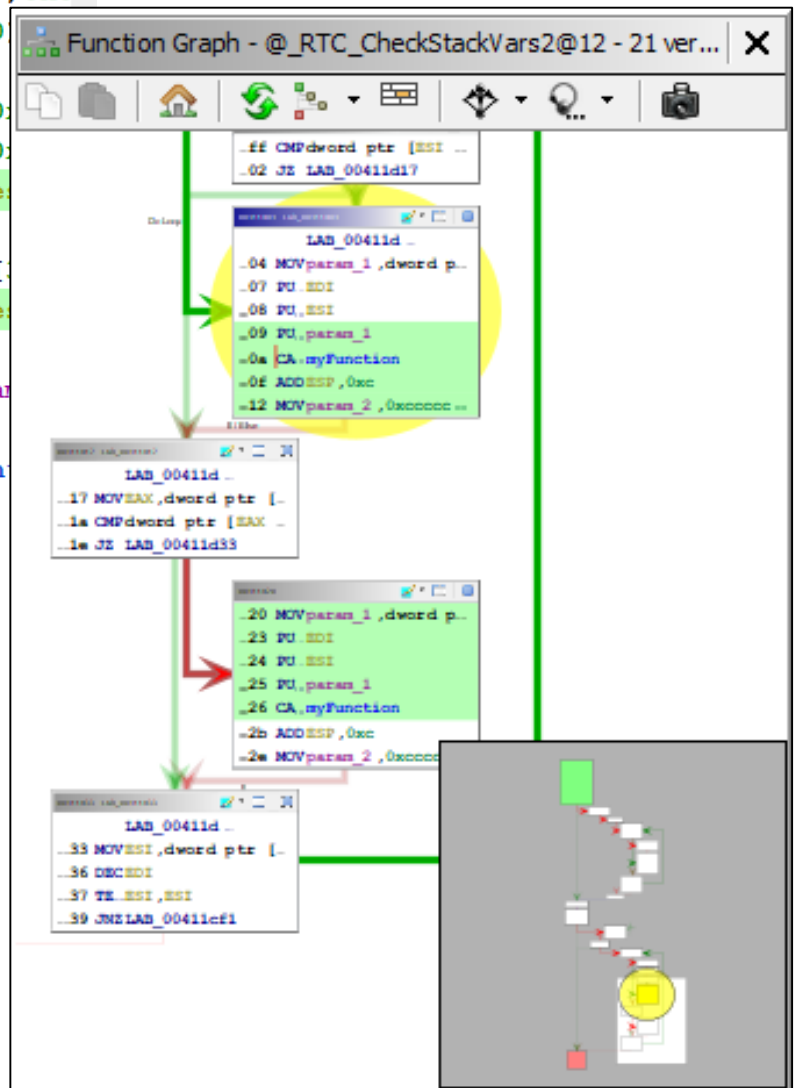


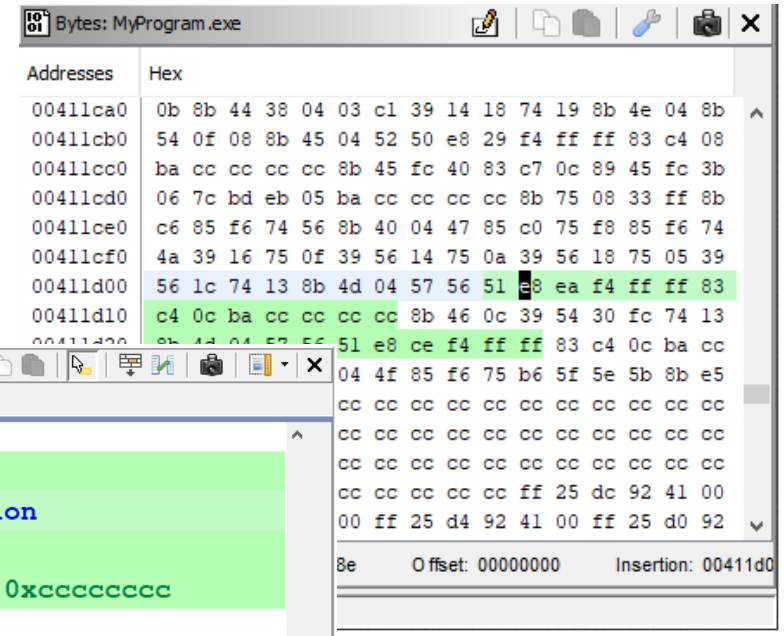
Configurable Environment

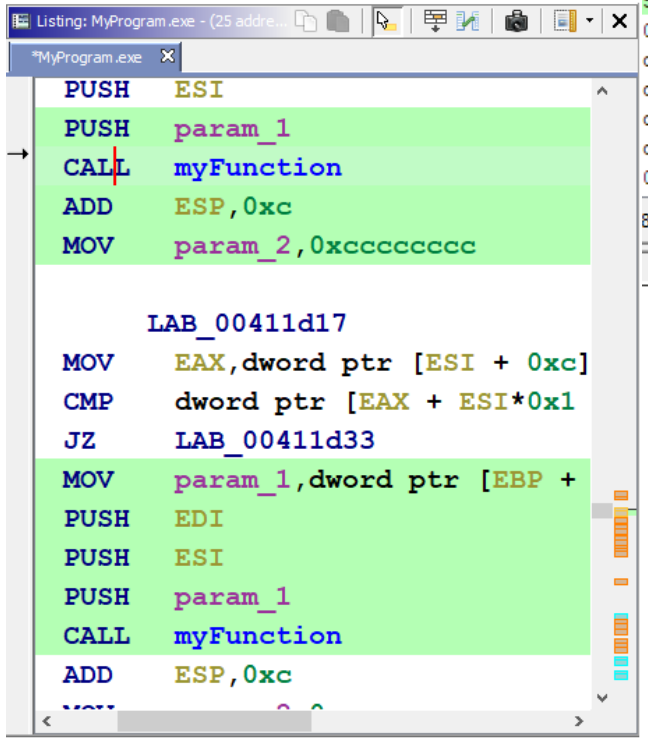


Many Views All in Sync









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
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 (((lVar1 != 0) && ((*(byte *)(lVar1 + 0
    (*(void **)(lVar1 + 8) != (void *)0x0))
  array_dispose(*(void **)(lVar1 + 8));
  *(undefined8 *)(lVar1 + 8) = *(undefinec
}
DAT_006df630 = *(undefined8 *)(puParm1 + 0
DAT_006df638 = *(undefined8 *)(puParm1 + 0
DAT_006dc7c8 = puParm1[0x10];
DAT_006dde7c = puParm1[0x11];
```

Decompiler

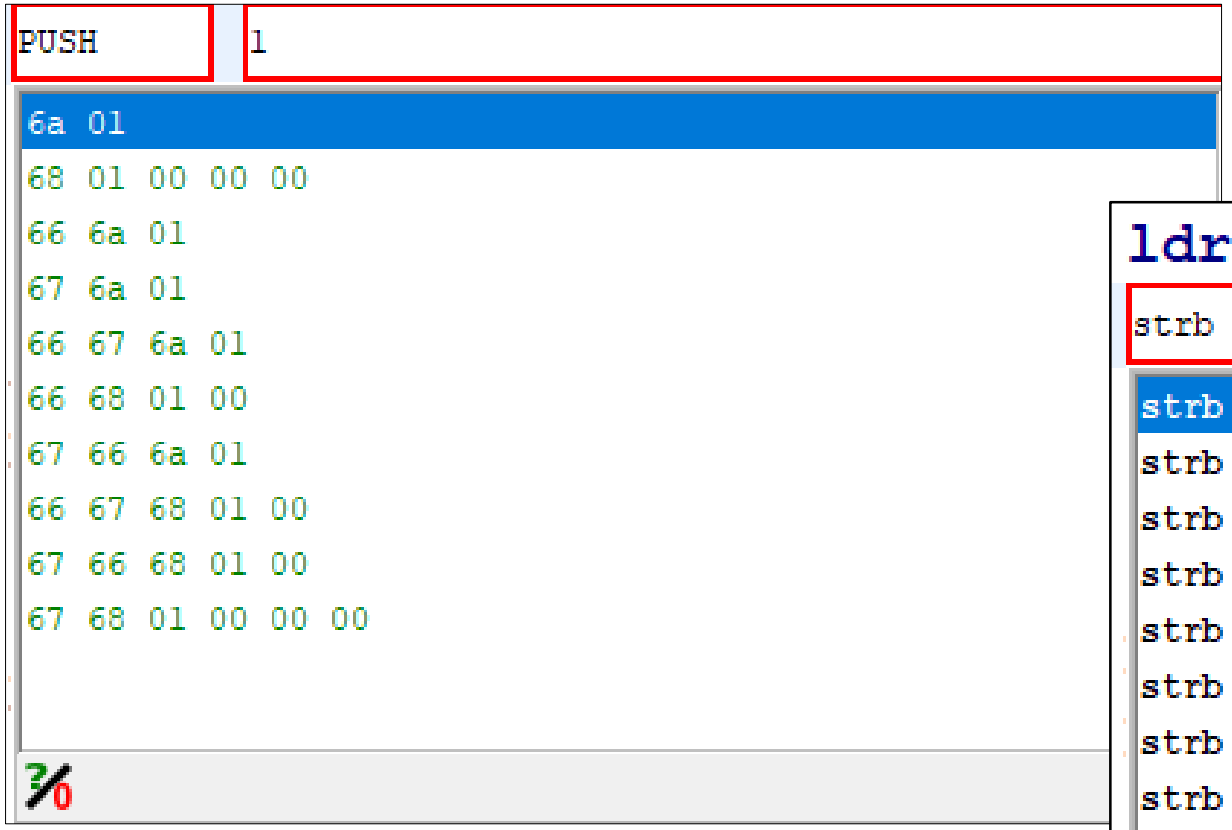
```
MOV     shell_var,qword ptr [RSP + 0x8]

MOV     qword ptr [RAX + shell_var->valu…
```

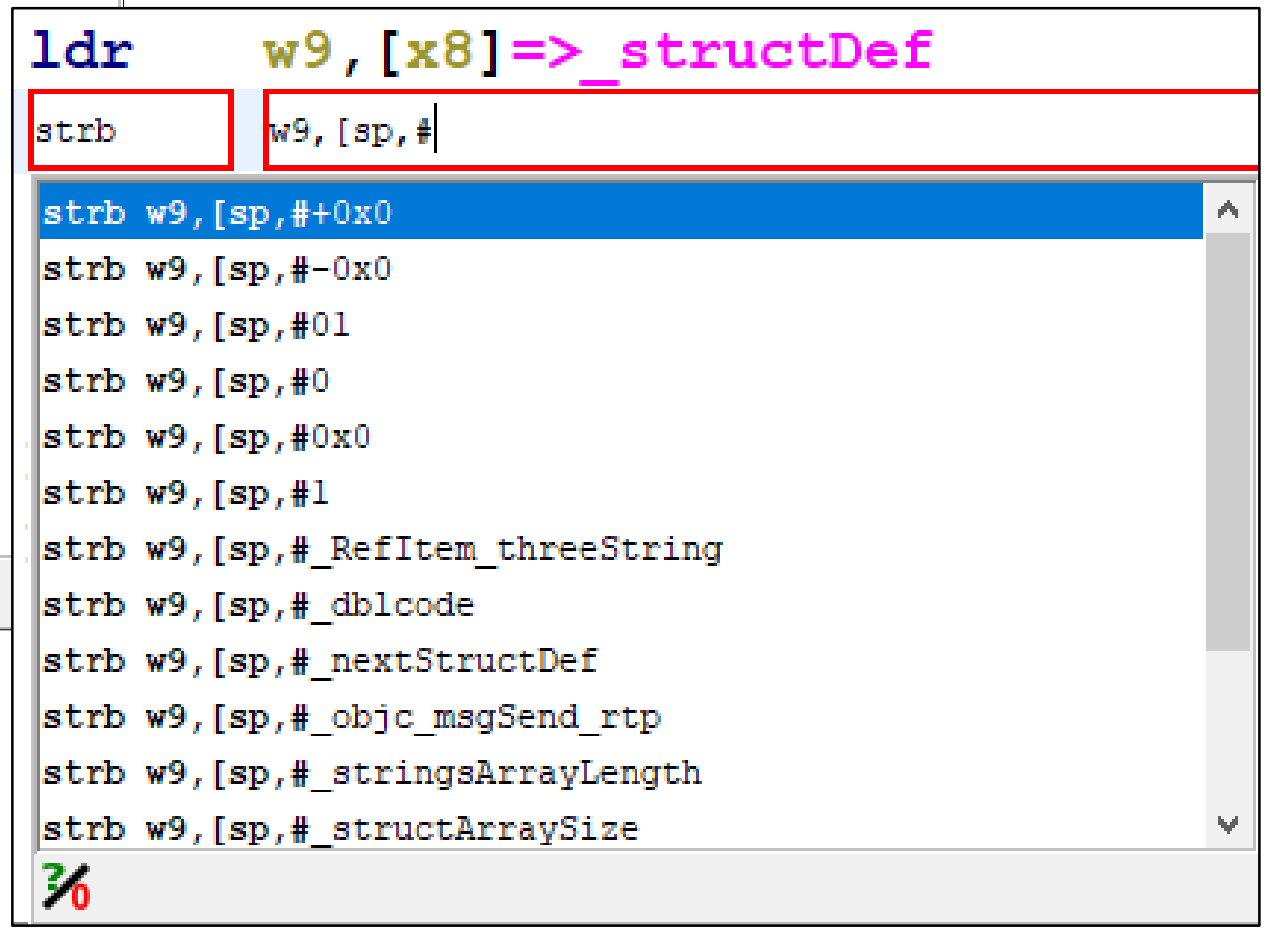
```
LAB_0041de45                        XREF[3]: 0041de1d(j),
                                             0041de23(j),
                                             0041de2c(j)
MOV     shell_var,qword ptr [RBX + 0x30]

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
```

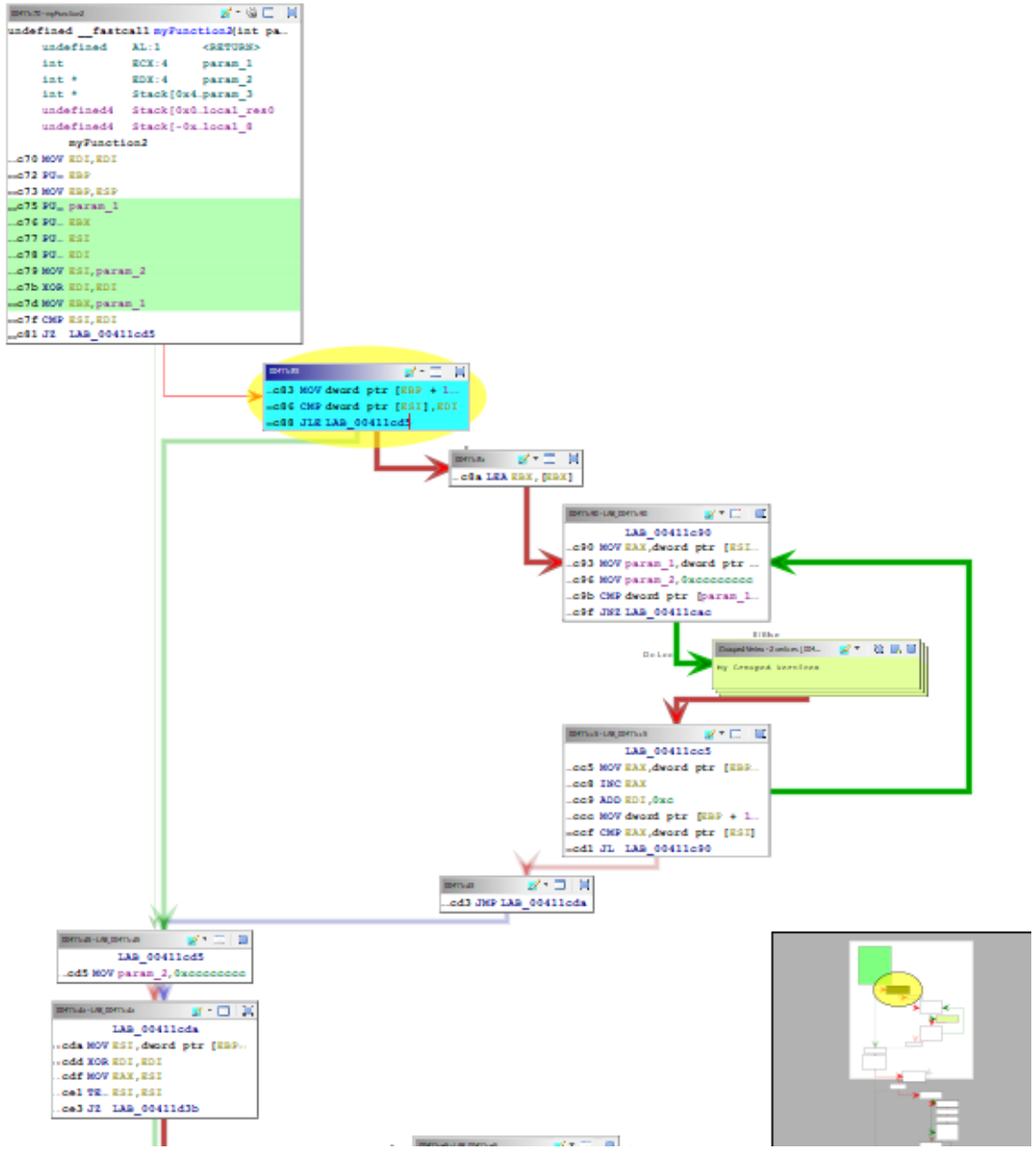
```
if (((shell_var != (SHELL_VAR *)0x0) && ((
    (shell_var->value != (char *)0x0)) {
  array_dispose(shell_var->value);
  shell_var->value = (char *)ps->pipestatu
}
DAT_006df630 = ps->last_shell_builtin;
DAT_006df638 = ps->this_shell_builtin;
DAT_006dc7c8 = ps->expand_aliases;
DAT_006dde7c = ps->echo_input_at_read;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

In-line Assembler

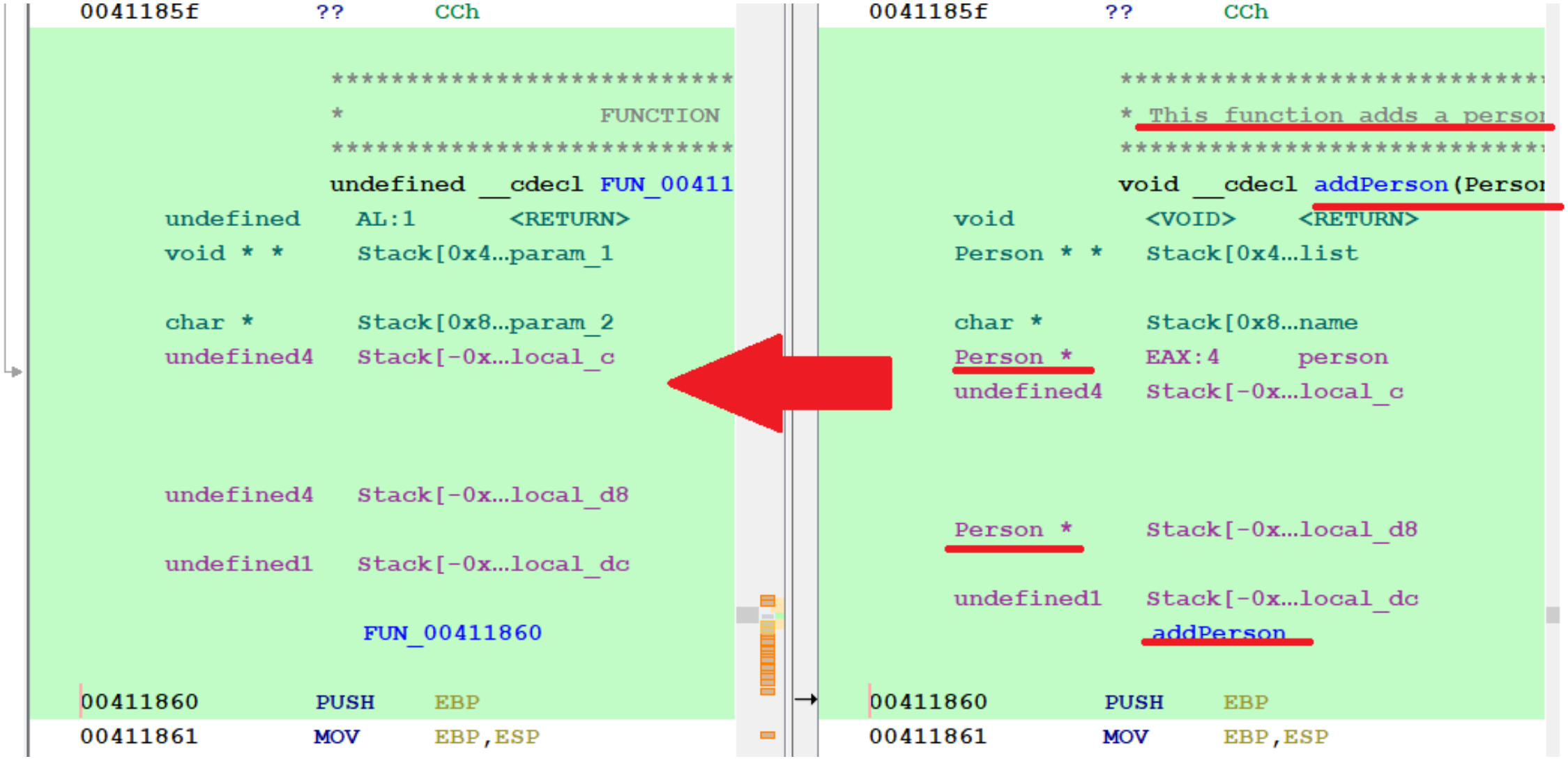




Function Graphs

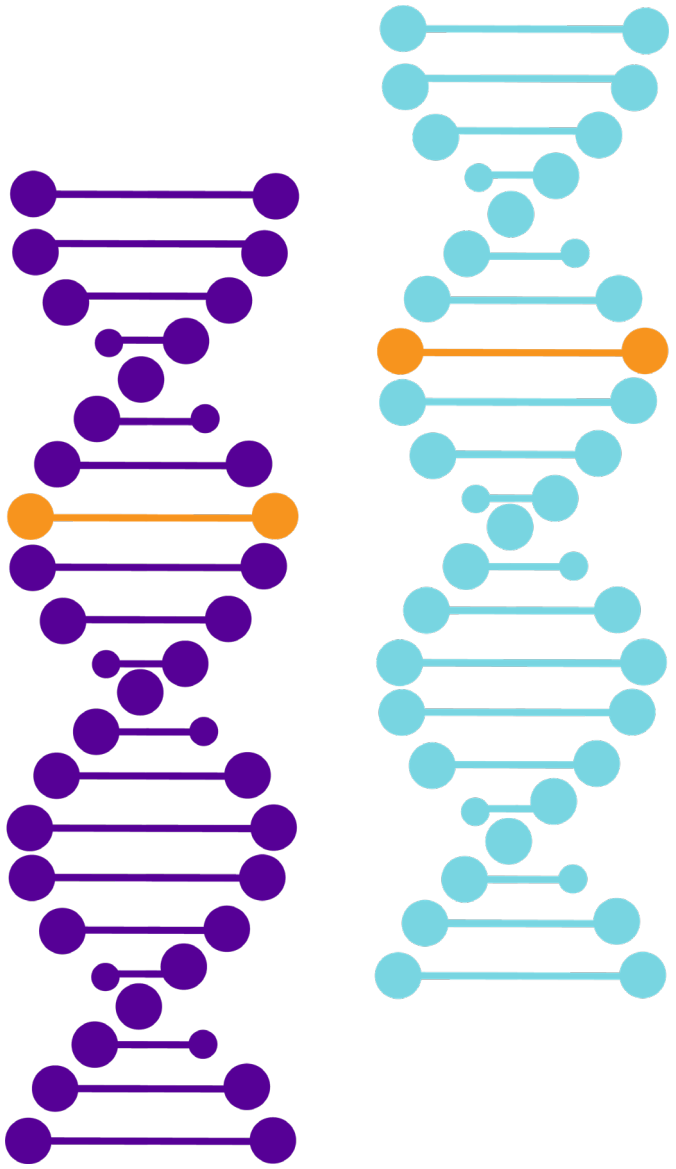


Annotated Differences



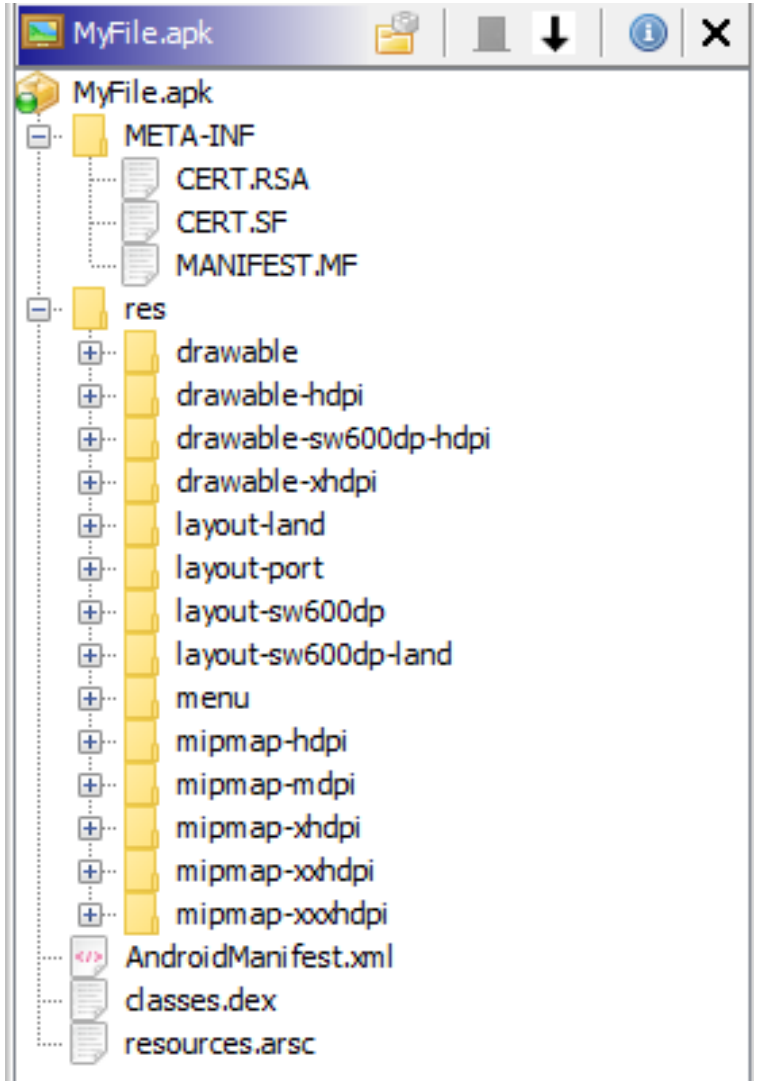
Version Tracking

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



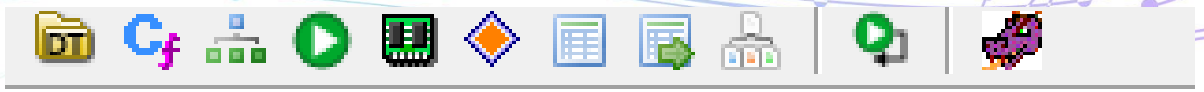
File System

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



Powerful Scripting

- Extends Ghidra
- Tightly integrated

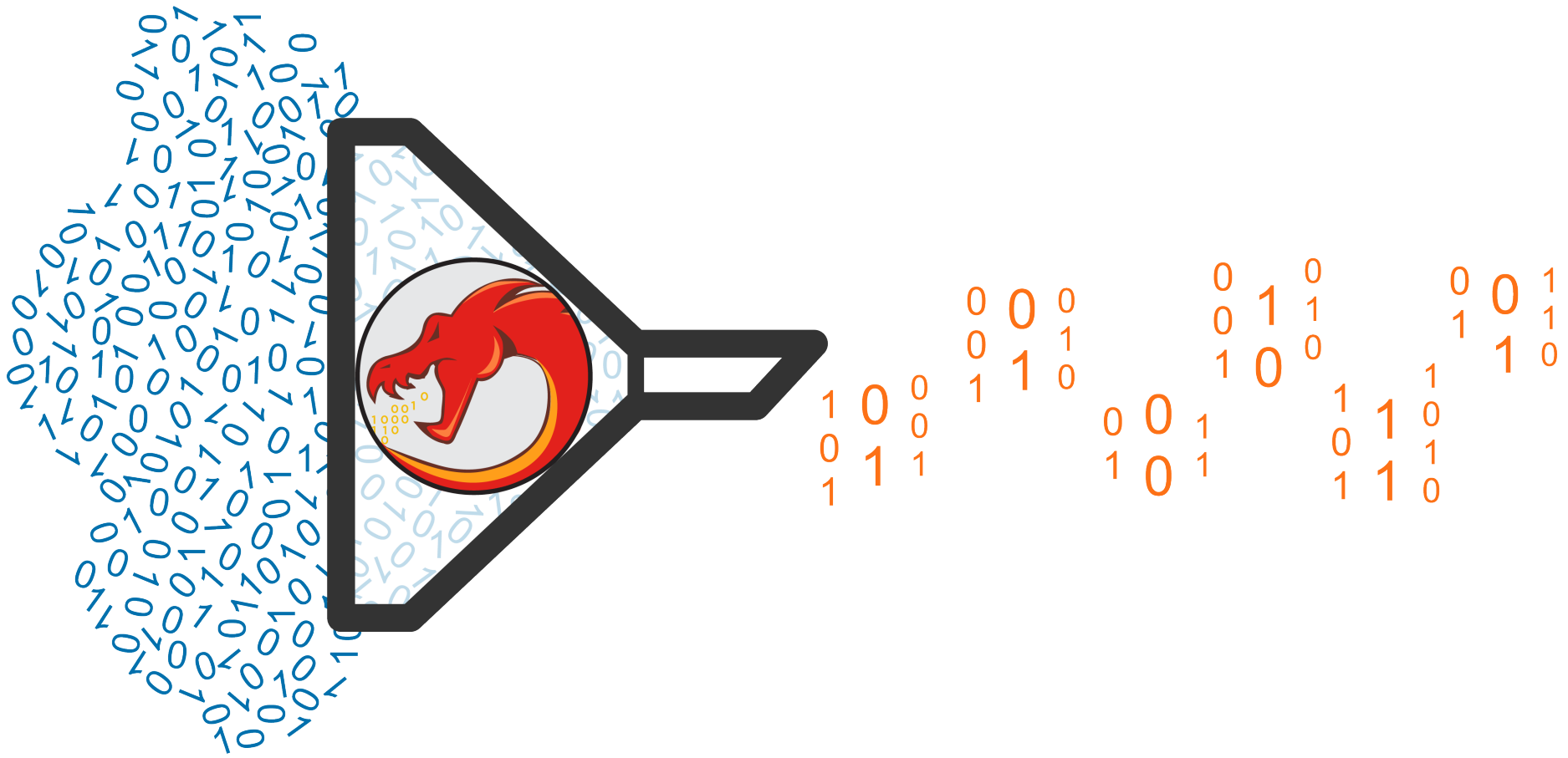


MyScript.java

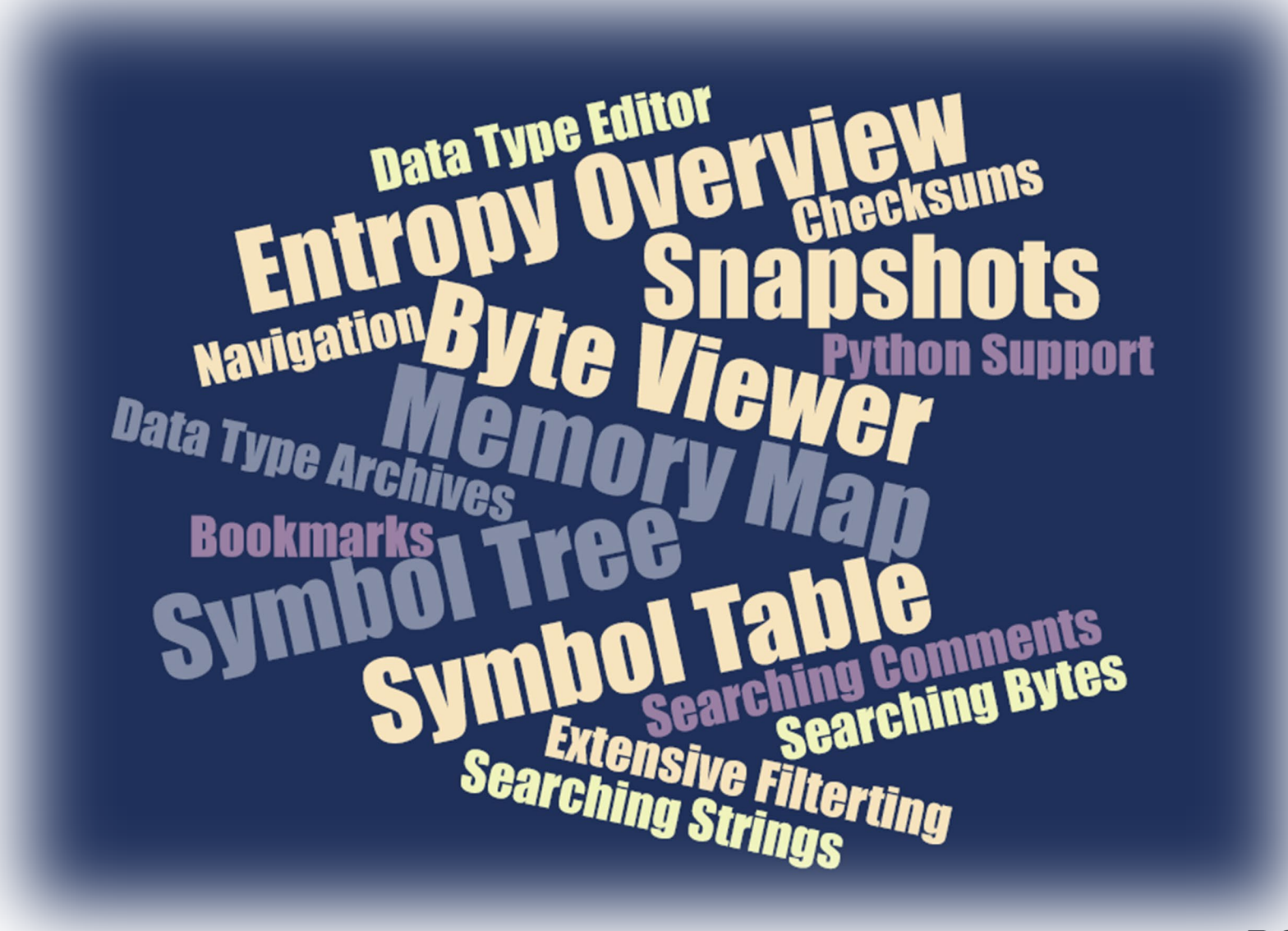
```
//My cool new script!
//@author Rob Joyce
//@category My Scripts
//@keybinding Alt R
//@menupath File.My Scripts
//@toolbar purple-Dragon.gif

import ghidra.app.script.GhidraScript;

public class MyScript extends GhidraScript {

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

}
```

Automating Analysis

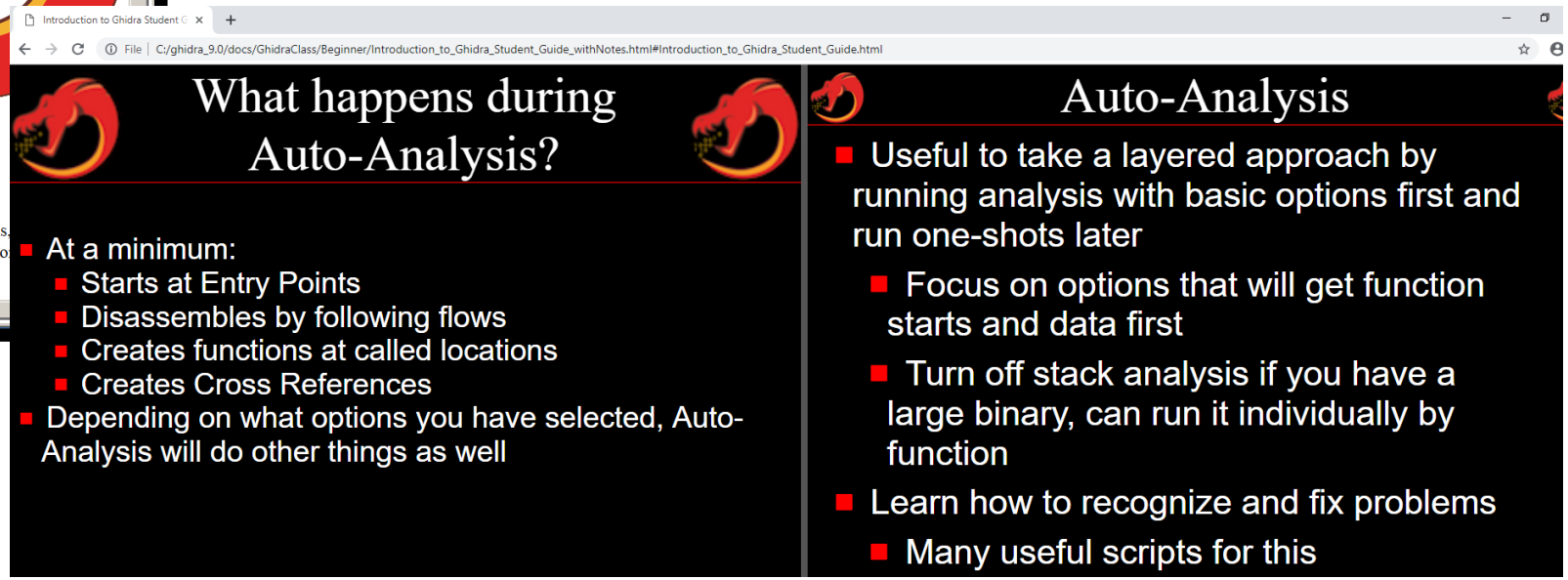
- Batch run Ghidra scripts without the GUI



And More Features Including:



Learning Ghidra

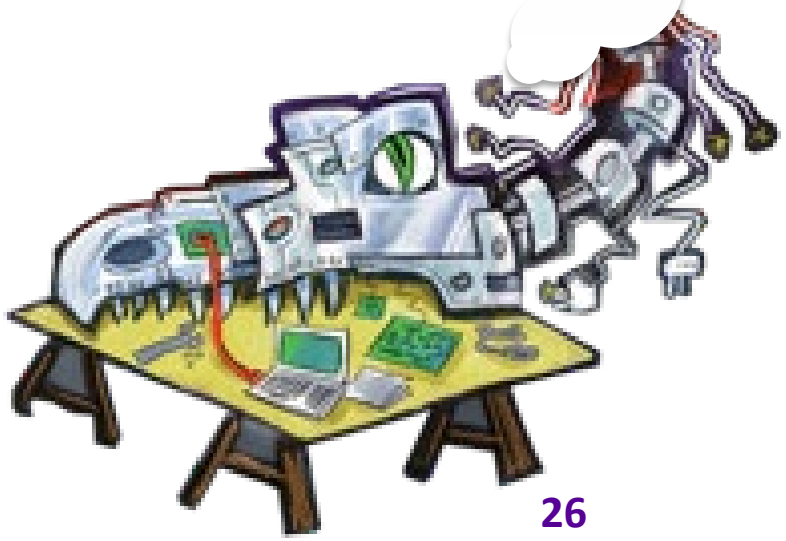


What's Next For Us?

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