Automating Binary Analysis with Ghidra's P-Code

Hacktivity 2022 - Gergely Revay

Gergely (Geri) Revay

Security Researcher at FortiGuard Labs

- Comes from Hungary
- Lives in Germany
- M.Sc. in Computer Engineering specialized on Information and Network Security
- 4 years as QA tester at a Firewall vendor (Balabit)
- 7 years as penetration tester at OptimaBit and Siemens both in Germany and USA
- 3 years offensive security research at Siemens with focus on binary analysis and reverse engineering
- Author of various online courses: https://hackademy.aetherlab.net
- FortiGuard Labs researcher doing malware reverse engineering and threat intel
- Youtube channel: https://youtube.com/aetherlabnet
- Twitter: @geri_revay

Agenda

- Ghidra
- Ghidra Scripts
  - Intro
  - Python vs Java
  - Headless Mode
  - Flat API vs SDK
- P-Code
  - Intro
  - Raw P-Code
  - High P-Code
  - But Why?
  - Examples
- Recap
- Q‘n‘A

https://media.makeameme.org/created/agenda-5b5740.jpg

What is Ghidra

NSA releases Ghidra, a free software reverse engineering toolkit

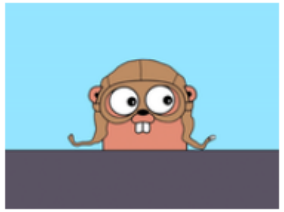
NSA's Ghidra greeted with positive reviews by the infosec community.

By Catalin Cimpanu for Zero Day | March 6, 2019 -- 02:12 GMT (02:12 GMT) | Topic: Security

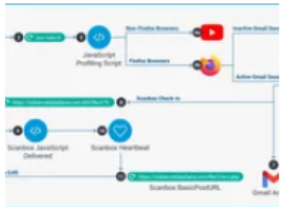
Image: NSA

MORE FROM CATALIN CIMPANU

Security
Chrome will soon try HTTPS first when you type an incomplete URL

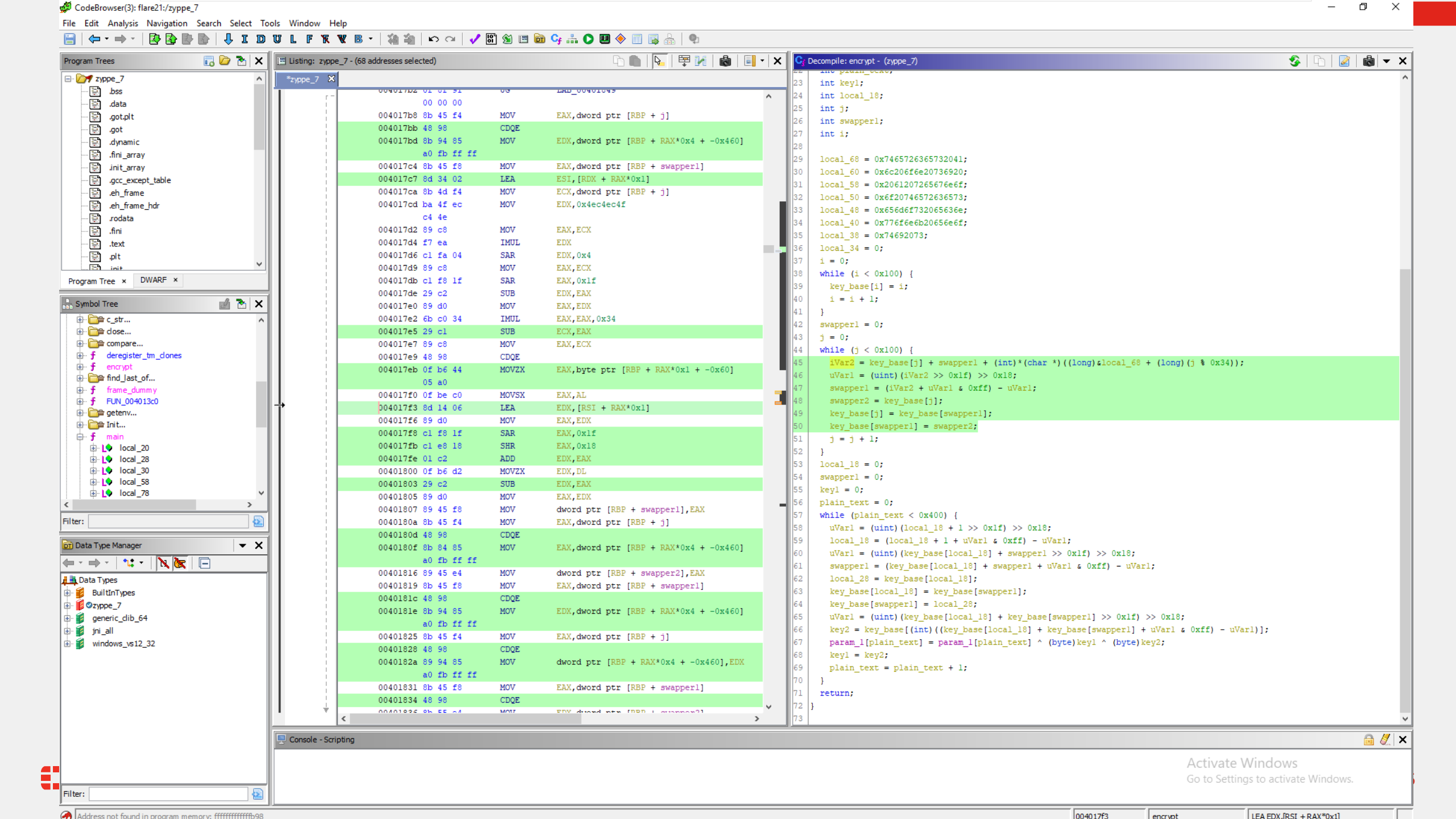


Security
Go malware is now common, having been adopted by both APTs and e-crime groups



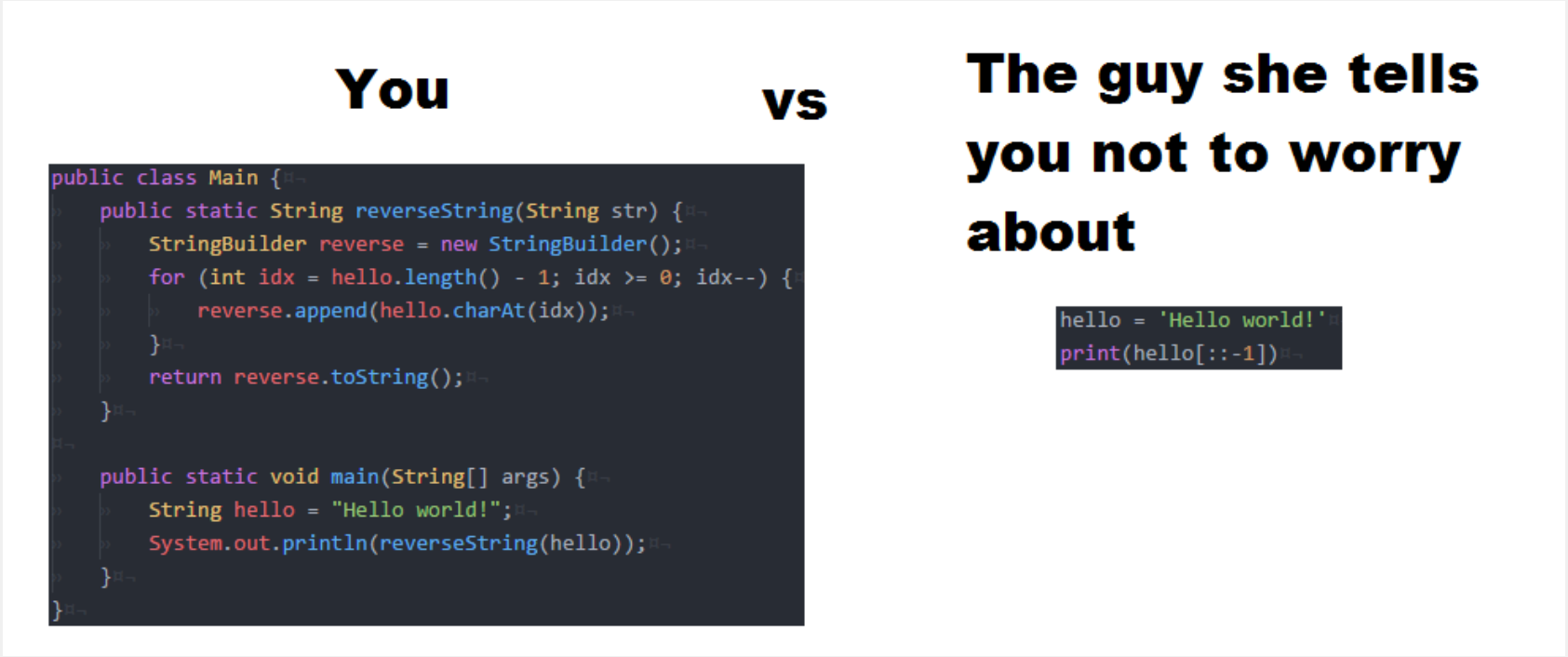
Security
Chinese cybersples targeted Tibetans with a malicious Firefox add-on

https://www.zdnet.com/article/nsa-release-ghidra-a-free-software-reverse-engineering-toolkit/

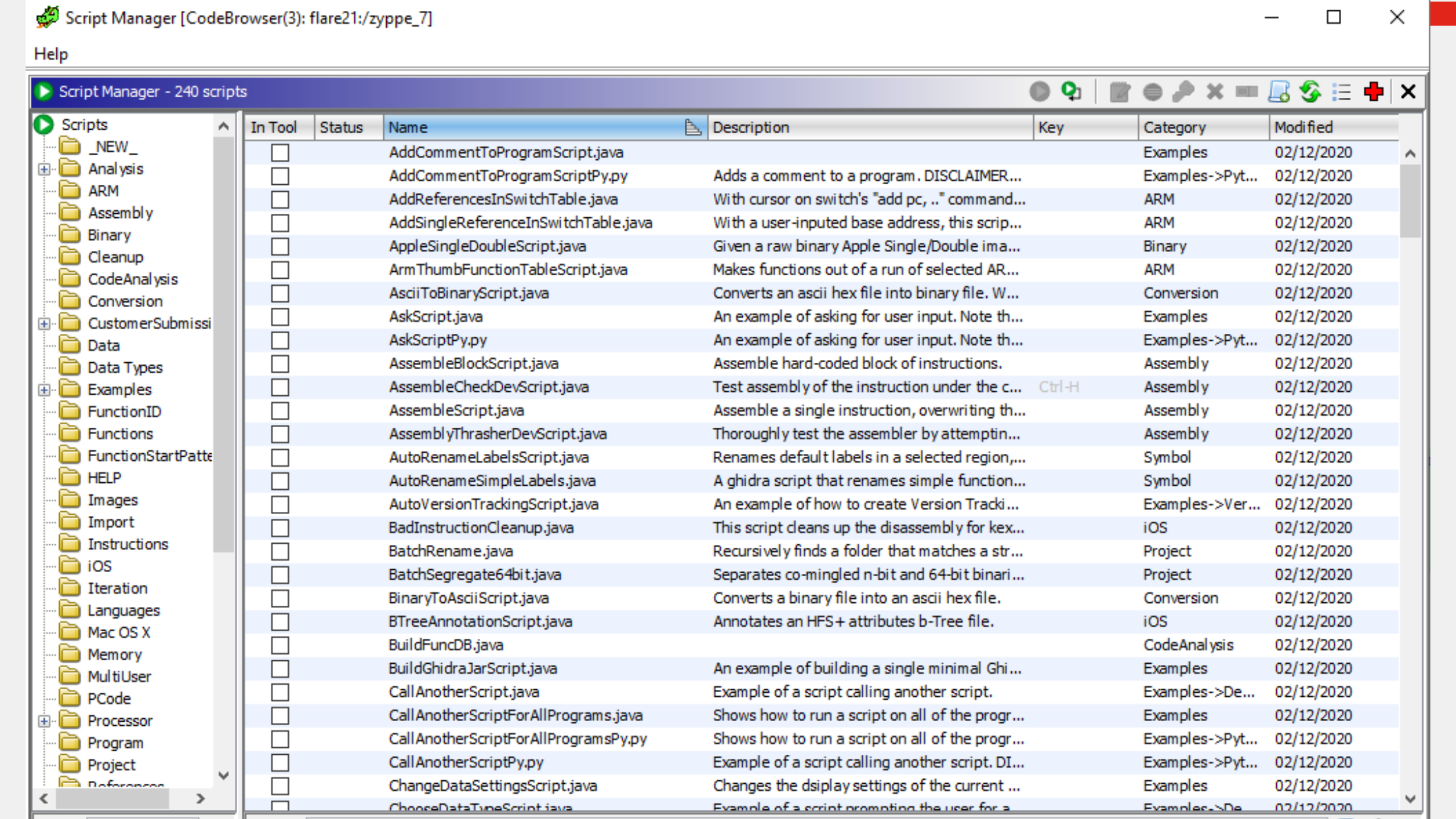

CodeBrowser(3): flare21:/zyppe_7
File Edit Analysis Navigation Search Select Tools Window Help
Program Trees
zyppe_7
.bss
.data
.got.plt
.got
.dynamic
.fini_array
.init_array
.gcc_except_table
.eh_frame
.eh_frame_hdr
.rodata
.fini
.text
.plt
Program Tree
DWARF
Symbol Tree
c_str...
close...
compare...
deregister_tm_clones
encrypt
find_last_of...
frame_dummy
FUN_004013c0
getenv...
Init...
main
local_20
local_28
local_30
local_58
local_78
Filter:
Data Type Manager
Data Types
BuiltInTypes
zyppe_7
generic_clib_64
jni_all
windows_vs12_32
Filter:
Listing: zyppe_7 - (68 addresses selected)
*zyppe_7
00 00 00
004017b8 8b 45 f4 MOV EAX,dword ptr [RBP + j]
004017bb 48 98 CDQE
004017bd 8b 94 85 MOV EDX,dword ptr [RBP + RAX*0x4 + -0x460]
a0 fb ff ff
004017c4 8b 45 f8 MOV EAX,dword ptr [RBP + swapper1]
004017c7 8d 34 02 LEA ESI,[RDX + RAX*0x1]
004017ca 8b 4d f4 MOV ECX,dword ptr [RBP + j]
004017cd ba 4f ec MOV EDX,0x4ec4ec4f
c4 4e
004017d2 89 c8 MOV EAX,ECX
004017d4 f7 ea IMUL EDX
004017d6 c1 fa 04 SAR EDX,0x4
004017d9 89 c8 MOV EAX,ECX
004017db c1 f8 1f SAR EAX,0x1f
004017de 29 c2 SUB EDX,EAX
004017e0 89 d0 MOV EAX,EDX
004017e2 6b c0 34 IMUL EAX,EAX,0x34
004017e5 29 c1 SUB ECX,EAX
004017e7 89 c8 MOV EAX,ECX
004017e9 48 98 CDQE
004017eb 0f b6 44 MOVZX EAX,byte ptr [RBP + RAX*0x1 + -0x60]
05 a0
004017f0 0f be c0 MOVSX EAX,AL
004017f3 8d 14 06 LEA EDX,[RSI + RAX*0x1]
004017f6 89 d0 MOV EAX,EDX
004017f8 c1 f8 1f SAR EAX,0x1f
004017fb c1 e8 18 SHR EAX,0x18
004017fe 01 c2 ADD EDX,EAX
00401800 0f b6 d2 MOVZX EDX,DL
00401803 29 c2 SUB EDX,EAX
00401805 89 d0 MOV EAX,EDX
00401807 89 45 f8 MOV dword ptr [RBP + swapper1],EAX
0040180a 8b 45 f4 MOV EAX,dword ptr [RBP + j]
0040180d 48 98 CDQE
0040180f 8b 84 85 MOV EAX,dword ptr [RBP + RAX*0x4 + -0x460]
a0 fb ff ff
00401816 89 45 e4 MOV dword ptr [RBP + swapper2],EAX
00401819 8b 45 f8 MOV EAX,dword ptr [RBP + swapper1]
0040181c 48 98 CDQE
0040181e 8b 94 85 MOV EDX,dword ptr [RBP + RAX*0x4 + -0x460]
a0 fb ff ff
00401825 8b 45 f4 MOV EAX,dword ptr [RBP + j]
00401828 48 98 CDQE
0040182a 89 94 85 MOV dword ptr [RBP + RAX*0x4 + -0x460],EDX
a0 fb ff ff
00401831 8b 45 f8 MOV EAX,dword ptr [RBP + swapper1]
00401834 48 98 CDQE
Decompile: encrypt - (zyppe_7)
23 int key1;
24 int local_18;
25 int j;
26 int swapper1;
27 int i;
28
29 local_68 = 0x7465726365732041;
30 local_60 = 0x6c206f6e20736920;
31 local_58 = 0x2061207265676e6f;
32 local_50 = 0x6f20746572636573;
33 local_48 = 0x656d6f732065636e;
34 local_40 = 0x776f6e6b20656e6f;
35 local_38 = 0x74692073;
36 local_34 = 0;
37 i = 0;
38 while (i < 0x100) {
39 key_base[i] = i;
40 i = i + 1;
41 }
42 swapper1 = 0;
43 j = 0;
44 while (j < 0x100) {
45 iVar2 = key_base[j] + swapper1 + (int)*(char *)((long)&local_68 + (long)(j % 0x34));
46 uVar1 = (uint)(iVar2 >> 0x1f) >> 0x18;
47 swapper1 = (iVar2 + uVar1 & 0xff) - uVar1;
48 swapper2 = key_base[j];
49 key_base[j] = key_base[swapper1];
50 key_base[swapper1] = swapper2;
51 j = j + 1;
52 }
53 local_18 = 0;
54 swapper1 = 0;
55 key1 = 0;
56 plain_text = 0;
57 while (plain_text < 0x400) {
58 uVar1 = (uint)(local_18 + 1 >> 0x1f) >> 0x18;
59 local_18 = (local_18 + 1 + uVar1 & 0xff) - uVar1;
60 uVar1 = (uint)(key_base[local_18] + swapper1 >> 0x1f) >> 0x18;
61 swapper1 = (key_base[local_18] + swapper1 + uVar1 & 0xff) - uVar1;
62 local_28 = key_base[local_18];
63 key_base[local_18] = key_base[swapper1];
64 key_base[swapper1] = local_28;
65 uVar1 = (uint)(key_base[local_18] + key_base[swapper1] >> 0x1f) >> 0x18;
66 key2 = key_base[(int)((key_base[local_18] + key_base[swapper1] + uVar1 & 0xff) - uVar1)];
67 param_1[plain_text] = param_1[plain_text] ^ (byte)key1 ^ (byte)key2;
68 key1 = key2;
69 plain_text = plain_text + 1;
70 }
71 return;
72 }
73
Console - Scripting
Activate Windows
Go to Settings to activate Windows.
Address not found in program memory: ffffffffffffb98
004017f3
encrypt
LEA EDX,[RSI + RAX*0x1]

Ghidra Scripts

- Like IDAPython
- Extend the functionality
- Full automation is possible
- Java or Python (Jython)
- FlatAPI vs SDK



https://www.reddit.com/r/ProgrammerHumor/comments/66jj7f/java_vs_python/


Script Manager [CodeBrowser(3): flare21:/zyppe_7]
Help
Script Manager - 240 scripts
Scripts
NEW
Analysis
ARM
Assembly
Binary
Cleanup
CodeAnalysis
Conversion
CustomerSubmissi
Data
Data Types
Examples
FunctionID
Functions
FunctionStartPatte
HELP
Images
Import
Instructions
iOS
Iteration
Languages
Mac OS X
Memory
MultiUser
PCode
Processor
Program
Project
In Tool
Status
Name
Description
Key
Category
Modified
AddCommentToProgramScript.java
Examples
02/12/2020
AddCommentToProgramScriptPy.py
Adds a comment to a program. DISCLAIMER...
Examples->Pyt...
02/12/2020
AddReferencesInSwitchTable.java
With cursor on switch's "add pc, .." command...
ARM
02/12/2020
AddSingleReferenceInSwitchTable.java
With a user-inputed base address, this scrip...
ARM
02/12/2020
AppleSingleDoubleScript.java
Given a raw binary Apple Single/Double ima...
Binary
02/12/2020
ArmThumbFunctionTableScript.java
Makes functions out of a run of selected AR...
ARM
02/12/2020
AsciiToBinaryScript.java
Converts an ascii hex file into binary file. W...
Conversion
02/12/2020
AskScript.java
An example of asking for user input. Note th...
Examples
02/12/2020
AskScriptPy.py
An example of asking for user input. Note th...
Examples->Pyt...
02/12/2020
AssembleBlockScript.java
Assemble hard-coded block of instructions.
Assembly
02/12/2020
AssembleCheckDevScript.java
Test assembly of the instruction under the c...
Ctrl-H
Assembly
02/12/2020
AssembleScript.java
Assemble a single instruction, overwriting th...
Assembly
02/12/2020
AssemblyThrasherDevScript.java
Thoroughly test the assembler by attemptin...
Assembly
02/12/2020
AutoRenameLabelsScript.java
Renames default labels in a selected region,...
Symbol
02/12/2020
AutoRenameSimpleLabels.java
A ghidra script that renames simple function...
Symbol
02/12/2020
AutoVersionTrackingScript.java
An example of how to create Version Tracki...
Examples->Ver...
02/12/2020
BadInstructionCleanup.java
This script cleans up the disassembly for kex...
iOS
02/12/2020
BatchRename.java
Recursively finds a folder that matches a str...
Project
02/12/2020
BatchSegregate64bit.java
Separates co-mingled n-bit and 64-bit binari...
Project
02/12/2020
BinaryToAsciiScript.java
Converts a binary file into an ascii hex file.
Conversion
02/12/2020
BTreeAnnotationScript.java
Annotates an HFS+ attributes b-Tree file.
iOS
02/12/2020
BuildFuncDB.java
CodeAnalysis
02/12/2020
BuildGhidraJarScript.java
An example of building a single minimal Ghi...
Examples
02/12/2020
CallAnotherScript.java
Example of a script calling another script.
Examples->De...
02/12/2020
CallAnotherScriptForAllPrograms.java
Shows how to run a script on all of the progr...
Examples
02/12/2020
CallAnotherScriptForAllProgramsPy.py
Shows how to run a script on all of the progr...
Examples->Pyt...
02/12/2020
CallAnotherScriptPy.py
Example of a script calling another script. DI...
Examples->Pyt...
02/12/2020
ChangeDataSettingsScript.java
Changes the dsiplay settings of the current ...
Examples
02/12/2020

Python vs. Java

https://pics.me.me/oh-noone-is-judging-you-sweetie-thats-your-conscience-talking-imgflip-com-no-51807818.png

- Whatever your flavor is, nobody is judging
- Ghidra is written in Java
- Python scripting works pretty well through Jython (python 2)
- Python 3: https://github.com/justfoxing/ghidra_bridge
- Java development environment is great
- Ghidra plugin for Eclipse
- Java dev is supported by NSA
- You might need to let go of your principles

Headless Mode

```
analyzeHeadless /Users/user/ghidra/projects MyProject -import hello.exe -preScript GetInfoScript.java
```

- Way to implement fully automated scripts
- Calling it could be nicer

https://sadanduseless.b-cdn.net/wp-content/uploads/2021/07/headless-gymnasts1.jpg

Flat API

- Interface to write simple scripts
- Reliability in long term is the main goal
- ~147 methods available
- GhidraScript class is a subclass of FlatProgramAPI
- Stick to it if possible

NOTE:

1. NO METHODS SHOULD EVER BE REMOVED FROM THIS CLASS.
2. NO METHOD SIGNATURES SHOULD EVER BE CHANGED IN THIS CLASS.

This class is used by GhidraScript.

Changing this class will break user scripts.

That is bad. Don't do that.

Program API

https://tenor.com/view/boom-mind-blown-mind-blowing-eyeglasses-gif-15569009

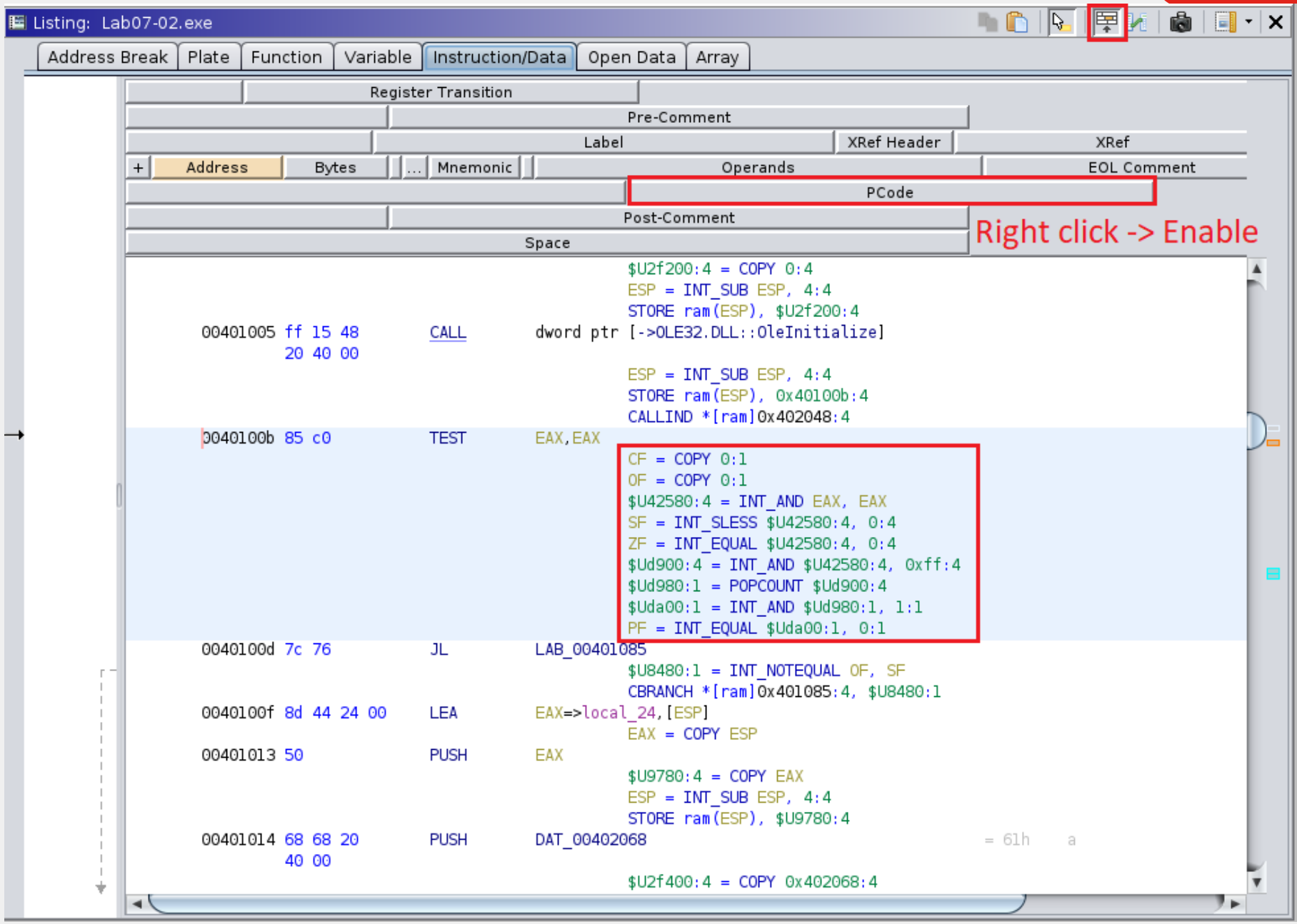
- Practically everything that is Ghidra
- All classes are available
- JavaDoc:
  https://ghidra.re/ghidra_docs/api/index.html
- Source
  https://github.com/NationalSecurityAgency/ghidra

Example 1: Listing imported functions

```
1  //This script lists all imported functions used by the binary.
7
8  import ghidra.app.script.GhidraScript;
10
11 public class brucon_01_list_imports extends GhidraScript {
12
13     @Override
14     protected void run() throws Exception {
15         String format = "%-30s %-30s\n";
16         SymbolTable symbolTable = currentProgram.getSymbolTable();
17         SymbolIterator symbolIter = symbolTable.getExternalSymbols();
18         printf("[-] Imported functions of %s\n\n", currentProgram.getExecutablePath());
19         printf(format, "Function Name", "Library");
20         printf("%s\n", "-".repeat(60));
21         for(Symbol symbol: symbolIter) {
22             printf(format, symbol.getName(), symbol.getParentSymbol());
23         }
24
25     }
26 }
```

P-Code

- Intermediate Representation: lies between the assembly code and the decompiled code that the Ghidra UI shows
- Register transfer language, it translates every individual processor instruction to a sequence of P-Code operations
- For each instruction it describes the processor instruction, including all side effects



Raw P-Code

- This is what you see when you start printing P-Code from a Ghidra Script (we will see later)
- There are no higher-level connections (arguments, variables, etc…)

```
                                        CALLIND (ram, 0x402048, 4)
0040100b 85 c0          TEST        EAX,EAX
                                    (register, 0x200, 1) = COPY (const, 0x0, 1)
                                    (register, 0x20b, 1) = COPY (const, 0x0, 1)
                                    (unique, 0x42580, 4) = INT_AND (register, 0x0, 4), (reg...
                                    (register, 0x207, 1) = INT_SLESS (unique, 0x42580, 4), ...
                                    (register, 0x206, 1) = INT_EQUAL (unique, 0x42580, 4), ...
                                    (unique, 0xd900, 4) = INT_AND (unique, 0x42580, 4), (co...
                                    (unique, 0xd980, 1) = POPCOUNT (unique, 0xd900, 4)
                                    (unique, 0xda00, 1) = INT_AND (unique, 0xd980, 1), (con...
                                    (register, 0x202, 1) = INT_EQUAL (unique, 0xda00, 1), (...
0040100d 7c 76          JL          LAB_00401085
```

High P-Code

- Result of HighFunction()
- “High-level abstraction associated with a low-level function made up of assembly instructions. Based on information the decompiler has produced after working on a function.”
- Closer to the C Pseudo Code
- Higher-level connections, such as function arguments, variables, etc...

https://southpark.fandom.com/wiki/Towelie_(character)

Why use P-Code?

- Because it’s cool
- Most of the binary analysis systems work with some kind of intermediate language.
- Compilers also use IR to separate the processes of parsing source code to a standardized format and translating the code to the machine code of the target architecture.
- Architecture independent
- Provides more information, then assembly (i.e. side effects)

https://pics.awwmemes.com/do-you-ever-think-happen-com-do-you-ever-think-screw-49053453.png

Demo

You get a Demo!

You get a Demo!

Everybody gets a DEMOOOOOO!

Example 2: Print high P-Code at the COM functions' call site

- Malware often use Component Object Model (COM) as obfuscation
- COM offers standardized interfaces to various functionality
  - Control IE through COM
  - Control the firewall through COM
- Script identifies if binary uses COM objects
- Prints call sites for COM functions

Example 3: Recover CLSID and IID to identify which COM object is used

- CLSID: globally unique identifier of a COM class objects
- CLSIDs and their associated programs are recorded in the registry: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{CLSID}
- To find out what COM object is used we need the CLSID
- IID: the interface that will be used to talk to the object.

```
C++

HRESULT CoCreateInstance(
  REFCLSID  rclsid,
  LPUNKNOWN pUnkOuter,
  DWORD     dwClsContext,
  REFIID    riid,
  LPVOID    *ppv
);
```

Recap

- Ghidra is great
- Ghidra scripts and P-Code have a lot of potential for automation
- Ghidra is relatively well documented
- There are still challenges when one starts to dig deep
- Choose your tool for the task

https://www.avira.com/de/blog/nsa-macht-ghidra-oeffentlich-zugaenglich

References and Thanks

- Alexei Bulazel and Jeremy Blackthorne: https://www.riverloopsecurity.com/blog/2019/05/pcode/
- Rolf Rolles: https://www.msreverseengineering.com/blog/2019/4/17/an-abstract-interpretation-based-deobfuscation-plugin-for-ghidra
- Carlos Garcia Prado (@m0n0sapiens) helped with his infinite wisdom

Thanks + Q&A

Gergely Revay

LinkedIn: https://www.linkedin.com/in/gergelyrevay/

Twitter: @geri_revay

Youtube: https://youtube.com/aetherlabnet


FORTINET