

12 Days of Ghidra

Nathan R

[Twitter](#) | [Mastodon](#)

\$whoami

Security Researcher

- Mobile devices
- Operating Systems

MSci Computer Science && PhD Student in InfoSec

Big fan of Hollow Knight and Star Wars

Day 0 – Hello, Ghidra!

Agenda

What is Reverse Engineering?

What is Ghidra

Ghidra Setup

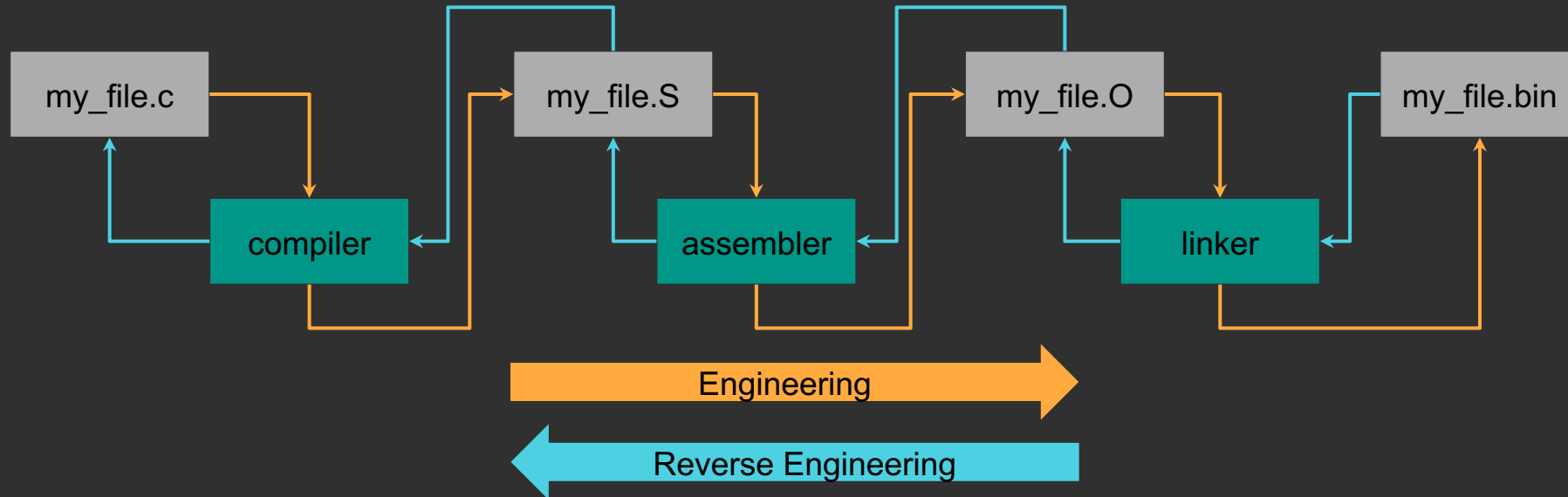
Ghidra walkthrough

Strings

What is Reverse Engineering (RE)?

Developer: source code -> binary format

Researcher: binary format -> source code



Why do we perform RE?

Understand what a binary is doing

Why?

- Malware Analysis
- Identify security issues
- Better understand undocumented features
- . . . ?

What is Ghidra?

Interactive software reverse engineering framework

Dissassembler

- Gives us a 'listing' of assembly code

Decompiler

- Attempts to present assembly code as close to source as possible

Its an investigative tool, you need to point Ghidra in the right direction :)

Ghidra Setup

Windows

Install a recent version of Java

Download Ghidra .zip from [Github](#)

Extract .zip to desktop (or any place you want)

Run **ghidraRun.bat**

Linux

Install Java JDK

- `sudo apt install openjdk-18-jdk`

Download Ghidra from Github

Run **ghidraRun.sh**

MacOS

Install [Brew](#)

- `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`

Install Java JDK

- `brew install --cask temurin`

Install Ghidra

- `brew install --cask ghidra`

ghidraRun

Ghidra on Apple Silicon

Need to rebuild support binaries for arm64:

Install gradle

- `brew install gradle`

Open Ghidra support directory

- `cd /opt/homebrew/Caskroom/ghidra/xx.x.x-xxxxxxx/ghidra_xx.x.x_PUBLIC/support`

Build support binaries for Arm

- `./buildNatives`

Using ghidra

Starting Ghidra

Navigate to Ghidra install directory

- `.../ghidra_<version>/`

Windows:

- `ghidraRun.bat`

Linux and MacOS

- `ghidraRun.sh`

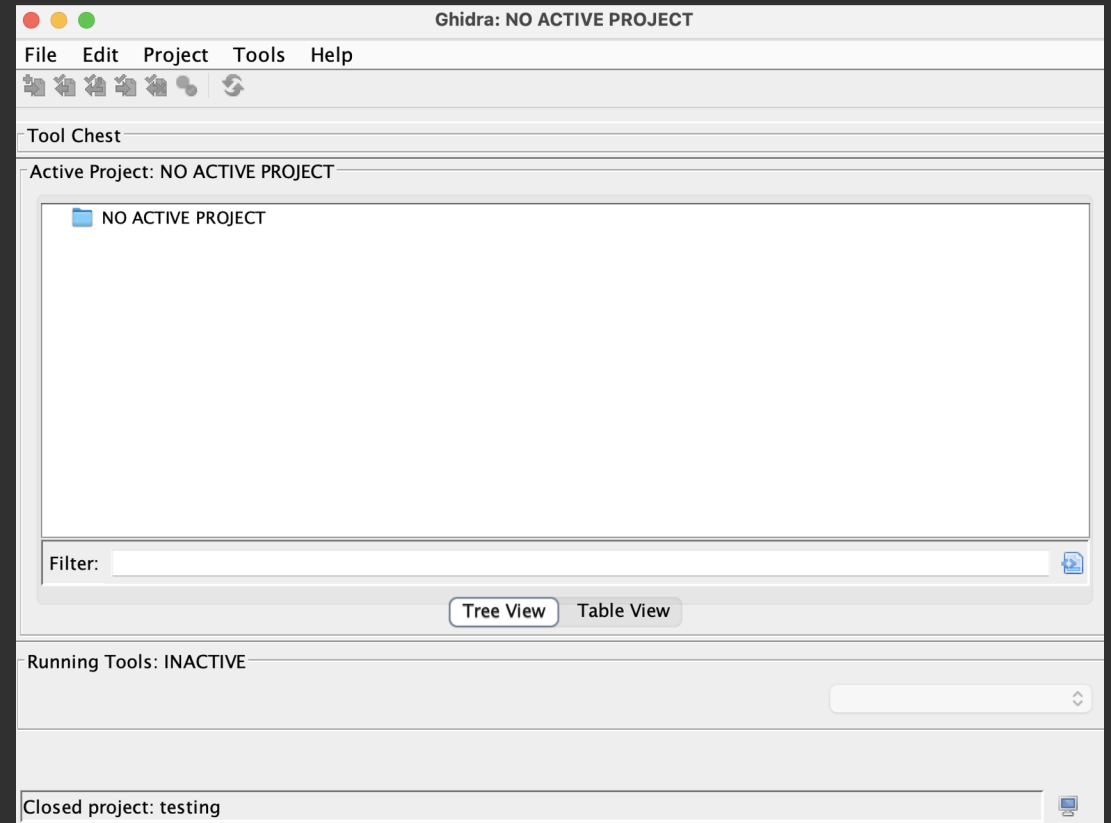
Program Manager

First window you will see in Ghidra

Hub for opening/creating projects

- Importing new files

Selecting tools



Program Manager – Creating a new Project

File -> New Project

Project Types:

- **Non-shared** for working alone
- **Shared** for working in group

Exercise 1:

- Create a new, non-shared project for 12 days of Ghidra



Tool Chest



Active Project: 12-days-of-ghidra

 12-days-of-ghidra

Filter:



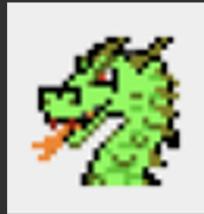
Tree View

Table View

Opening a Project in Ghidra

Select the project in the program manager

Click on the dragon button at the top



Importing a binary into Ghidra

Keybind -> I

- Or file -> import file

Select the correct compiler version

- Or make a best guess, tools like **binwalk** can be helpful

Select analysis you want to run

- Default will normally be enough

Ghidra Analysis

Aims to improve the quality of your decompilation

- Finding functions
- Finding cross-references (XRefs)
- Identifying different sections of memory
- Identifying stack variables
- Evaluating control flow

Analysis passes can take a while

- Defaults will be enough for these sessions

RE First Steps

Strings are a great first step when REing a new binary

Why?

- Can use them to identify what function is doing
- Error messages can give you good content
- Important data can be included in the binary, e.g. crypto keys

Ghidra has a built in Strings search tool

Day 0 Binary

1. Install Ghidra
2. Create a new project
3. Import day0.bin
4. Find the main function for your systems binary
5. Run the binary in a terminal to see what it does
6. See if you can find the flag in the binary
 - a. It may not be in the main function ;)

Flags are in the form vrc_flag{}

- DM me on Discord @nathanr to claim a flag :)

Prizes

We also have a couple of prizes to give away:

- Choice of book from No Starch Press
- Hoodie from @InterruptLabs

Categories:

1. For whoever completes all 12 binaries first
2. Most engaged participant
 - a. Helping on discord
 - b. Contributing to discussions
3. Best write-up

Day 1 – Crackme

Editing Function Signatures

Ghidra's analysis is a best guess

- Params and return values are often not the correct type

Ghidra allows you to edit the function signature

- Improves readability
 - array index in decompilation
 - Pointer types

RE challenge 1 -> adding correct function signatures

Ghidra Variable Types

Default variable names do not provide much context

- pVar1, pVar2, undefined8

Can be useful as you go along to rename variables to what you think they might be used for (keybind -> L)

- Remember strings from yesterday? Use them to help you :)

For instance - password_maybe, sus_checking_password

Can help you build a more complete picture as you go

Day 1 Crackme

You should RE the static check to produce the correct password string.

Suggestions:

1. Setup correct function signature for main
2. Modify variable names and types to help improve the decompilation

Hints:

- Consider how different data types impact the length of data read

Day 2 – XOR operator

Bitwise Operators and Obfuscation

Developers can try and hide data values using obfuscation

- You have already seen this with yesterday's binary ;)

XOR is a very common operator in this context:

$$A \wedge B = C$$

A = some text

B = key (needs to be same length as text)

C = XOR'D text

Day 2 Binary

Have a look at today's binary and see if you can break the password obfuscation scheme.

Suggestions:

- Write a (python) script to help you
- Make use of tools like CyberChef

Hints:

- XOR has an interesting property that allows you to reverse it when you only have partial information

Day 3 – More Crypto

Cryptography Terminology

Plaintext -> 'Normal' text

Ciphertext -> Text passed through a cryptography algorithm

Encryption -> Convert Plaintext to ciphertext

Decryption -> Convert Ciphertext back into Plaintext

Key -> Secret value that adds 'randomness'

Types of Crypto Algorithms

Encryption/Decryption

- Reversible
- Useful for hiding data

Hashing

- One-way (Not reversible)
- Useful for verification of data, or to improve search efficiency

Day 3 Binary

Improves the cryptography of the binary from yesterday

- No more XOR

Suggestions:

1. See if you can identify what the encryption scheme is
2. Reverse the encrypt algorithm and write the decryption function in a scripting language like Python

How might you find the encryption scheme?

- Variable names, any statically defined constants?

Day 4 – Debuggers

Static and Dynamic Analysis

Static Analysis

- Look at code without running it

Dynamic Analysis

- Run the binary and observe its behaviour

So far, we have focus on static analysis, Ghidra can also help us perform dynamic analysis

Debuggers

Allow us to step through program execution line by line

- View variable values
- Register Values
- Contents in memory

Why might we need to do this?

- Information that is only generated at runtime

Ghidra Debugger Tool

Connects to a system debugger

- GDB -> Linux
- LLDB -> MacOS
- WinDBG -> Windows

Set breakpoints

- Where you want to stop execution

View variable and memory contents in the GUI

Send input to the program

Setting up Ghidra Debugger

Open the **program manager** and click on **blue bug**

Open a terminal window – [here](#) for why

- Run “tty” and take note of value e.g. /dev/pts/0
- “sleep 1000000”

Inside Ghidra create a new GDB Target

- “IN-VM GNU gdb local debugger”

In Ghidra GDB Interpreter:

- “set inferior-tty <tty-val>” – e.g. set inferior-tty /dev/pts/0

Day 4 Binaries

Two binaries

1. Generic binary that should run on all OS's
2. A bonus binary that will only work on Linux and MacOS
 - a. Try using WSL if you are running windows

Suggestions:

1. Set breakpoints at places where dynamic values are being used, e.g. comparison operations

Day 5 – Structs

Structs

You may have noticed an AES struct in the previous binary

Structs are how variables are group in C / C++

They form a composite data type

- A type composed of different variables

How to know if it is a struct or an array?

Arrays are made up of elements of the same types

- Therefore, offset calculations can be consistent
 - `mov r0, array_base + (idx * sizeof(type))`

Structs are composed of many different types

- Therefore, offset calculations are going to be relative to each of the fields in the struct
 - `mov r0, struct_base + 0x4 // char (1 byte)`
 - `mov r1, struct_base + 0x8 // int (4 bytes)`
 - `mov r0, struct_base + 0x1E`

Ghidra Autostruct

Right-click on a variable and select 'create autostruct'

Can edit the variable type to update fields in the struct

Day 5 Binary

Suggestions:

- Use ghidra's autostruct feature to generate astruct types for a variable
- malloc() can be a good way to check that the array size is correct
- Check how each of the struct fields are used in the program logic
- Try and identify what file the application is expecting

Day 6 – Reversing C++ Objects

Intro to C++

Object-Oriented version of C

- C is a subset of C++

Introduces:

- Classes
 - Similar in principle to C structs
 - Difference is encapsulation - public, private, and protected keywords
- Methods
 - Functions that are bound to a class/object
 - I.e. the first parameter will always be 'this'

Constructors

Called when you create a new C++ object

- `Object obj = Object();`

Defines how you create a new object of that class type

- Typically, you will see some resource initialization

new vs malloc()

In previous binaries you may have noticed calls to **malloc()**

- malloc() allocates some memory and returns a pointer to this

new serves a similar purpose to malloc()

- In addition it also calls the class constructor

```
Object* obj = new obj();
```

```
obj->method();
```

Day 6 Binary

This is a more complicated binary than before

- Requires you to understand more about the program flow than before

There is no text prompts when you run the bin

- You need to RE what information to pass to main

Suggestions:

- Pay special attention to what happens inside of constructors

Day 7 – Ghidra Script

Introduction

Allows you to extend the functionality of ghidra

- E.g. -> [VTgrepGhidra](#)

Written in Python(2) or Java

- Python3 is available using third party extensions (not covered here)

API for interacting with core Ghidra components

- Program database -> var info, addr info

Setup

Can be open within Ghidra itself

- Window -> script manager || Window -> python

Ghidra integrates with Eclipse for better IDE support

- [Download Eclipse](#)
- On Ubuntu can use snap -> `snap install --classic eclipse`
- On MacOS use brew -> `brew install --cask eclipse-java eclipse-cpp`

Python Example

```
fm = currentProgram.getFunctionManager()  
  
funcs = fm.getFunctions(True) # True means 'forward'  
  
for func in funcs:  
    print("Function: {} @ 0x{}".format(func.getName(),  
func.getEntryPoint()))
```

Print the names of every function in the program

Day 7 Binary

You need to construct a key using GhidraScript

- Concatenate function names based on forward CFG

Suggestions:

- Demangle the function name to remove noise
- Open the binary in ghidra to find a good entry point function
- A recursive algorithm could be useful

Day 8 – Remote Connections

Day 7 Binary

Some binaries will communicate with external web-severs to get some data

- Encryption key
- Heart-beat

Suggestions:

- URLs are commonly obfuscated – how might you recover it?
- Does the webserver have any other resources you can access?

Day 9 – Network Programs

Day 9 Binary

Interact with the webserver to recover the key

Suggestions:

- Build on top of yesterday's script to develop some automated interaction
- Run the binary multiple times to see if anything changes
- See if any output from the previous run can be used to improve the next one

Let me know if you have any issues running this binary, it can be a little temperamental from my testing depending on your setup.

Day 10 – Function ID

Stripped Binaries

So far all of our binaries have had symbols included

- Function Names

A lot of binaries you see in the wild will be stripped

- No functions names

In Ghidra, all of the functions will just be called FUN followed by random values

This can make RE very time consuming, especially in static binaries

* What are the important functions?

Function ID

Ghidra has a feature to help with this called function ID

- Anyone familiar with IDA/Hex-Rays will know this as FLIRT

A database for function hashes

Ghidra compares the stripped function hash with a known function hash to see if they match

* If they do, this will be marked in the disassembly view

How do I create a .fidb?

Identify what library the function is from

- Versions must match

Compile a static binary with symbols using the target library

Load the known binary into Ghidra

Tools->Function ID->Create empty new fidDB

Tools->Function ID->Populate fidDB from programs

- Make sure the language matches the compiler used on the binary

Using the fidDB

Import a binary that matches the language specified when populating your db

Under the analysis options “Function ID” should appear

If not, make sure the compiler and language are the same

Day 10

More of an exercise than a challenge today on account of the many steps involved

Take today's binary and try and cover information using function id

I will consider this binary done when you can tell me:

1. What library is being used
2. What function is called for printing "hello world"

Suggestions:

- Remember to look at string information
- Google is your friend
- I recommend doing this in a Linux VM
- In order to statically compile a binary you need to use the `--static` flag
 - This looks for `.a` files rather than `.so` files

Day 11 – Final Challenge

Final Challenge

C++ Binary with multiple static checks to pass

- Both stripped and unstripped depending on how much of a challenge you want

Multiple approaches to this one:

- Reimplement in python and run yourself
- Use a mixture of dynamic and static analysis to bypass checks

Suggestions:

- Consider what is happening in constructors
- Create python scripts to automate your interactions

Day 12 – Wrap-up

What have we learned?

How to reverse engineer C/C++ binaries in Ghidra

Identifying and undoing common data obfuscation mechanisms

Working with cryptography algorithms

Working with network protocols

Dynamic analysis using Ghidra's debugging tool and GDB

Using GhidraScript to automate parts of our static analysis

Creating an fidDB for helping us recover information about a stripped binary

Prizes

First to complete all the challenges

- Joseph B

Community Prize (for being engaged and helping others)

- Hubert (__H)

Want More?

Books (anything by 'No Starch Press'):

- Hacking The Art of Exploitation – J Erickson
- Practice Reverse Engineering – Dang, Gazet, Bachaalany
- Practical Malware Analysis – Sikorski, Honig

Websites:

- Azeria Labs
- Malware Unicorn

Careers:

- Vulnerability/Security Researcher
- Security Consultant
- Threat Analyst

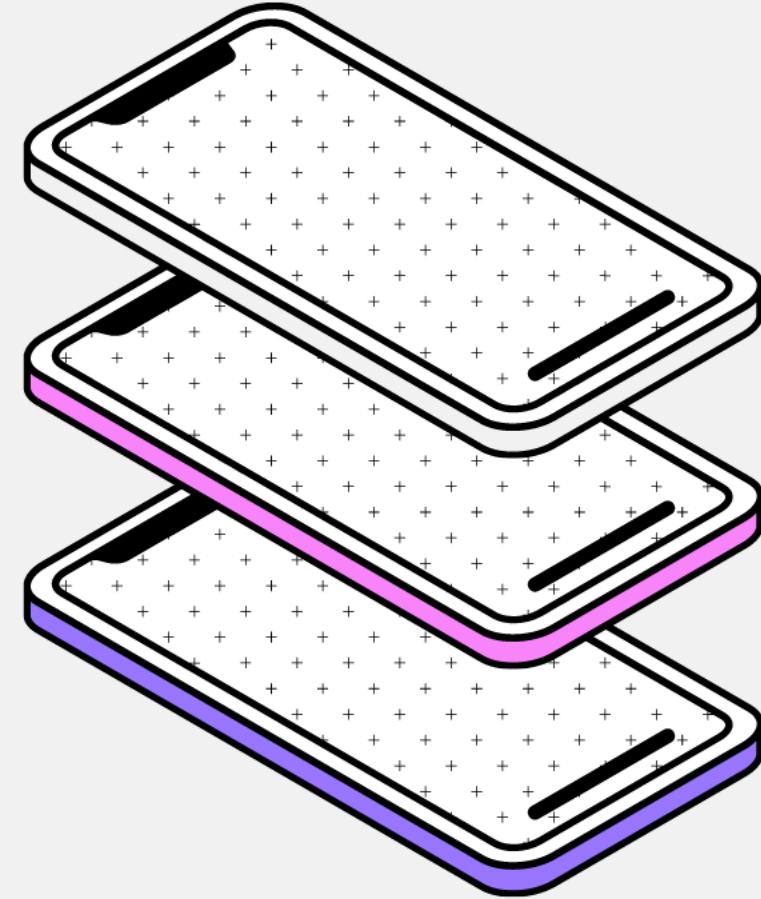
Keep involved with VR Campus on Discord :)

4 HOUR WEEKEND WORKSHOP (28TH OF JANUARY)

REGISTER ON VR CAMPUS NOW FOR A JOINER PACK!

MASTERING ANDROID APPLICATION REVERSE ENGINEERING

- Fundamentals of the **Java** programming language, Android apps, and OS.
- Android application development.
- **Reverse Engineer** Android applications and identify common **security misconfigurations**.



JAMESSTEVENSON.ME

TODO.COURSES