ARL-SR-0486 ● Dec 2023

Hands-on Cybersecurity Studies: Ghidra Plugin Scripts for Simple Decoding

by Alejandra De La Pena and Jaime C Acosta

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-SR-0486 ● DEC 2023

Hands-on Cybersecurity Studies: Ghidra Plugin Scripts for Simple Decoding

Alejandra De La Pena
University of Texas at El Paso

Jaime C Acosta
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
December 2023	Special Report	2/01/2023	9/30/2023

4. TITLE AND SUBTITLE
Hands-on Cybersecurity Studies: Ghidra Plugin Scripts for Simple Decoding

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Alejandra De La Pena and Jaime C Acosta

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-ND Adelphi, MD 20783	ARL-SR-0486

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Jaime C Acosta, 0000-0003-2555-9989

14. ABSTRACT
This document describes a hands-on cybersecurity exercise that focuses on developing a plugin script for the Ghidra publicly available and open-source software tool released by the National Security Agency in 2019. In the exercise, participants learn the basics of the tool and are then tasked with creating scripts that will automatically decode portions of a file to identify pertinent information.

15. SUBJECT TERMS
cybersecurity, reverse engineering, decoding, Collaborative Innovation Testbed, CyberRIG, Network, Cyber, and Computer Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	39

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Jaime C Acosta	(575) 993-2375

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures iv

1. Introduction 1

1.1 Exercise Overview 1

1.2 Ghidra Overview 1

1.3 Gaining a Deeper Understanding 2

2. Setup and Configuration 2

3. Learning Objectives 3

4. Exercise 3

4.1 Step 1: Decoding a Ciphertext Manually 4

4.2 Step 2: Introduction to Ghidra and the XOR Memory Script 6

4.3 Step 3: Encrypt Your Own Message in XOR 16

4.4 Step 4: Create Your Own Ghidra Script 21

5. Conclusion 29

6. References 30

List of Symbols, Abbreviations, and Acronyms 31

Distribution List 32

List of Figures

Fig. 1 Login window 4
Fig. 2 Material folder 5
Fig. 3 Ciphertext file 5
Fig. 4 Encrypted text 5
Fig. 5 Key template 5
Fig. 6 Description table 6
Fig. 7 Description table extended 6
Fig. 8 XOR folder 7
Fig. 9 Inside XOR folder 7
Fig. 10 Terminal icon 7
Fig. 11 Encoded executable 8
Fig. 12 Executable prompt 8
Fig. 13 Ghidra icon 9
Fig. 14 New project Ghidra 9
Fig. 15 Non-shared project Ghidra 10
Fig. 16 Project location Ghidra 10
Fig. 17 Importing a file in Ghidra 11
Fig. 18 Preconfigured options Ghidra 11
Fig. 19 Accessing Ghidra file 12
Fig. 20 Analyze Ghidra window 12
Fig. 21 Symbol tree window Ghidra 12
Fig. 22 Message location Ghidra 13
Fig. 23 Green play button 14
Fig. 24 Script manager Ghidra 14
Fig. 25 XorMemoryScript Ghidra 15
Fig. 26 XorValue Ghidra 15
Fig. 27 Copying command Linux 16
Fig. 28 XOR folder 17
Fig. 29 Testing.c 17
Fig. 30 Encryption/encoding CyberChef 18
Fig. 31 XOR option CyberChef 18

Fig. 32 Null preserving CyberChef 19
Fig. 33 Input area CyberChef 19
Fig. 34 Bytes format 20
Fig. 35 solLen variable 20
Fig. 36 Key variable 20
Fig. 37 Compiling testing.c 20
Fig. 38 Access granted output 20
Fig. 39 Voldemort's vault folder 21
Fig. 40 Terminal window 21
Fig. 41 Executable prompt 22
Fig. 42 Ghidra icon 22
Fig. 43 Script template folder 23
Fig. 44 Template.py 23
Fig. 45 Copy text 24
Fig. 46 Create new script Ghidra 24
Fig. 47 Script type 24
Fig. 48 Name script 25
Fig. 49 CaesarCipher script 26
Fig. 50 Save button 26
Fig. 51 Password location Ghidra 26
Fig. 52 Data string Ghidra 27
Fig. 53 CaesarCipherDecoder 27
Fig. 54 Caesar key Ghidra 28

1. Introduction

Encryption is the process of converting information into a secret code that can only be decoded or decrypted by someone who has the appropriate key or password. Encryption is used to protect sensitive data, such as personal information, financial data, and confidential messages, to avoid theft or undesired interception.[1] The main idea behind encryption is converting original data, also known as plaintext, and transforming it using encryption algorithms, known as a cipher, to produce ciphertext. Today multiple encryption ciphers are in wide use, and new ciphers are being developed. During this educational exercise, participants will explore the use of two widely recognized and effective ciphers, the XOR cipher and the Caesar cipher, providing participants with an engaging and productive workshop that imparts the fundamental principles of these encryption algorithms. This report explains how to employ the open-source reverse engineering tool Ghidra to decrypt messages encrypted using the XOR and Caesar ciphers. The exercise described provides participants with comprehensive knowledge about these encryption algorithms, as well as demonstrating the practical usage of Ghidra and its plugin scripts, and aims at teaching participants necessary skills to decipher simple obfuscated ciphertexts.

1.1 Exercise Overview

The objective of this workshop is to introduce participants to the encryption, with a specific focus on the XOR and Caesar ciphers. In this scenario-based exercise, participants assume the role of Harry Potter, who, through his extraordinary magical abilities, intercepts a message sent by the notorious Lord Voldemort. Despite his magical prowess, Harry finds himself unable to decipher the encrypted message using conventional magical techniques. As a result, he must turn to non-magical tools and techniques to break this encryption. Recognizing Ghidra as a powerful software for reverse engineering, Harry uses his expertise in code breaking and his familiarity with Ghidra's capabilities. Armed with this knowledge, Harry unveils the sinister content of the encrypted message, uncovering Lord Voldemort's malevolent plan to launch a devastating attack on Hogwarts School of Witchcraft and Wizardry. Harry quickly alerts the wizarding community and prepares to defend the school against the impending threat.

1.2 Ghidra Overview

Ghidra is a robust reverse engineering framework developed by the National Security Agency (NSA) and made available as an open-source tool.[2] While IDA

Pro and similar software often come to mind when thinking about reverse engineering, Ghidra offers comparable functionality and brings additional features to the table. One such feature is the ability to decompile binary files back into source code.[3] Another notable aspect of Ghidra is its extensibility through scripting, facilitated by its extensive application programming interface (API). This empowers users to develop custom scripts and plugins tailored to their specific requirements. By automating repetitive tasks and extending Ghidra's capabilities, these customized tools enhance efficiency and address users' unique needs.

1.3 Gaining a Deeper Understanding

During this exercise, participants will gain knowledge about two encryption cipher algorithms. They will also learn the fundamental uses of Ghidra, such as creating a new project and importing files into the Ghidra software. Furthermore, participants will understand the importance of creating their own Ghidra scripts tailored to their unique requirements. By running a Ghidra script and completing the provided tasks, participants will gain hands-on experience with basic Ghidra scripting syntax. The primary objective of this workshop is to instill confidence in participants in using Ghidra. Participants will discover that creating a script from scratch is not a daunting task, but rather an accessible and straightforward process, even for those with limited experience in this reverse engineering tool.

2. Setup and Configuration

The setup of the exercise includes a virtual machine (VM) with the following software.

- Oracle VirtualBox 6.1.30 64-bit[4]
- Kali Linux 2023.2 64-bit VM[5]
- Ghidra 10.3[6]
- CyberChef[7]

The US Army Combat Capabilities Development Command Army Research Laboratory South Cyber Rapid Innovation Group (CyberRIG) Collaborative Innovation Testbed[8] (CIT) is used to host the Kali Linux VM. The VM includes the Ghidra reverse engineering tool as well as several compiled binary executable files and sample plugin scripts. The VM is used by participants as their primary working environment to conduct binary analysis and plugin development.

Configurations required by the participant for the Kali VM are explained in the exercise.

3. Learning Objectives

The purpose of the exercise is to provide participants with a deeper understanding of encryption, reverse engineering, the XOR and Caesar ciphers, and Ghidra scripts. By delving into these topics, participants will gain valuable insights into the principles and techniques that underpin cryptographic systems, code-breaking methodologies, and custom script development.

The learning objectives associated with the exercise are as follows:

- **Encryption.** Exploring the purpose and significance of encryption, participants will understand how it plays a crucial role in protecting sensitive information from unauthorized access or interception. The workshop will delve into the XOR and Caesar ciphers.
- **Reverse Engineering and Ghidra.** Participants will be introduced to the concept of reverse engineering, a process of analyzing and understanding the inner workings of software. Specifically, they will discover the capabilities of Ghidra, an open-source reverse engineering tool developed by the NSA.
- **Ghidra Scripts.** Individuals will also gain knowledge for creating their own scripts based on their specific needs using Ghidra. They will use Ghidra's scripting capabilities and learn how to automate repetitive tasks, extend Ghidra's functionality, and tailor the tool to their unique requirements. By exploring Ghidra scripts, participants will unlock the potential for customization and efficiency in their reverse engineering workflows.

4. Exercise

The following exercise is intended to provide participants with a comprehensive walkthrough, comprising a series of step-by-step tasks. Each task will be conducted within the Kali Linux VM. It is essential to note that all data and systems involved in this exercise are entirely fictional and simulated, exclusively for educational purposes.

This mission briefing is presented to all participants to provide essential information and instructions for the upcoming exercise:

> *Story: You are Harry Potter, who has gained access to an encrypted message from Lord Voldemort. Despite your magical abilities, you cannot decode it. Seeking help, you turn to Ghidra, a powerful code analysis tool. Following a tutorial, you learn about encryption and attempt manual*

decoding. Discovering Ghidra's script feature, you realize it can simplify the process. With Ghidra's assistance, it is your job to decode the message and prevent Voldemort's plans.

Inside the Kali Linux VM, you will find Ghidra already installed, with its icon visible on the desktop. Each step of the workshop is represented by separate folders on the desktop. You will also come across two binary files that are preloaded with code. You will only need to make a few changes or additions to these binaries to ensure Ghidra works properly.

4.1 Step 1: Decoding a Ciphertext Manually

In this step, we will simulate a tutorial that Harry Potter discovered on the web. During this initial part, you will have the opportunity to interact with a straightforward cipher by manually decoding a ciphertext.

1) Open the Kali VM.

   If you are prompted for a login on the VM in the browser, use the following credentials (Fig. 1):

   username: **kali**

   password: **kali**

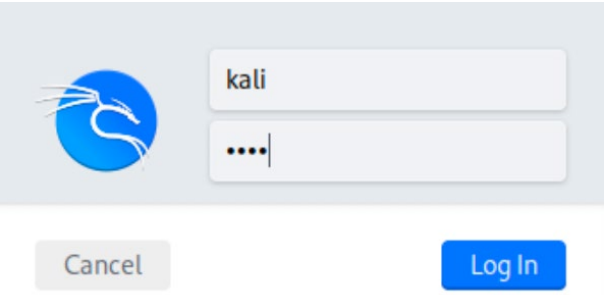

Fig. 1 Login window

2) To open the folder needed for this first step, access the desktop folder (Fig. 2)**.** Double-click in the folder **Decoding by Hand** to access the needed material.

Fig. 2 Material folder

3) Once the **Decoding by Hand** folder has been open, double-click the .txt file named **ciphertext.txt** (Fig. 3).

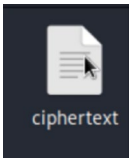

Fig. 3 Ciphertext file

Upon opening the ciphertext file, you will discover an encoded text that will require decoding through several subsequent steps (Fig. 4).

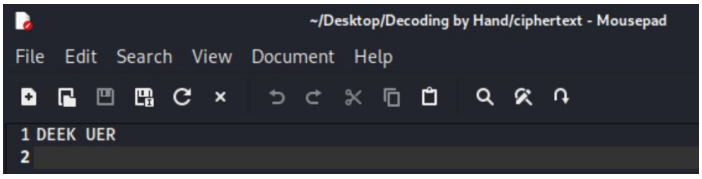

Fig. 4 Encrypted text

However, at this stage, it is important to focus on identifying the type of cipher used to encode the text, as the decoding process will be covered later.

4) Go back to the **Decoding by Hand** folder. Inside this folder you will find two PNG files, double-click the **Key Template** file (Fig. 5).

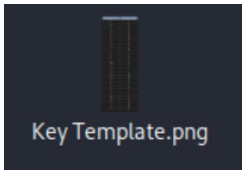

Fig. 5 Key template

By opening this file, you will gain access to the key that holds the solution to decrypting the ciphertext located in this very folder. Take a moment to carefully analyze the key, and then proceed to the next step accordingly.

5) In the same folder, access the **Description Table** (Fig. 6) file.

Fig. 6 Description table

By opening this file, you will be presented with multiple descriptions detailing various ciphers. Your task is to carefully analyze each descriptive section, shown in Fig. 7, and proceed to the next step accordingly.

Cipher Type	Description
Hill Cipher	The Hill cipher is a polygraphic substitution cipher based on linear algebra. It encrypts messages by multiplying the plaintext with a matrix and then taking the result modulo a number. The same matrix is used to decrypt the ciphertext.
XOR Cipher	An XOR cipher encrypts plaintext by bitwise XORing each character with a key. The same key is used decrypt the ciphertext.
Substitution Cipher	In a substitution cipher, each letter of the plaintext is replaced by another letter or symbol. The key specifies the substitution for each letter.

Fig. 7 Description table extended

6) After carefully analyzing the description table shown in Fig. 7, **what type of cipher do you think was used to encode the mysterious message found in the ciphertext.txt file**? _____

7) Now that you have everything you need and understand how the cipher for this ciphertext works, please take a moment to decode the text in the ciphertext.txt file._____

Great work! You have successfully deciphered the hidden message. As you experienced, manually decoding a text can be a time-consuming process. Now, you are prepared to witness how effortlessly tools can be employed to unravel concealed messages.

4.2 Step 2: Introduction to Ghidra and the XOR Memory Script

In this step, you will be introduced to Ghidra, a powerful software engineering tool that will greatly simplify your work with its multitude of features. In this

particular section of the tutorial, you will have the opportunity to explore the XOR cipher and engage with it directly.

1) On the desktop there is a folder named **XOR.** Double-click it to access the necessary material for this section (Fig. 8).

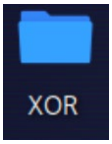

Fig. 8 XOR folder

2) The only file inside this folder is a C program named **encoded.c** (Fig. 9)**.**

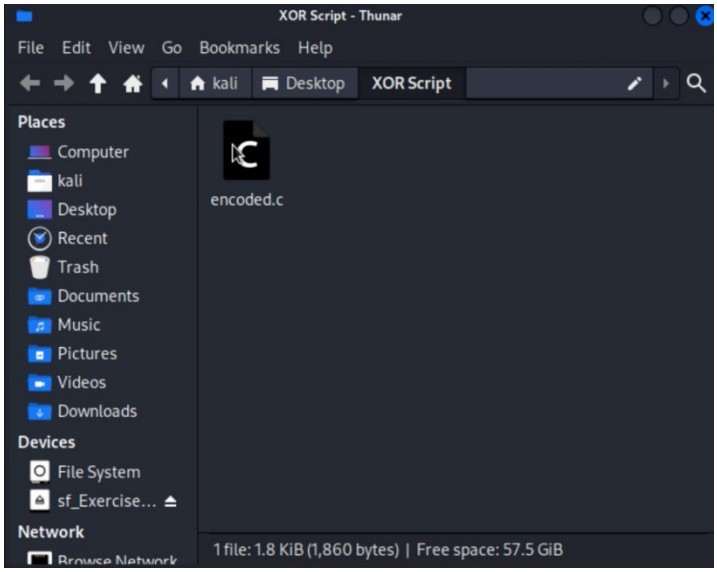

Fig. 9 Inside XOR folder

It is your job to compile this program to create an executable file. The following steps will guide you through the process.

3) At the top menu, double-click the terminal icon (Fig. 10).

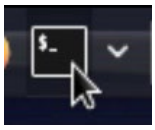

Fig. 10 Terminal icon

4) Once the terminal window opens, type the following:

cd Desktop/ *press enter*

cd XOR/ *press enter*

You are now inside the **XOR folder** and will be able to compile the program in the next step.

5) You are now inside the **XOR folder.** To successfully compile the **encoded.c** program, write the following in the terminal window:

gcc -O0 -oencoded encoded.c

If you compiled the program correctly, a new executable file should appear in the **XOR folder** (Fig. 11).

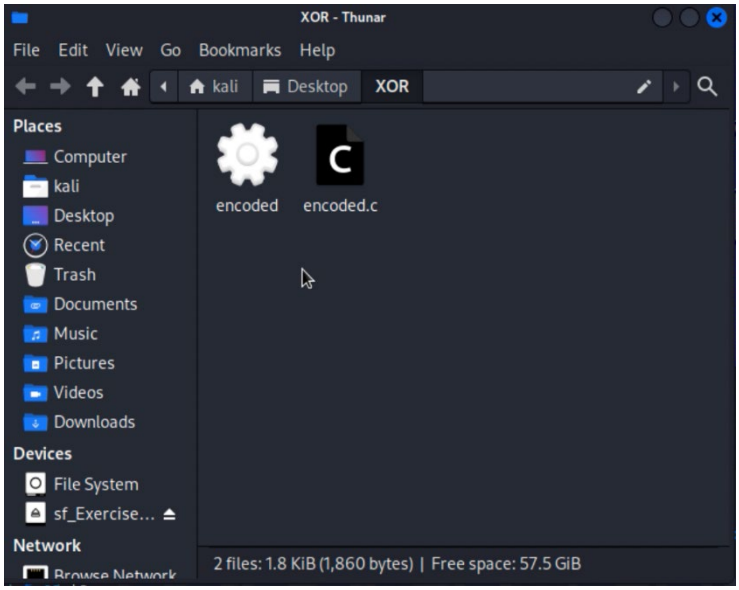

Fig. 11 Encoded executable

6) To execute the program, return to the terminal window (ensure you are in the correct folder) and enter the following command:

./encoded

You should see the prompt in Fig. 12.

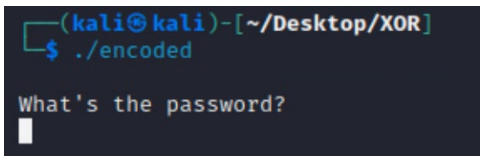

Fig. 12 Executable prompt

7) Enter your name as the password and press enter.

8) What is the program's output when you enter your name as the input?

_____

****

Next, you will be guided to obtain the correct password for the **encoded.c** program. To obtain the password, we will use Ghidra.

Ghidra offers a wide array of features and capabilities. In this phase, we will concentrate on **Ghidra's plugin scripts**. You will gain firsthand experience with the fundamentals of this tool.

****

9) On the desktop window, find the Ghidra Icon and double-click it to open and access the Ghidra software (Fig. 13).

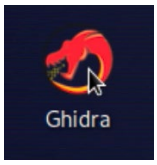

Fig. 13 Ghidra icon

10) Once Ghidra opens, go to the **File** tap, and double-click the **New Project** option (Fig. 14).

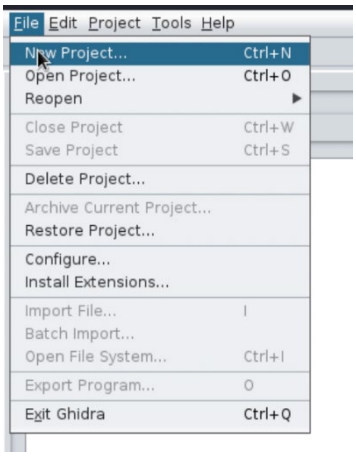

Fig. 14 New project Ghidra

11) The New Project window will pop up. Select the **Non-Shared Project** button and hit **Next** (Fig. 15).

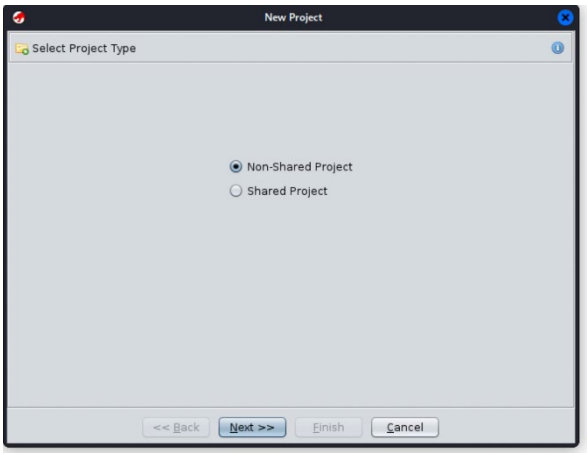

Fig. 15 Non-shared project Ghidra

12) Please keep the project in its suggested location and choose a suitable name for your project. A simple and relevant name for this Ghidra project would be “encoded” (Fig. 16). Please press **Finish** once you have given a name to your project.

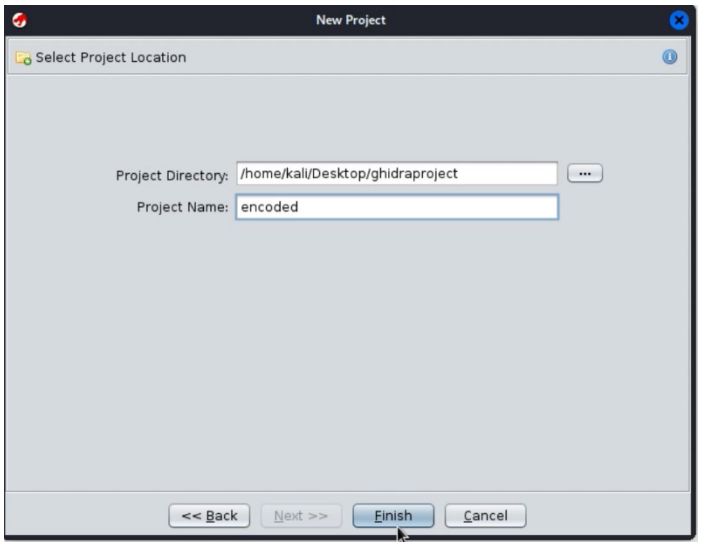

Fig. 16 Project location Ghidra

13) Now it is time to import the file into our encoded project. To do this, go to the **File** tab and click the **Import File** option (Fig. 17).

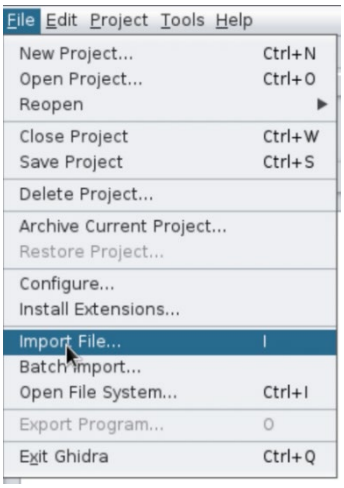

Fig. 17 Importing a file in Ghidra

14) Make sure you are in the **XOR folder** and select the **encoded** file (we need the executable, not the encoded.c file) Select the file and click the **Select File to Import** button at the bottom.

15) A window will pop up. Keep the preconfigured option unchanged (Fig. 18) and click **Ok.**

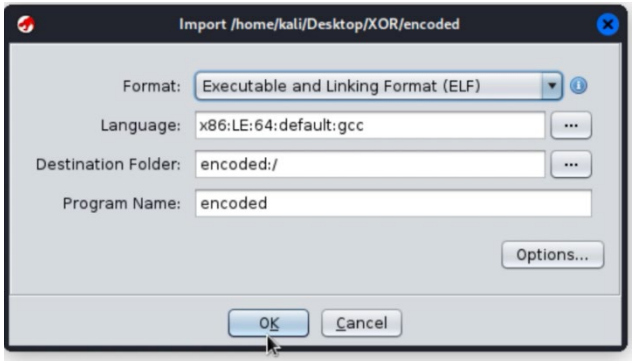

Fig. 18 Preconfigured options Ghidra

16) Wait for the file to be completely imported and click **Ok** on the Import Results Summary.

17) To access the highlighted “encoded” file, you have two options. You can either double-click it (Fig. 19) or click the dragon icon displayed in the Tool Chest menu.

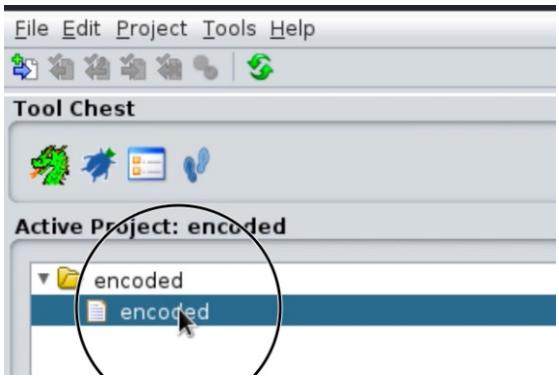

Fig. 19 Accessing Ghidra file

This process could take a minute. Be patient.

18) Click **Yes** on the Analyze window (Fig. 20).

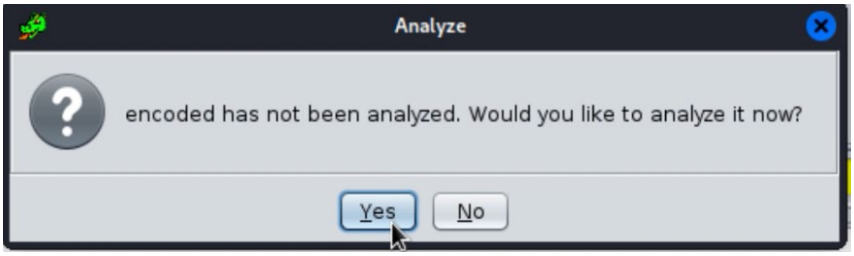

Fig. 20 Analyze Ghidra window

A second window named Analysis Options will appear, and without making any changes to the preconfigure options, simply press **Analyze.**

19) On the left side of Ghidra's main window, locate the **Symbol Tree** small window (Fig. 21).

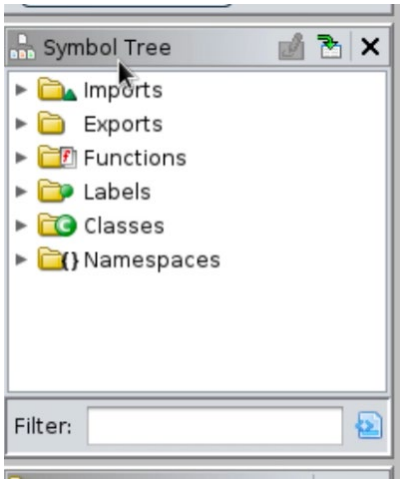

Fig. 21 Symbol tree window Ghidra

Our objective is to locate the **Key** that was used to encrypt the password in the **encoded.c** program. Spend a moment identifying a key word that will help you in discovering the answer. (Hint: Look at the Exports folder and the Labels folder for valuable clues.)

20) What is the Key that was used to encrypt the password for this program?

    _____

21) Now that you have found the Key, your next goal is to find the encrypted text. Use the knowledge you gained in step 19 and find a key word that will help you find the correct location.

22) At what memory address is the encrypted message located? (Hint: The memory address starts with 00.)

    Memory Address Location: _____

23) Now that you successfully have found the location holding the hidden **message**, you need to highlight the area as shown in Fig. 22.

Fig. 22 Message location Ghidra

24) To make things easier, we will convert the series of bytes into a readable string. Follow these steps: highlight the desired area, **right click**, choose the **Data** option, and then select the **string** option.

25) What are the first eight letters of the encrypted password?

    _____

****

Ghidra scripts are sets of code snippets written in Python or Java. These scripts automate various tasks and operations, allowing the users to extend Ghidra's functionality.

In this section we will work with the **XorMemoryScript**. The XOR Memory script automates the process of decrypting XOR-encrypted sections of memory and revealing the original content. This script is written in Java and is available in the list of preexisting scripts within Ghidra's collection.

****

26) In the top menu, click the **green play button** (Fig. 23).

Fig. 23 Green play button

This will open the **Script Manager** window (Fig. 24).

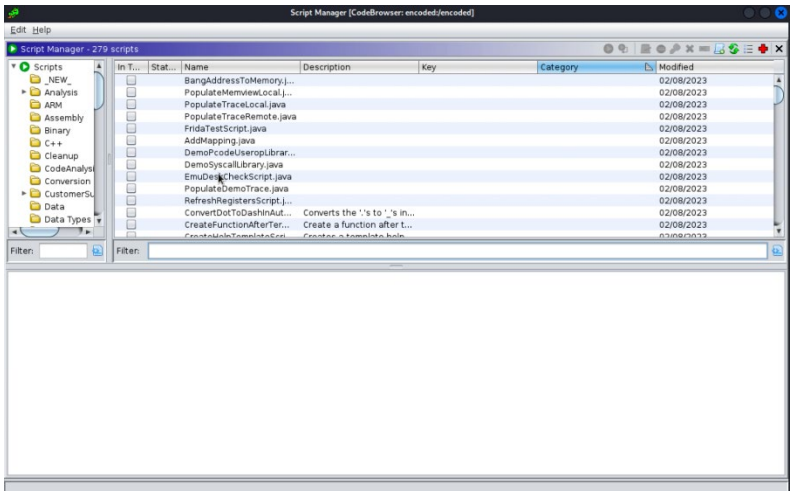

Fig. 24 Script manager Ghidra

As mentioned previously, the **XorMemoryScript** is part of the preexisting scripts in Ghidra.

27) In the **Filter** area, type **XOR** and the **XorMemoryScript** will appear (Fig. 25).

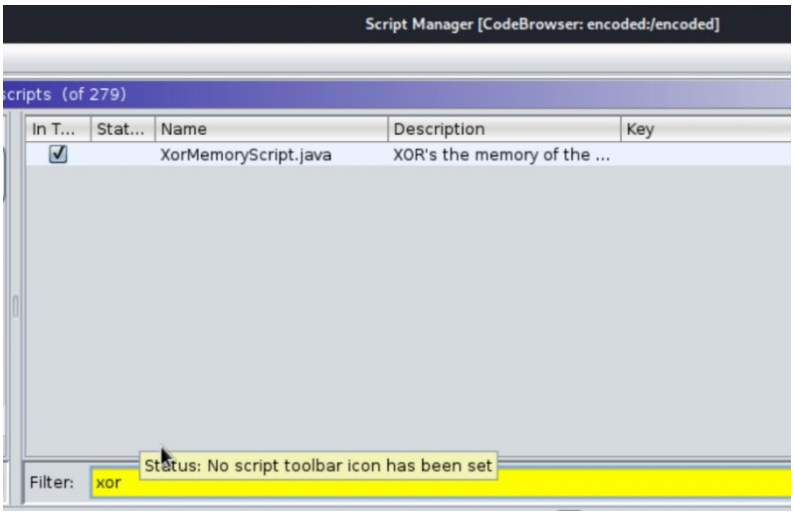

Fig. 25 XorMemoryScript Ghidra

28) In what programming language is the XorMemoryScript written?

_____

29) Now, let us uncover the encrypted password. Ensure that you have highlighted the **message portion** in the listing window. Next, follow the previous steps to access the **XorMemoryScript.** Double-click the **XorMemoryScript** and provide the **Key** you discovered in step 20 (Fig. 26).

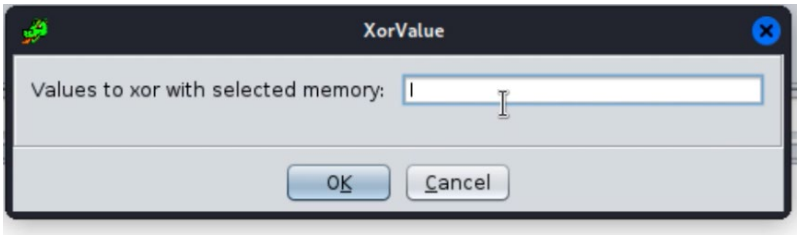

Fig. 26 XorValue Ghidra

30) If you have followed the steps correctly, you will find the unhidden password displayed in the Listing window. **What is the password?** (Hint: It should be in located in the same spot where you found the answer to step 25.)

_____

31) You can now execute the **encoded.c** program and provide the correct password as input. If you need a refresher on how to compile or run the program, please refer to steps 5 and 6.

32) **Once you input the correct password, what will be the output of the encoded.c program?** _____

Excellent work! You have successfully mastered the fundamentals of Ghidra and gained knowledge about how Ghidra scripts operate. You are now prepared to proceed with your mission of decoding the secret message you intercepted from Lord Voldemort.

4.3 Step 3: Encrypt Your Own Message in XOR

To gain a complete understanding of how the encoded.c binary operates, you will now have the opportunity to encrypt your own message, using the same logic this binary used. We recommend encrypting your own name for this exercise. Additionally, you will need to choose the key to be used for this encryption.

1) The **encoded.c** program will work perfectly for this section. You just need to make a few changes. You will begin by copying the program and changing its name (Fig. 27).

cp encoded.c testing.c

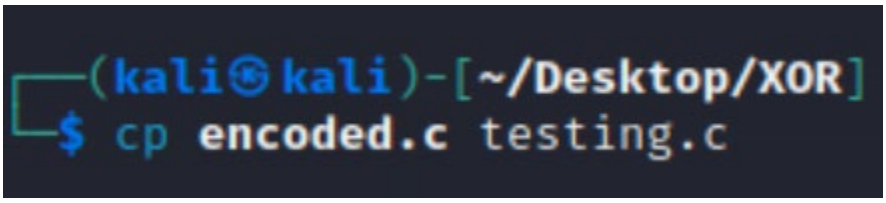

Fig. 27 Copying command Linux

This command will make a copy of the "encoded.c" file named "testing.c"
Make sure you are in the XOR folder.

2) In this section you will be working on the **testing.c** program. To open the program, right-click in the file and click the **Open With "Mousepad"** option (Fig. 28).

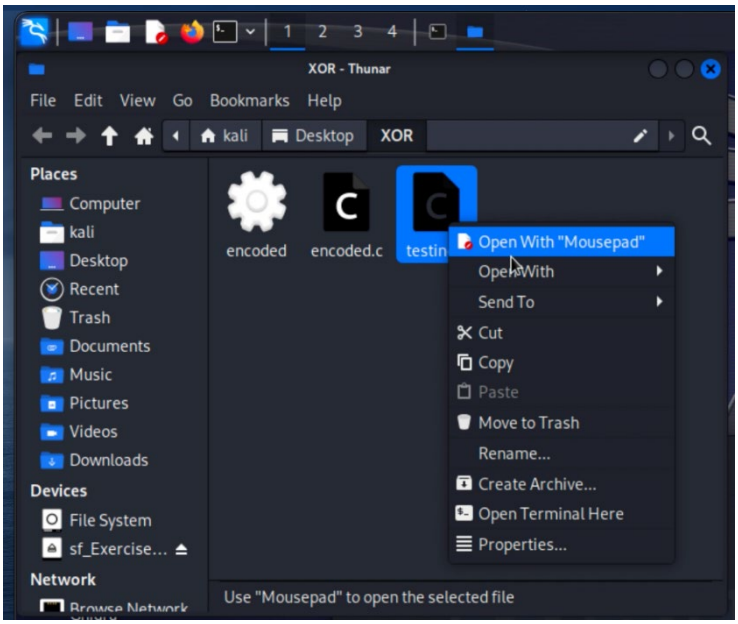

Fig. 28 XOR folder

3) Once the program is open in **mousepad,** you will see the window in Fig. 29.

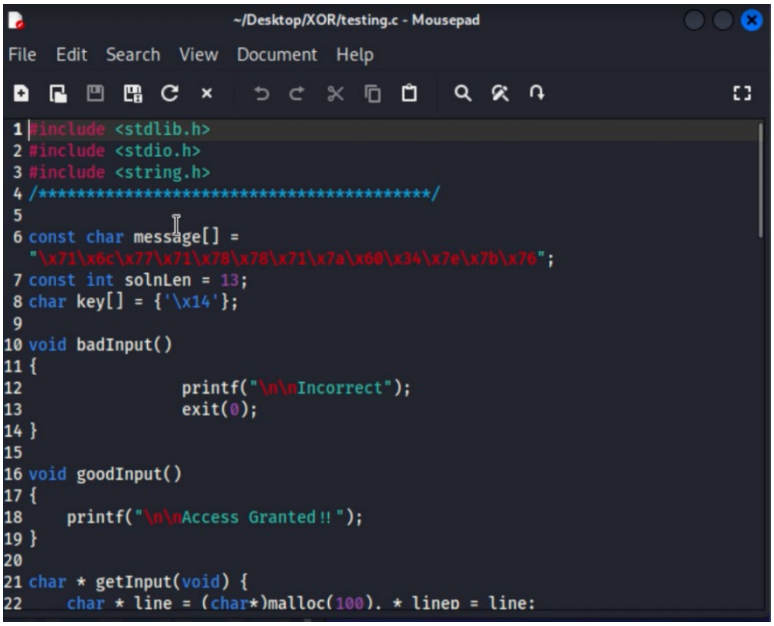

Fig. 29 Testing.c

Only the following parts need to be changed: the variables **message[], solLen, and key[].**

4) Take note that the **message** array consists of hexadecimal bytes. These bytes have already been encrypted using the XOR cipher.

5) You will need to first encrypt your name with a key of your choice. (We recommend using CyberChef for simplicity.)

6) Open your browser.

7) Navigate to the CyberChef application at the following URL:

https://gchq.github.io/CyberChef/

****

CyberChef is a user-friendly web application that facilitates a wide range of "cyber" operations directly within a web browser. **CyberChef** will allow us to encrypt any text in any basic or advanced encoding.

****

8) On the left-hand side of the website, you will come across various options. Your task is to locate the Encryption/Encoding option (Fig. 30) and click it.

Fig. 30 Encryption/encoding CyberChef

9) Scroll through the panel on the left side of the screen and find the **XOR** option (Fig. 31) and drag it to the "Recipe" area as shown in Fig. 32.

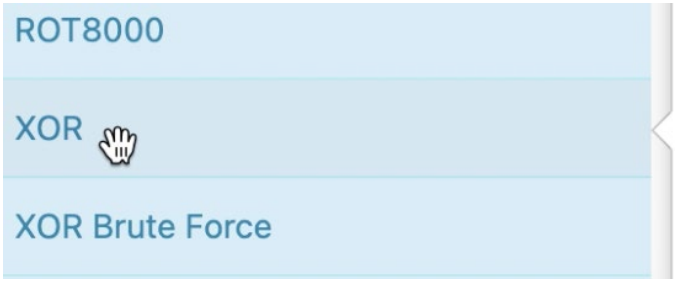

Fig. 31 XOR option CyberChef

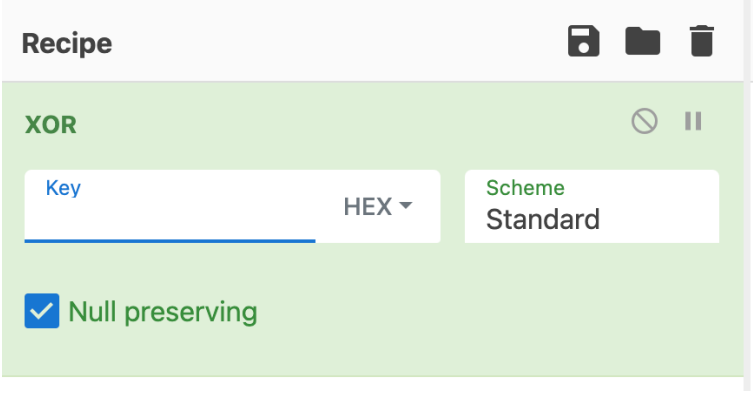

Fig. 32 Null preserving CyberChef

Ensure that the "Null preserving" option is chosen/selected.

10) Enter your chosen key in the **Key** box. This key will be used to encrypt your message using the XOR cipher.

    Write down your key below for future reference.

    Key: _____

11) In the **Input** area (Fig. 33), type the text you want to encrypt (name, random text, etc.).

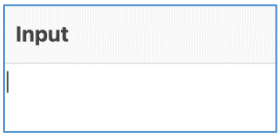

Fig. 33 Input area CyberChef

    Write down your text below for future reference.

    Text: _____

12) While you enter your text in the **Input** box, the **Output** area simultaneously displays the encrypted text using the provided **key**. Copy the text displayed in the **Output** area.

13) Now, you need to convert the ASCII text to hexadecimal bytes. To do this, you can search for an “ASCII to Hex converter” using your preferred search engine and use the website of your choice for the conversion.

14) After obtaining the hexadecimal bytes, you should copy the bytes and return to the **testing.c** program you have opened in **Mousepad**. Next, paste the converted hexadecimal bytes into the **message[]** array within the code. The bytes should be in the format \x75\x34 as shown in Fig. 34.

Fig. 34 Bytes format

15) Now count the bytes and change the length in the **solLen** variable (Fig. 35).

Fig. 35 solLen variable

16) In the **Key[]** array change the byte to the key you chose in step 10 (Fig. 36).

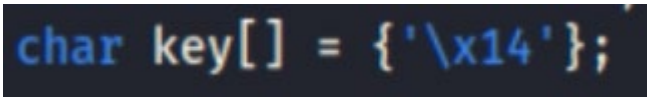

Fig. 36 Key variable

Remember this should be in hexadecimal notation. Please save your changes before proceeding to the next step.

17) Now it is time to see if you made the changes correctly. Go back to the terminal and compile the **testing.c** program and then run it (Fig. 37).

gcc -O0 -otesting testing.c

./testing

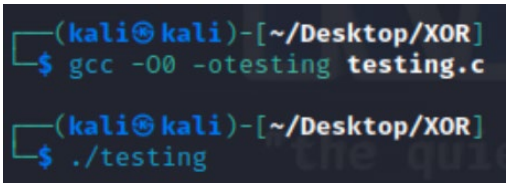

Fig. 37 Compiling testing.c

18) The program will prompt you to enter the password. After providing the encrypted text you just generated, you should observe the message **Access Granted!!** displayed, as illustrated in Fig. 38.

Fig. 38 Access granted output

If the program responds with the message **Incorrect**, it indicates that you need to review and revise your work for this section.

Congratulations! By successfully completing the exercise of encrypting your own message using the logic employed by the **encoded.c** binary, you have gained a comprehensive understanding of its operation. It is a remarkable achievement. Not only have you mastered the usage of tools like CyberChef for text encryption with the XOR cipher, but you have also acquired the knowledge of copying files in Linux using the terminal. This is an essential skill that will greatly enhance your command-line proficiency. Great job!

4.4 Step 4: Create Your Own Ghidra Script

You, Harry Potter, snuck into a hidden chamber and discovered Voldemort's computer. While exploring, you came across a strange folder named "Voldemort's Vault." Inside the folder, you found a peculiar binary file that demanded a password. You had a strong intuition that once you entered the correct password, it would reveal a very important message. Now, armed with the knowledge from steps 1 to 3, you are ready to create your own Ghidra script to suit your needs and decode the needed password.

1) To begin, please close all open applications. Once you are on the desktop, locate the folder called **Voldemort'sVault** and simply double-click it to gain access (Fig. 39).

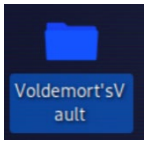

Fig. 39 Voldemort's vault folder

2) Within this folder, you will come across an executable file called **DarkSolemnity**. To execute it, open the terminal window by double-clicking on the corresponding icon located in the top menu, as illustrated in Fig. 40.

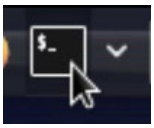

Fig. 40 Terminal window

3) Once you have accessed the **Terminal Window**, make sure you are in the correct folder.

cd Desktop/Voldemort\'sVault

4) Run the executable by typing the following:

./DarkSolemnity

5) The binary will prompt you for a **password** (Fig. 41). You are free to enter any password at this stage.

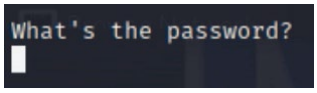

Fig. 41 Executable prompt

6) When an incorrect password is provided, the executable will produce an output. What exactly is the output message? _____

7) You now must access Ghidra (Fig. 42), **create a new project, and import the DarkSolemnity file.** (Hint: Refer to step 2 if you need some guidance.)

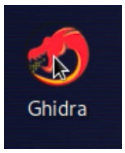

Fig. 42 Ghidra icon

8) After successfully importing and analyzing the **DarkSolemnity** file, your next objective is to locate the **Key** used for password encryption. Look in the **Symbol Tree** window located in the left-hand side (In case you need a refresher on locating the key within Ghidra, feel free to refer back to step 2 for guidance. Remember, it can serve as a helpful reference if you encounter any difficulties during the process.)

   Key: _____

9) Take a minute to find in the **Symbol Tree** window displayed in the left-hand side Now that you have found the Key, your next goal is to find the encrypted text. Use the knowledge you gained in step 2 and find a key word that will help you find the correct location.

10) At what memory address is the encrypted password located? (Hint: The memory address starts with 00.)

    Memory address location: _____

****

Similar to the previous binaries we encountered in steps 2 and 3, this binary operates under the same logic. To facilitate the workshop, a prefilled Ghidra script template has been provided, containing a few **TODO** tasks. These tasks will enable you to gain a deeper understanding of how Ghidra scripts function.

****

11) Go back to the desktop and find a folder named **Script Template.** Double-click the folder to access it (Fig. 43).

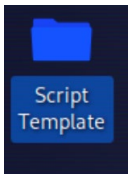

Fig. 43 Script template folder

12) Once you have accessed the folder, you will find a **template.py** file (Fig. 44). Double-click the program to open it.

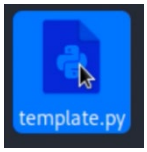

Fig. 44 Template.py

13) Now, right-click and choose the option **Select All**. You can also do this by pressing Ctrl+A**.** Copy the whole text by pressing Ctrl+C as shown in Fig. 45.

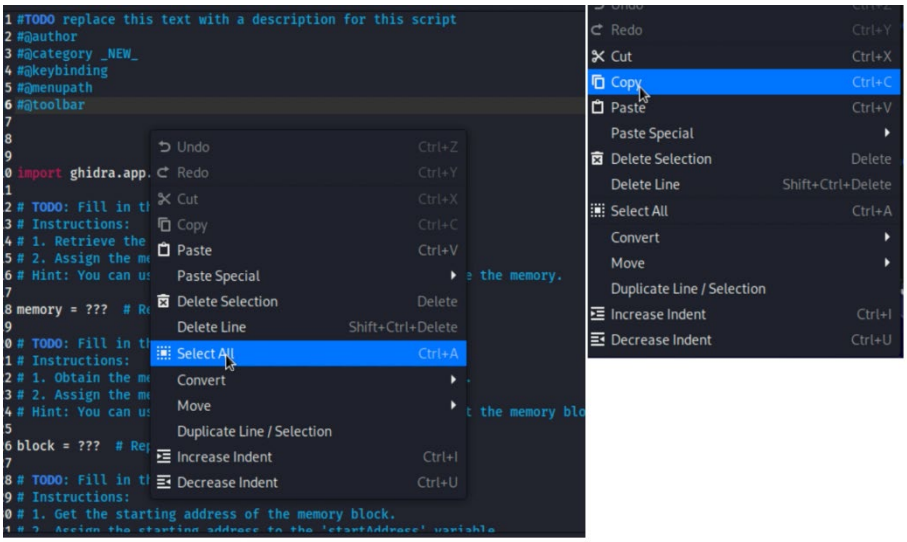

Fig. 45 Copy text

14) You are now ready to create your own Ghidra script. To do so, access the **Script Manager** in the top menu inside **Ghidra**.

15) Once the **Script Manager** window is opened, you need to click the **Create New Script** button in the top menu as shown in Fig. 46.

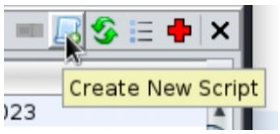

Fig. 46 Create new script Ghidra

16) Select **Python** and press Ok (Fig. 47)**.**

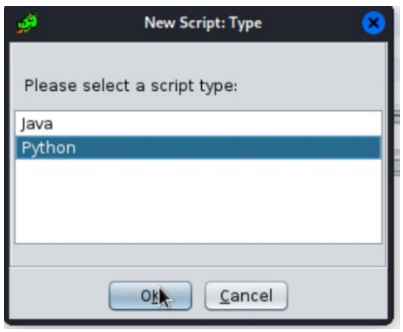

Fig. 47 Script type

****

The objective of this workshop is to equip you with the necessary knowledge and skills to create and interact with scripts in Ghidra effectively. To simplify the exercise, here is a helpful hint**: the password has been encrypted using a Caesar Cipher encryption method.**

****

17) The script will be saved in the default directory. Please name your script accordingly (Fig. 48), such as **CaesarCipherDecoder.py**, which would be a suitable choice. Click ok to continue.

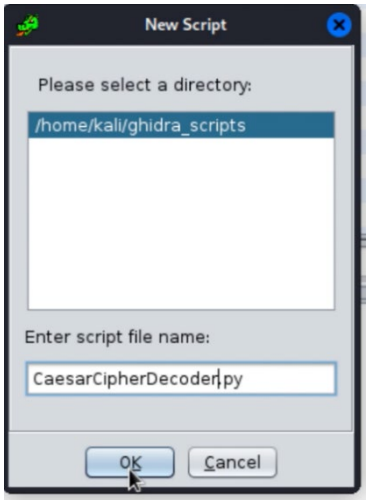

Fig. 48 Name script

18) Now you can paste the script template you copied in step 13 into the **CaesarCipherDecoder.py script** (Fig. 49)**.**

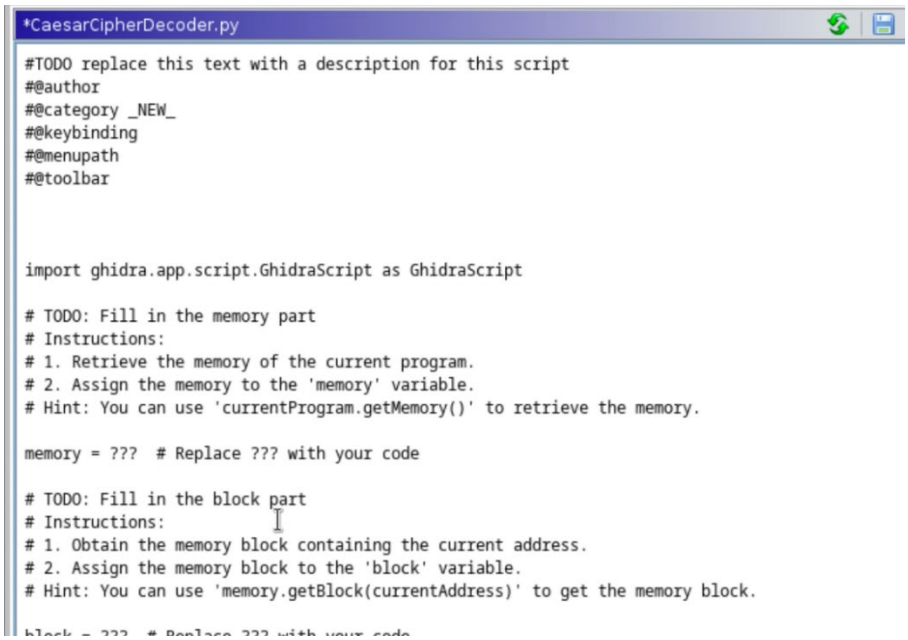

Fig. 49 CaesarCipher script

19) As you examine this template, you will come across a few **TODO tasks**. We encourage you to spend a few minutes completing these tasks, as they are designed to familiarize you with essential functionalities of the Ghidra API. Doing so will enhance your understanding of the API and its capabilities.

20) After completing the **TODO** tasks, remember to save your work by clicking on the **Save** button located in the top-right corner of the menu (Fig. 50).

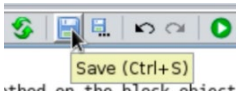

Fig. 50 Save button

You can now close the script manager.

21) You are now ready to test your script. Please go to the **password** location and highlight the area as shown in Fig. 51.

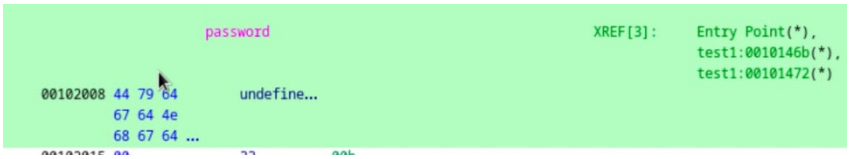

Fig. 51 Password location Ghidra

22) We want to convert the bytes into a readable string. To do this**,** right-click the highlighted area, choose the **Data** option, and select the **String** option as shown in Fig. 52.

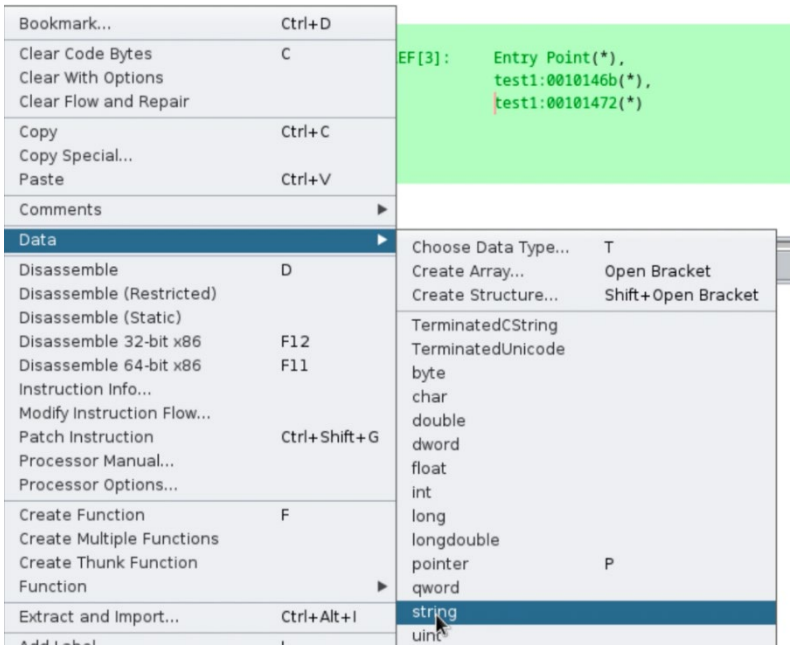

Fig. 52 Data string Ghidra

23) Once you have completed the previous instruction, **what string does the password display**? _____

24) In the top menu, click the green play button to open the **Script Manager** window.

25) Once the **Script Manager** window is opened, in the **Filter** area, type **Caesar** and the **CaesarCipherDecoder** will appear (Fig. 53).

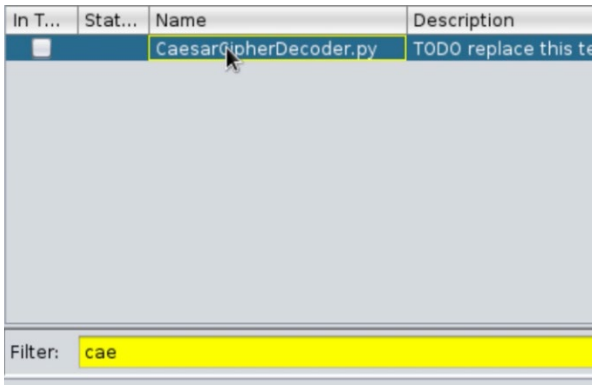

Fig. 53 CaesarCipherDecoder

26) Now, make sure you have the **password** portion highlighted in the listing window. Next, double-click in the CaesarCipherDecoder and provide the **Key** you discovered in step 8 (Fig. 54).

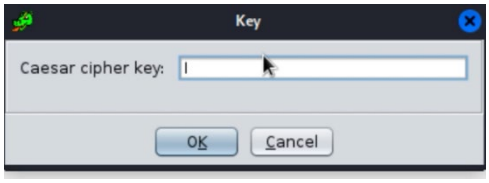

Fig. 54 Caesar key Ghidra

27) If you have followed the steps correctly, you will find the unhidden password displayed in the **Listing window**.

What is the password? _____

28) You can now execute the **DarkSolemnity** binary in the terminal window and provide the correct password as input.

29) Once you input the correct password a message will be output.

To which location are Death Eaters summoned to meet Lord Voldemort? _____

Congratulations! If you have followed the instructions correctly, you have successfully uncovered the sinister plans of Lord Voldemort to attack Hogwarts. Along this journey, you have not only created your own Ghidra script but also gained hands-on experience with basic Linux commands. Furthermore, you have delved into the world of encryption and discovered why ciphers play a crucial role in safeguarding personal information from unauthorized access. Moreover, you have firsthand experience in understanding the weaknesses of certain ciphers by exploiting their vulnerabilities.

By reaching this point, you have demonstrated your proficiency in using Ghidra, familiarity with Linux commands, and understanding of encryption and cipher vulnerabilities. These skills will undoubtedly empower you to tackle future challenges in the realms of reverse engineering, security, and information protection.

Well done on your accomplishments!

5. Conclusion

This report provides a basic learning module that introduces participants to binary analysis, including the cases when basic encoding is in place. Participants are exposed to Ghidra its scripting system, and some examples of how these can be used for automated decoding and analysis. This hands-on exercise allows the participant to gain a fuller understanding on how Ghidra can be used in different scenarios that will fit their own needs. In a world where digital security and reverse engineering are crucial, the knowledge gained from this module is valuable. Participants are now better equipped to tackle complex binary challenges, decode tricky codes, and navigate digital systems more confidently. As they continue their journey in binary analysis, we hope these skills will help them make meaningful contributions to the field of cybersecurity and beyond.

6. References

1. What is encryption? Cloudflare, Inc.; c2023 [accessed 2023 Apr]. https://www.cloudflare.com/learning/ssl/what-is-encryption.
2. Fox N. How to use ghidra to reverse engineer malware. Varonis; c2023 [accessed 2023 Apr]. https://www.varonis.com/blog/how-to-use-ghidra.
3. National Security Agency. Ghidra is a software reverse engineering (SRE) framework. GitHub; c2023 [accessed 2023 May]. https://github.com/NationalSecurityAgency/ghidra.
4. VirtualBox. VirtualBox; c2023 [accessed 2023 Apr]. https://www.virtualbox.org/.
5. Kali, Kali 2023.2; c2023 [accessed 2023 Apr]. https://www.kali.org/blog/kali-linux-2023-2-release/.
6. Ghidra, Ghidra 10.3; c2023 [accessed 2023 Apr]. https://ghidra-sre.org.
7. CyberChef. Crown; c2016 [accessed 2023 Apr]. https://gchq.github.io/CyberChef/.
8. Acosta C, Clarke L, Medina S, Akbar M, Hossain MS, Free-Nelson F. Repeatable experimentation for cybersecurity moving target defense. In: Garcia-Alfaro J, Li S, Poovendran R, Debar H, Yung M, editors. International Conference on Security and Privacy in Communication Systems; 2021. Springer, Cham; c2021. p. 82–99.

List of Symbols, Abbreviations, and Acronyms

API	application programming interface
ARL	Army Research Laboratory
CIT	Collaborative Innovation Testbed
CyberRIG	Cyber Rapid Innovation Group
DEVCOM	US Army Combat Capabilities Development Command
NSA	National Security Agency
URL	Uniform Resource Locator
VM	virtual machine

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
1 (PDF)	DEVCOM ARL FCDD RLA ND J ACOSTA