Reverse Engineering Labs - Folgearbeit

Bachelor Thesis

Department for Computer Science
OST - Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Semester: Spring 2023

Authors:	Gianluca Nenz
	Ronny Müller
	Thomas Kleb
Project Advisor:	Ivan Bütler
Expert:	Benjamin Fehrensen
Reviewer:	Markus Stolze
Release:	E-Prints
Version:	Friday 16th June, 2023

Department for Computer Science
OST Eastern Switzerland University of Applied Sciences

Abstract

Background This bachelor thesis is based on a previously created "Reverse Engineering Lab" term project by the team, which consists of beginner level hands-on exercises (challenges) for students at OST to get into software reverse engineering. However, some important aspects were not covered in the previous lab. This bachelor thesis is geared towards advanced reverse engineering in order to go deeper and into in-depth reverse engineering techniques.

Purpose This bachelor thesis extends the existing "Reverse Engineering Lab", adding 10 more complex practice labs and exercises by introducing new reversing methods, tools, and frameworks. The new and advanced exercises can then be used by the teachers at the OST to lecture on the subject of reverse engineering. This gives the students a better insight into the subject and a more enriched, practical, and hands-on experience.

Methods First, we created a collection of topics not yet covered in the previously created "Reverse Engineering Lab". These topics were then evaluated by the team and the advisor based on personal interest, usefulness, and importance in the field of reverse engineering. This evaluation was used to discuss which topics we should create challenges for. During the semester we used Scrum to iteratively create the challenges. Whenever a challenge was finished, it was tested by us, fellow students, and other volunteers. This process ensured the high quality of the challenges.

Results The goal of this bachelor thesis, the creation of 10 new reverse engineering challenges covering new methods, tools, and frameworks, was successfully achieved. All the challenges are hosted on Hacking-Lab, an online platform for cybersecurity training and ethical hacking. Hacking-Lab provides students with everything they need to improve their reverse engineering skills.

Conclusions The aim was to teach students techniques that would reveal potential attack vectors. The final product is a collection of many advanced reverse engineering topics, providing deeper insight and teaching problem-solving skills.

Legal disclaimer This course aims to understand hacking methods in order to effectively counter them. It is unethical and potentially illegal to use the knowledge gained for malicious purposes. This course promotes responsible use with an emphasis on digital security and protection.

Lay Summary

Overview

What is Reverse Engineering

As described in our term project, reverse engineering is the process of analysing a product or system to understand how it works, how it was made, or how it can be improved. It involves taking the product or system apart, examining its components, and understanding how they fit together and interact. In the context of software, reverse engineering is the process of analysing a computer program to understand how it works and how it was implemented. This may involve disassembling the program, studying its code and documentation, in order to recreate or modify it. Reverse engineering can be done for a variety of reasons: to learn about new technologies, to fix flaws / security vulnerabilities, or to create competing products. In most cases, reverse engineering is a challenging and time-consuming process, requiring a deep understanding of the underlying technologies and systems. It is often used by experts in fields such as computer science, engineering, and security.

Current Situation

For a computer scientist, it is always useful to have some knowledge of cybersecurity subjects. In order to introduce students to the world of cybersecurity, the Ostschweizer Fachhochschule (OST) has implemented several modules such as "Cyber Security Foundations", "Secure Software", "Cyber Defence" and "HackLab". In these modules, students use the Hacking-Lab platform to solve hands-on exercises (challenges). In the preliminary work, an introduction with challenges on the basics of reverse engineering was created. The plan is to extend the current state with reverse engineering challenges about new tools, frameworks and techniques. The goal of them is to bring students closer to the subject and explain more advanced aspects of reverse engineering.

Approach

To achieve this, new challenges will be added to Hacking-Lab OST environment, which is, as mentioned above, a platform which students are already accustomed to. These challenges will be created for the students to go through and will be built with the idea of future additions in mind.

Procedure

The scope defined at the beginning of the project was the basis from which the challenges were created. This scope includes the topic, the know-how to be taught, and the tools to be used by the students to complete the tasks. In addition to these points, the platform on which the students should work is defined. In order to solve the given tasks, the students need instructions to follow.

Technologies

The challenges are created for either the Windows or Linux operating system, depending on the software required. This allows students to have the option to complete each challenge on either system, using a virtual machine if necessary. All the challenges are hosted on a Hacking-Lab tenant provided by the advisor, firstly on the demo tenant to test all the features and set up, then on the OST tenant for official use. Hacking-Lab is a website that offers a range of cybersecurity services, including training, simulations, and challenges. It is designed for cybersecurity professionals, as well as students and enthusiasts who are interested in learning about and improving their skills in the field. Because of this, the OST uses it to host various exercises to teach the fundamentals of cybersecurity to its interested students.

Results

The goals were defined during the inception phase of the project. It was planned to have 8 - 10 challenges by the end of the semester. This requirement was fulfilled thanks to a strict plan and coordination between the students. During the project, 10 challenges were created and uploaded to Hacking-Lab.

Future

The challenges created in this project, combined with the challenges of the preliminary work, bring the students to a high level of understanding in the topic of reverse engineering. These challenges combined with the feedback promise to be a suitable series to introduce the students to advanced reverse engineering.

Acknowledgement

First of all, we would like to express our sincere gratitude to Ivan Bütler for allowing us to continue our term project. We thank him for his guidance and advice throughout the project.
Second, we are sincerely thankful to our fellow students and other volunteers who graciously dedicated their time to test our product. Their feedback and unique perspectives greatly improved the quality and robustness of the challenges.
We would like to thank Compass Security for providing Hacking-Lab as the hosting platform for the challenges.
Finally, we are grateful to the OST for the opportunity to undertake this project. The school's resources have been fundamental to our intellectual growth and the successful completion of this project.
In conclusion, this bachelor thesis would not have been possible without the help and support of these individuals and institutions, and we are deeply grateful for their contributions.

Contents

Glossary

Term	Description
Angr	An open-source binary analysis platform for Python.
Basic Block	A code sequence with no branches [1].
Binary	A pre-compiled, pre-linked program that is ready to run under a given operating system; a binary for one operating system will not run on a different operating system.
Buffer Overflow	Vulnerability where excess data overflows into adjacent memory.
Challenge	Name for a single hands-on exercise on the Hacking-Lab platform.
Docker	A platform for developing, shipping, and running applications using containerization.
Flag	In cybersecurity, a piece of data that proves a challenge was successfully completed.
Frida	Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers.
GDB	The GNU Debugger, a Linux tool for dynamically debugging software programs.
IDA	An interactive disassembler and debugger, used for software reverse engineering.
Indirect jumps	Assembly jump to address in a register instead of directly to an address.
Opaque predicates	Expression which evaluates true or false and is hard to understand at first glance.
Path explosion	Exponential growth of program execution paths with increased program size.
Scrum	An agile framework for managing complex projects.
Scylla	Tool to dump (parts of) a process' memory.
Term Project	The project done one semester before the bachelor thesis.
Tenant	A unique subdomain in a shared web environment.
Unflattening	The process of deobfuscating Control Flow Flattening.
Unified method signatures	Unify function signatures to a single signature.
Ghidra	A software tool for reverse engineering and static malware analysis.
x64dbg	An open-source x64/x32 debugger for Windows.

Acronyms

API	Application Programming Interface
BB	Basic Block
DLL	Dynamic Link Library
CFF	Control Flow Flattening
CFG	Control Flow Graph
GDB	GNU Debugger
GUI	Graphical User Interface
OST	Ostschweizer Fachhochschule
UUID	Universal Unique Identifier
ROP	Return Oriented Programming
RVA	Relative Virtual Address

Part I

Technical Report

Chapter 1

Project Idea

1.1 Introduction

The project team consists of three students. Two students major in cybersecurity and one majors in software engineering. In addition, an advisor, an expert, and a reviewer are a part of the process. The basic idea of this project is to create a series of hands-on exercises (challenges) to teach students at the OST about various topics of software reverse engineering. This documentation with the challenges is the main product of this bachelor thesis. Hands-on training helps to put theory into practice. This allows students to apply their knowledge and perform tasks which could be relevant for their future working lives.

1.2 Problem statement

The term project covered the basics to learn reverse engineering. The aim of the term project was to provide a suitable introduction to the vast field of reverse engineering. Therefore, many of the foundations for this work were already laid. Within the term project, a problem domain overview (Figure 1.1) was created, which represented the planned content.

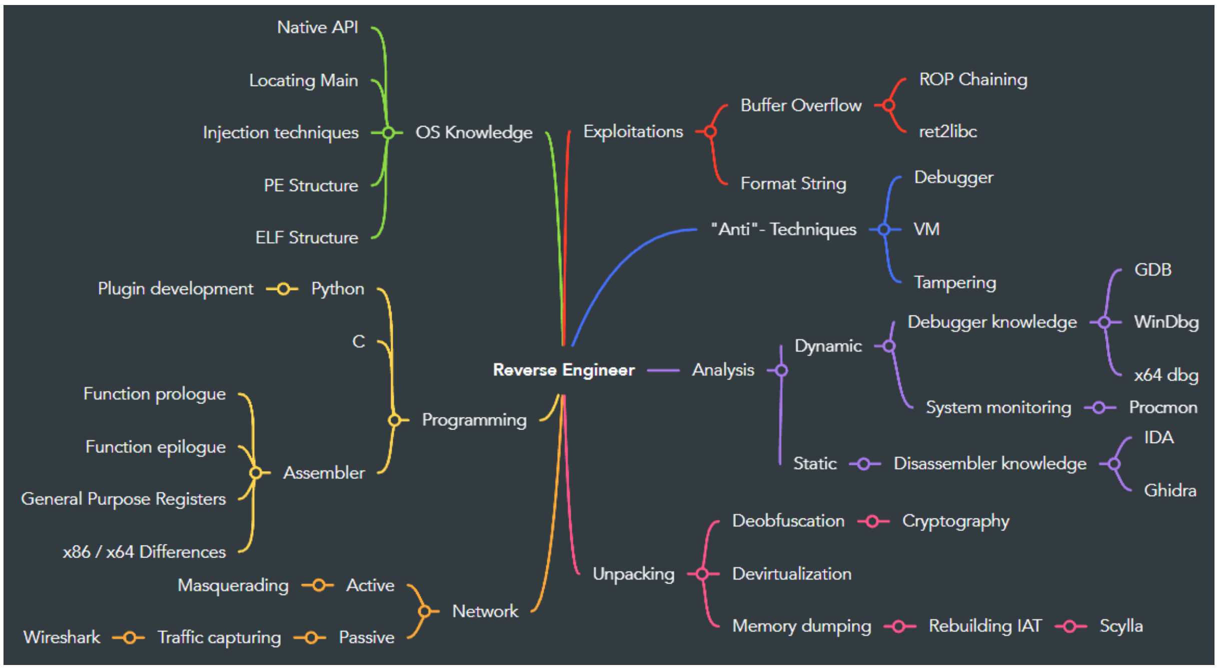


Figure 1.1: RE Domain from preliminary work

A large part of this graph was covered by the term project and challenges were created for it. Topics covered include the programming path and the entire analysis path. Buffer overflows were

also covered. A few topics were left open, although they could also be of relative importance to a reverse engineer, such as ROP chaining, anti-techniques, and unpacking.
Because the field is so broad and there are still many important and exciting topics, an extension of the term paper as a bachelor thesis was considered. This creates a need for this bachelor thesis to explore these techniques and teach them to students.

1.2.1 Delimitation

Of course, the product of this work is not the only course to learn reverse engineering. But these courses usually cost money and therefore cannot be used by our supervisor for teaching at the OST. Thus, this product differs not only in content, but also in use and purpose.
Examples of such courses would be Infinite Skills' Udemy course [2] or Josh Stroschein's Skills course [3].

1.3 Objectives

The task of this bachelor thesis was very openly defined. Basically, the aim was to expand on the topics covered in the preliminary work.
The concise objectives were defined by the team and the advisor in the first advisor meeting. The defined objective was to create between eight and 10 challenges on advanced reverse engineering topics. Further details on what should be taught in these challenges would be defined after some ideas have been brainstormed and ranked. This is explained in section 1.5.

1.4 Methodology

Based on the experience gained in the term project, this bachelor thesis should follow the same workflow to fulfil the objective. Scrum will be used with weekly sprint meetings and occasional advisor meetings for questions and to update the advisor. Testing the challenges is used to confirm that they will able to teach a concept. In addition, the challenges will have to follow the requirements mentioned in chapter 4.

1.4.1 Conditions

The bachelor thesis is worth 12 ECTS. As a rule of thumb, 30 hours of work are required per credit. This means that each team member has to invest about 360 hours in this project. These hours combined point out that more than 1000 working hours will be spent on the project, see chapter 8 for more information.

1.5 Expected Outcomes

As mentioned in section 1.3 the team created a list with well known topics in reverse engineering. Each member of the team and the advisor then individually rated the topics from one to five. This resulted in a score of up to 20. The full list can be seen in the appendix in Figure F.1. Table 1.1 shows a condensed version.

Type	Score	Platform	Description
Ghidra Introduction	20	Windows	Installation and orientation inside Ghidra
Anti-Techniques	20	Windows and Linux	Introduction to different anti-techniques, such as Debugging, Tampering, or Hooking
Obfuscation	19	Windows and Linux	Show and reverse multiple advanced obfuscation techniques
ROP Chaining	18	Linux	Use of ROP chaining to perform function calls within a binary
Combination Challenge	18	Windows	Final challenge as wargame as kind of exam to test if the student learned something
Ghidra Plugin	17	Linux	Create a custom plugin in Ghidra to use for further challenges
Injection Techniques	17	Linux	Teach a type of code injection and how to do it
Ghidra AI Plugin	14	Linux	Use of AI to help with the static analysis of a binary in Ghidra

Table 1.1: Challenge Concepts

This list is not yet arranged to support a common thread. These topics should then be converted into a challenge. The chosen sequence will be visible in chapter 5. To measure the effectiveness of the challenges, surveys of testing participants will be conducted. We hope that the challenges will teach them the defined concepts and help them to recognise them in practice.

1.6 Conclusion

The defined topics are also unknown to the team, so the team has to plan enough time to research these topics and come up with a good way to teach them. However, there should be no possibility of not being able to cover a topic adequately, as the team can always fall back on the advisors experience. The main goal is to create 8 to 10 challenges on the topics mentioned in Table 1.1.
The lessons learned from the preliminary work are applied to this bachelor thesis. The testing process, especially when filling out the form, was too inconsistent, which prevented a detailed evaluation in the documentation. It is therefore a key objective to strictly adhere to this process. The documentation was also the weakest part of the term project, and therefore it is a goal not to repeat the same mistakes and to try to improve the quality of the documentation. At the beginning of the term project, time tracking was not clearly regulated. This led to inconsistencies in the evaluation, which had to be corrected at the end. In addition, the distribution of weekly working hours among the team members was not uniform. Often there were weeks in which one of the team members could not reach the weekly working hours due to too little work, but then had to try to make up for this in the following weeks by doing more work. The goal is also to distribute these times more evenly using the same time tracking tool.

Chapter 2

Tools and Frameworks

2.1 Ghidra

Ghidra is an open source suite of tools for reverse engineers. It was developed and released in March 2019 by the National Security Agency to support its cybersecurity mission. The program is written in Java, but the integrated decompiler is implemented in C++. It can be used on all major platforms (Windows, MacOS and Linux). Ghidra can disassemble, assemble, decompile and much more. Ghidra also supports many processor instruction sets and executable formats. It also allows users to develop their own plugins or scripts to help them in their workflow. [4] [5]

2.2 OpenAI API

The OpenAI Application Programming Interface (API) is used for the ”Ghidra GPT” challenge, which uses it to help a student to reverse engineer a binary. This API was chosen for the project because of its current relevance in many areas of computer science. All the information needed to integrate it into a challenge can be found either in the API reference [6] which explains how to use the API, or in the general documentation itself. [7]

2.2.1 Models

Models for the OpenAI API are advanced artificial intelligence language models that have been trained on large amounts of text data. Developers can use the API to access these models, which can perform a variety of tasks such as content generation, summarisation, translation, question answering, and more. A list of all available models can be found in the documentation. [8]
The model used for text generation in this project is ”gpt-3.5-turbo”. It uses a message-based system rather than a single input prompt, unlike other models such as ”text-davinci-003”. Each message consists of two parts: the content and a role (either ”system”, ”user” or ”assistant”). A ”system” message controls the behaviour of the model, while a ”user” message acts as an instruction or query. The model playing the ”assistant” role generates responses based on the context of the conversation. GPT-3.5-turbo maintains context, adapts to user instructions, and produces more coherent and relevant output by taking into account the entire message history.

2.2.2 Tokens

OpenAI API tokens are units of text that are processed and generated by the language models. For the OpenAI models, tokens are: "common sequences of characters found in text. The models understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens" [9]. They can be as short as a single character or as long as a word, and include letters, numbers, symbols, and spaces. When using the OpenAI API, both input and output tokens count towards the usage. Token limits play a crucial role in determining the model's ability to process a query, as each model has a maximum token capacity. In addition, tokens are the basis for billing, as the cost of an API call is determined by the total number of tokens involved.
To illustrate how tokens are processed using the API, visit OpenAI's tokenizer [9]. This shows that the sentence "Hello World!" consists of three tokens: "Hello", " World" and "!".

2.3 x64dbg

x64dbg is an open-source binary debugger for Windows, aimed at both beginners and experts. It was developed by Duncan Ogilvie (mrexodia on github) with the primary aim of providing an efficient and flexible debugging environment. This debugger is coded in C++ and offers an interactive, dynamic and user-friendly Graphical User Interface (GUI). It supports both 32-bit and 64-bit systems, making it widely applicable across a range of platforms. x64dbg offers a wide range of features such as disassembling, debugging and graphing executable code, as well as the ability to create custom scripts or plug-ins. It's user-driven development model allows for enhancements that are regularly implemented based on community feedback. All these features make it an important tool for a reverse engineer, which is why it was chosen for this project. [10] [11]

2.4 GDB

GNU Debugger (GDB), like x64dbg, is a very versatile open-source debugger that helps developers analyse errors during program execution. GDB was first released by Richard Stallman in 1986 as part of the GNU system. Written in C and C++, it is compatible with many programming languages, including C, C++, Rust, Go and others. It is designed for the Linux operating system and can be used via text commands in a shell or terminal environment. As well as performing typical debugging tasks such as setting breakpoints, stepping through code and examining variables, GDB also supports reverse debugging, a rare feature among debuggers. As GDB is an established tool in the reverse engineering world, reverse engineers have to know how it is used. [12] [13]

Chapter 3

Hacking-Lab

3.1 Overview

Hacking-Lab is an online platform that provides a virtual environment for users to learn and practice various cybersecurity skills. One of the key benefits of this platform is its flexibility, as users can choose from a variety of systems and technologies that focus on different aspects of cybersecurity. This allows teachers to tailor their lessons to specific topics or skills. Hacking-Lab also offers a comprehensive set of tools and resources for teachers, including pre-built challenges and assessments, as well as the ability to create custom challenges and track student progress. All of these benefits make Hacking-Lab an ideal playground for students who are interested in the vast play field of cybersecurity. This is why the OST, in collaboration with our advisor, decided to create a tenant for the various cybersecurity lectures.

3.2 Tenants

When working with Hacking-Lab, a tenant is a separate environment for hosting labs and challenges. For this project, two of such tenants are of note: The OST tenant[1], which is the platform used by the OST to host its labs and challenges for the different modules, and the demo tenant[2], which is used to test and deploy initial iterations. The challenges created in this project were hosted on the demo tenant, which is explained in subsection 3.2.2.

3.2.1 OST Tenant

Teachers of cybersecurity subjects in the OST use this tenant to upload challenges and create graded exercises. It is currently used in the following modules: ”Cyber Security Foundations”, ”Secure Software”, ”Cyber Defence” and ”HackLab”. More information on these subjects can be found on the OST overview for main areas of study. [14]
Access to this tenant requires an OST account or a global Hacking-Lab single sign-on (SSO). [15]

3.2.2 Demo Tenant

The Hacking-Lab team uses the demo tenant to develop and evaluate various labs and challenges before making them available on a public tenant. In addition to testing the challenges, this tenant is also used to experiment with new features and user interface designs for the underlying platform. As a result of these frequent updates, the platform can experience unavailability and connectivity issues, making a testing environment like this essential.
For this project, the challenges were created and hosted on the demo tenant. This approach ensures that any issues or bugs can be identified and resolved before the challenges are moved to the OST tenant. This not only helps to maintain the quality of the challenges but also ensures the overall stability and reliability of the platform.

[1]OST tenant: https://ost.hacking-lab.com/
[2]Demo tenant: https://demo.hacking-lab.com/

3.3 Subdomains

The tenants in the lab structure have subdomains that serve different purposes. These subdomains can be accessed by adding a prefix to the URL. In this project, both the ”editor” subdomain (with the prefix ”editor”) and the ”resource” subdomain (with the prefix ”res”) were used for editing labs and challenges. These are explained in more detail in the following sections.

3.4 Structure

This section explains the key structural elements of the Hacking-Lab platform. Understanding these aspects will enable users to effectively create and manage labs and challenges within the platform.

3.4.1 Macro Overview

To understand how a Hacking-Lab editor can create and maintain different labs on the platform, one has to understand how Hacking-Lab is structured and how the tools explained in section 3.3 are used.
The most important part of understanding the Hacking-Lab structure is to know how it is built: Each of the different tenants is represented as a GitHub repository. This means that a lab for a tenant can either be created directly in the repository or by using the ”editor” and the ”res” prefix as an interface. This structure has to be in mind when creating a lab, as the editor interface does not yet allow for deletion of a lab (this must be done directly on the repository).

3.4.2 Lab and Challenge Structure

These different parts of the platform are accessed through different roles. When an account is created, it is given the default role of ”Student”. To get more UI access to the platform, an admin has to change this role. For this project, these different roles were used to create and test the labs and challenges. While Hacking-Lab has several roles available, four of them (Table 3.1) are important for the basic use of this platform.

Role	**Description**
Student	A standard user of Hacking-Lab, which participates in events.
Teacher	Responsible for grading the solutions submitted by users.
Manager	Sets up the events, teams and classes, and manages resources.
Editor	Creates and edits challenges, quizzes and theories. Responsible for adding, updating or deleting resources.

Table 3.1: Relevant Hacking-Lab roles for this project

Each lab is based on a different Markdown (see section 3.6 for more information on this) based text, formatted to suit the requirements of the platform. The ”resources” subdomain can be used to link various files and Docker images to the challenge, making them available for students to download or work with. Figure 3.1 shows how the lab is structured to make it as easy to use as possible.

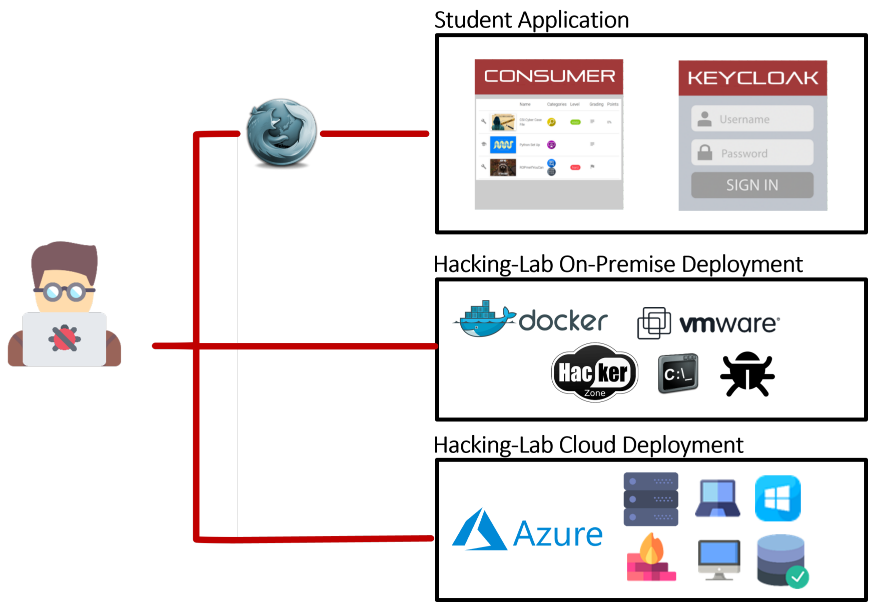


Figure 3.1: Hacking-Lab structure overview

3.5 Hacking-Lab as a user

This section provides the reader with basic information on how to navigate Hacking-Lab as a user or student and how to complete the labs and challenges provided by this project. For a more in-depth overview and information about the Hacking-Lab platform, please visit its info page. [16]

3.5.1 Labs and Challenges

A student of the OST can log in to Hacking-Lab either with the SWITCH edu-ID [17] or by creating an SSO for Hacking-Lab [15]. Once logged in, the student will either have events set up or will have to enter an access code to view them (see Figure 3.2).

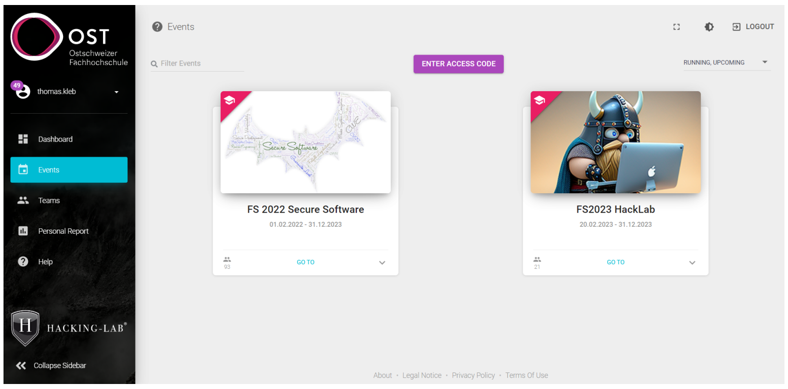


Figure 3.2: Hacking-Lab Event overview

Each event consists of at least one lab that focuses on a specific topic. The lab is then broken down into a series of challenges designed to hone specific aspects of the broader topic. In addition, the challenges are tailored to the skill level of the participants, allowing them to learn at their own pace and grasp the concept effectively.

To solve a Hacking-Lab challenge, students can provide either a flag, a writeup or both. If a student submits a flag, it can be automatically checked to determine whether it is correct or not. However, if a student submits a writeup, it must be graded by another user who has the ”Teacher” role (Figure 3.3). This ensures that the grading is accurate and consistent. The writeup must meet certain criteria and guidelines to receive a passing grade, while the flag only needs to match the predefined correct answer. These two options provide flexibility in the grading process and cater to different learning styles and skill levels.

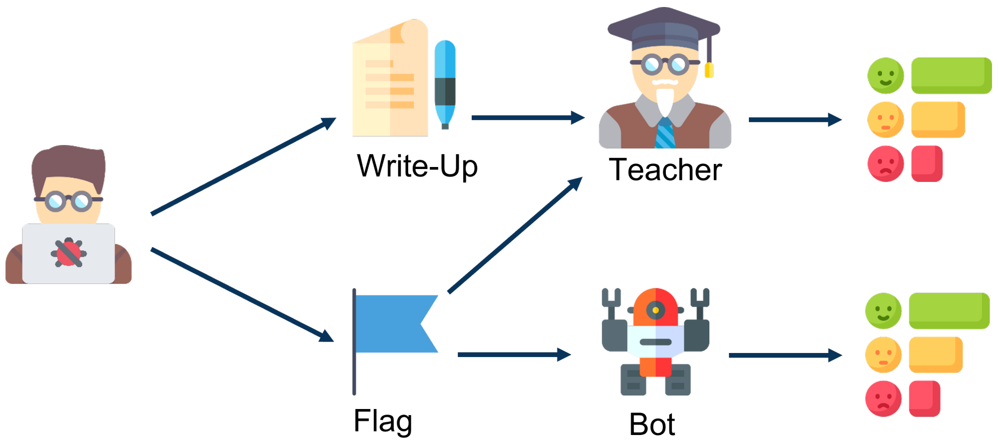


Figure 3.3: Hacking-Lab grading process

Once the solution has been submitted, the student must wait for it to be graded. There are several possible outcomes for it: Rejected (rework of the solution is needed), Partial Points (partially correct, some points are missing) and Full Points (the solution meets all requirements and is complete). This progress is also visible in the progress bar in the top panel of the lab. This bar has a cell for each challenge which is either grey (challenge not started), yellow (some units of the challenge started but not all) or green (all units finished).

3.6 Hacking-Lab as an editor

Challenges have an owner and editors. The owner is the person who created the challenge and has additional options to those of an editor:

- The type of hand-in in any combination (writeup, flag)
- The type of the challenge (Table 3.2)

Each challenge consists of the following parts, which can be edited through the interface or directly on the repository: category & tags, grading & flag, sections and resources. For a challenge to work, each of these must be set (except resources). For this project, only the editor was used.

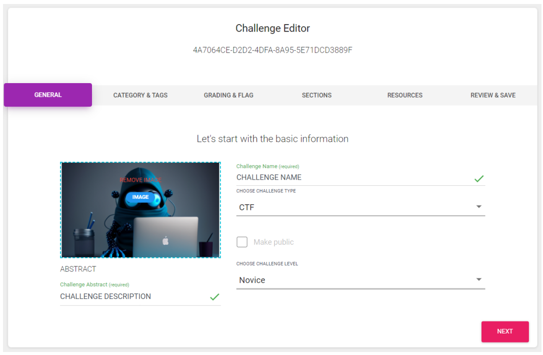
Type	Description
Training	Used for walkthroughs on a certain topic.
Optional Steps	Challenges with explanations that can be viewed for a deduction of points
Competition	These challenges have no walkthrough, only an objective that the student must solve on its own

Table 3.2: Challenge Types

3.6.1 Challenge Creation

This section shows a short walkthrough of how to create a challenge using the editor interface. All the images can also be found in the section D "Screenshots".

When creating a challenge, the editor is first prompted with the general options where the challenge name, description, image, type, and level are set. To continue editing the challenge, a title and description must be set. The other options are optional, but can all be changed later.



Challenge Name (required): CHALLENGE NAME
CHOOSE CHALLENGE TYPE: CTF
CHOOSE CHALLENGE LEVEL: Novice
Challenge Abstract (required): CHALLENGE DESCRIPTION

Figure 3.4: Challenge Editor - General

The next step is for the editor to choose which category the challenge belongs to. This can be done either by selecting one of the available options or by adding a custom one in the field on top. Each of the challenge categories has an icon, which will be displayed on the final page and indicate the domain to the student.

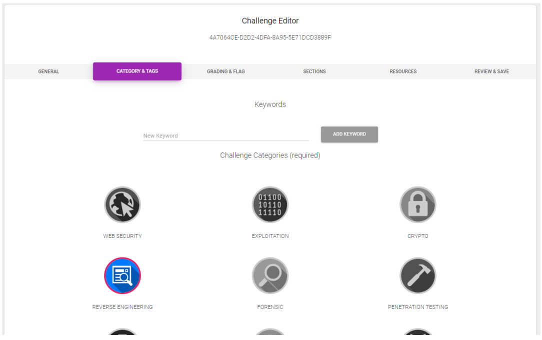
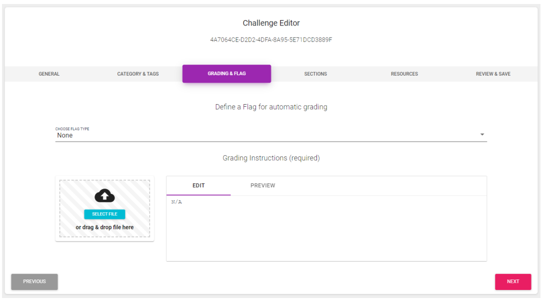


Figure 3.5: Challenge Editor - Categories

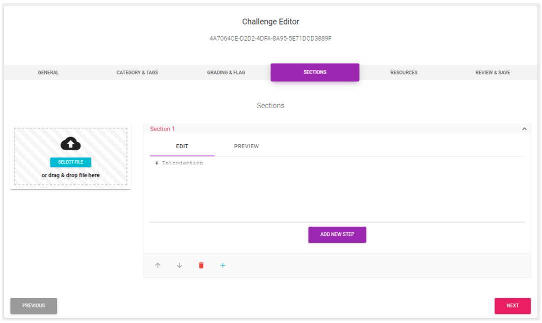
The next setting to change is the flag type from the top dropdown. A flag can either be static, which means it is the same flag for every student, or dynamic, which allows Hacking-Lab to generate it. If the static option is selected, the editor must set it here. In addition to the flag settings, the editor can define grading instructions which will only be visible to the teacher.



CHOOSE FLAG TYPE: None

Figure 3.6: Challenge Editor - Grading

This part of the challenge creation process is used to generate the text and images that will be displayed on the website. A challenge is made up of sections and steps. Sections are the different parts of the challenge, while steps are the drop down menus. All text uses Markdown as its interpreter, and images can be included by uploading them using the upload interface on the left.

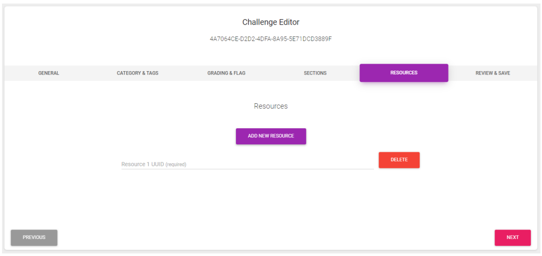


4A7064CE-D2D2-4DFA-8A95-5E71DCD3889F

```
# Introduction
```

Figure 3.7: Challenge Editor - Sections

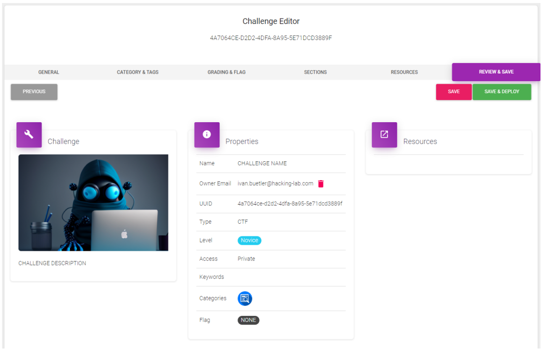
This step involves adding resources from the ”res” subdomain. This is done by uploading the resource, which can be anything from a simple file or zip to an entire Docker instance. To upload such a resource, the generated Universal Unique Identifier (UUID) must be copied and pasted into the field.



4A7064CE-D2D2-4DFA-8A95-5E71DCD3889F

Figure 3.8: Challenge Editor - Resources

The last tab is for checking that everything is as it should be. It shows the challenge as a student would see it when it is deployed. At this point, the editor has two options: either simply save the challenge to edit it later, or save and then deploy the changes. This will bring up a status window and display any errors that have occurred. If there are no errors, the challenge is deployed and can be viewed on the tenant.



4A7064CE-D2D2-4DFA-8A95-5E71DCD3889F

CHALLENGE DESCRIPTION

Name	CHALLENGE NAME
Owner Email	ivan.buetler@hacking-lab.com
UUID	4a7064ce-d2d2-4dfa-8a95-5e71dcd3889f
Type	CTF
Level	Novice
Access	Private
Keywords	
Categories	
Flag	NONE

Figure 3.9: Challenge Editor - Review

3.7 Docker variations

The Hacking-Lab platform offers two different deployment types for the services needed within a challenge (Figure 3.10). These are designed as ”ready-to-run” Docker containers for the solver of a challenge. It is also worth noting that, due to IP whitelisting, the Docker services are only available to people who are currently logged into the Hacking-Lab platform.

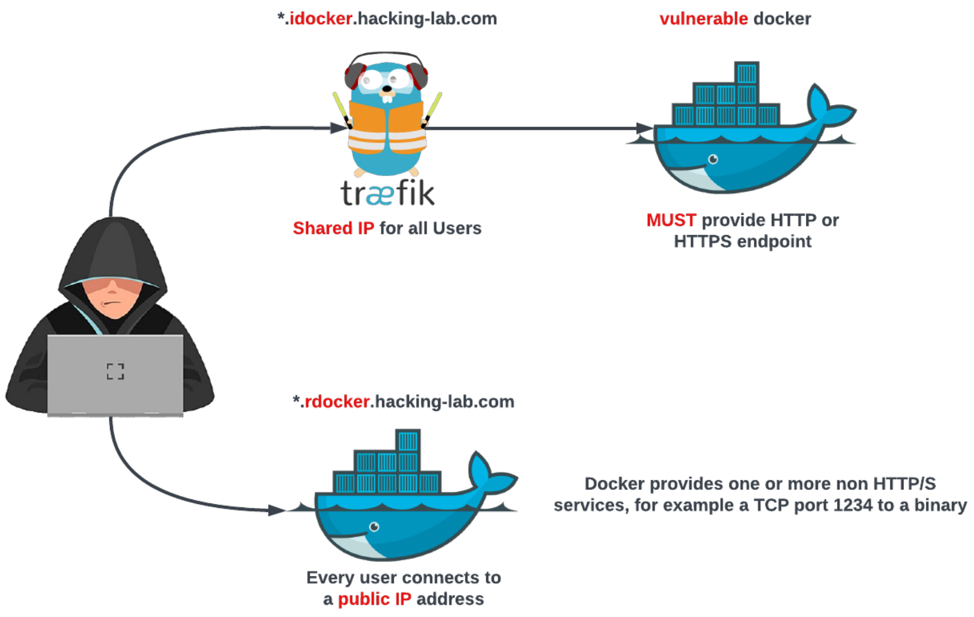

*.idocker.hacking-lab.com
vulnerable docker
træfik
Shared IP for all Users
MUST provide HTTP or HTTPS endpoint
*.rdocker.hacking-lab.com
Docker provides one or more non HTTP/S services, for example a TCP port 1234 to a binary
Every user connects to a public IP address

Figure 3.10: Docker overview

3.7.1 idocker

The **idocker** is useful if the challenge provides either an HTTP or HTTPS endpoint. This type of Docker container is provisioned through the Traefik load balancer and is assigned an internal IP from the private hosting network. [18]

3.7.2 rdocker

The **rdocker** is used when the challenge provides anything **besides** HTTP/S (with very few exceptions, such as exploiting a bad certificate). Rdockers are assigned a public IP address that is accessible from anywhere. This type of Docker could be a binary exposed over TCP, for example, and is the variant we have most often used when a Docker was needed. [19]

Part II

Product Documentation

Chapter 4

Requirements

4.1 Overview

This chapter details the requirements for reverse engineering challenges on the Hacking-Lab platform to ensure a smooth process and high quality results. It covers several aspects such as accessibility, pedagogical effectiveness, manageability, and ethical responsibility.

4.2 Challenge Requirements

The requirements listed here are in addition to those defined by the Hacking-Lab platform itself [20]. They ensure a smooth progression through the semester and a high quality product.

Target Audience

The target audience of the challenges is OST students in the fifth semester. In addition to OST students, the target audience includes any user with similar knowledge.

Language

All challenges and their descriptions are written in English. This ensures that all students and other interested parties can follow the instructions and solve the challenges.

Platform

Hacking-Lab provides Kookarai[1], a Kali Linux based distribution. A challenge that requires the use of Linux must be solvable on Kookarai. Each challenge must indicate which platform the student needs to solve it, both with a morphological box and with tags.

Exercise Grading

Challenges are graded using either a flag or a writeup. Flags are automatically checked by Hacking-Lab itself, but writeups must be graded by the teacher. To make the grading process as easy as possible, sample solutions and a walkthrough video are provided for the teacher.

Time Expenditure

The challenges are designed to be completed by students either individually or in groups. The time for completion should generally not exceed 45 minutes (one class period). This time includes the solving of the challenge and writing of the report.

[1]Kookarai documentation: https://kookarai.idocker.hacking-lab.com/

Programming Languages

C is used to build the binaries used as examples and introductions. Python is the language of choice for writing the scripts used to solve the various challenges.

Visualization

All challenges must be coherent and divided into sections and steps. To ensure this, a template will be created at the beginning of the project and must be followed.

Maintainability

All challenges must be easy to manage and debug. In the future, the artifacts and binaries must not require multiple updates and must work without complication.

Content Restrictions

All challenges must be neutral and must not offend other people, cultures, sexual orientation, politics or religion. They must not contain content that involves drugs, nudity, human exploitation, violence or death. Challenges must not encourage illegal activity.

Chapter 5

Lab Documentation

5.1 Subject Identification Process

In the early stages of the project, a significant portion of the effort was focused on planning and preparing for the coming weeks, as outlined in section 7.3. A crucial aspect of this planning process was the identification and selection of the topics and challenges that would be taught to the students. Based on previous experience with the term project, it was found that it was important to have a variety of potential topics to choose from. This led to discussion and coordination of ideas with the advisor before the construction phase began. The result of these discussions was a table that set the initial boundaries for this project. Table 1.1 from the first chapter shows the final result of our discussions, displaying the topics that were selected to be taught. The full list used in this process can be found in Appendix F. This list was the basis for the construction phase. During the process of creating the challenges, it was decided to split the obfuscation challenge into two and add a hooking challenge, which is why the final list (Table 5.1) differs from the initial one.

Subject Selection		
Priority	Subject	Description
1	Ghidra Introduction	Introduction to Ghidra
2	Ghidra Script Development	Develop a plugin in Ghidra which helps with understanding the code
3	Ghidra GPT Plugin Showcase	Explain how to install and use the Chat GPT plugin for Ghidra
4	Advanced Obfuscation	Advanced obfuscation techniques
5	Control Flow Flattening	Specific and more complicated obfuscation technique
6	Hooking with Frida	Use the tool "Frida" to hook into a running process
7	Anti-Debugging Techniques	Introduction to anti-debugging and how to circumvent it
8	Memory Dumping	Understand how to dump memory
9	ROP Chaining	Use of ROP chaining to perform function calls within a binary
10	Combination challenge	Combination of different challenges

Table 5.1: Table displaying the selected subjects

5.1.1 Decision Process

Ghidra Introduction

As a powerful and widely used reverse engineering tool, it is essential for a reverse engineer to have a solid understanding of Ghidra's capabilities and features. By learning how to use Ghidra effectively, a reverse engineer can analyse and understand complex software systems more efficiently.

Ghidra Script Development

Creating custom scripts for Ghidra can greatly improve a reverse engineer's workflow and efficiency. By understanding how to develop scripts for Ghidra, a reverse engineer can customise the tool to better suit their specific needs and automate repetitive tasks.

Ghidra AI Extension Showcase

By demonstrating the use of Artificial Intelligence (AI) extensions within Ghidra, a reverse engineer can learn how to use AI technologies to speed up the reverse engineering process. This can help to identify potential vulnerabilities and threats within software systems more quickly and accurately.

Advanced Obfuscation

Sophisticated software obfuscation techniques can make it difficult for a reverse engineer to understand and analyse a program. By learning advanced obfuscation techniques, a reverse engineer can better understand how to deobfuscate code and identify potential vulnerabilities.

Control Flow Flattening

Control Flow Flattening (CFF) works by restructuring the code execution path into a single loop along with control flow primitives, making traditional static code analysis less effective. It's understanding is invaluable to a reverse engineer, as it allows for the deobfuscation of complex code structures that could hide malicious code.

Hooking with Frida

Frida is used as a toolkit to intercept and modify function calls in real time, allowing, for example, the arguments passed to a function to be modified, thereby altering the result when executed. A reverse engineer can use this tool to monitor and manipulate the software, allowing for a more in-depth analysis and vulnerability discovery.

Anti-Debugging-Techniques

Anti-debugging techniques are used to complicate engineering efforts by interfering with or preventing the operation of debugging tools. Knowledge of these techniques is beneficial to reverse engineers, as it allows them to bypass these barriers and continue to examine the software.

Memory Dumping

Memory dumping is the process of extracting the contents of a program's memory space, often used for debugging and software analysis. It's a valuable technique for reverse engineers because it reveals runtime configurations, potential vulnerabilities, and variable values, providing deeper insight into software operations and potential points of failure.

ROP Chaining

Return Oriented Programming (ROP) is a technique used by attackers to bypass existing security measures in software. By understanding how ROP chaining works, a reverse engineer can better understand how to detect and defend against ROP attacks.

Combination Challenge

At the end of a lab, students can put their acquired knowledge to the test. This serves not only as a self-check to see if the student has understood the previous topics, but also as a good exercise to manifest what has been learned.

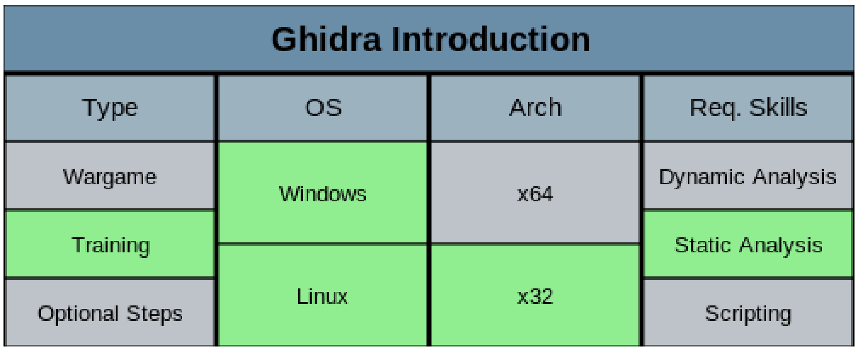
5.2 Morph Box Generator

The challenges also needed to include an image that conveyed the essential elements at a glance. After some brainstorming, it was decided to use a so-called morphological box. This is usually used for the brainstorming itself, but can also be used to give an easy-to-understand overview. The aim of this generator is to be highly customisable, as it will potentially be used in other Hacking-Lab challenges as well as in this project.
Since there was no customisable generator available, it was decided to create a small piece of software to generate these graphics. Python was chosen for this task because of its simplicity and high configurability.
The software now generates such morphological boxes from a JSON template, ready to be used in the challenges for overview purposes. The following listing shows an example of the JSON templates:

```json
{
    "Ghidra Introduction": {
        "Type": {"Wargame": false, "Training": true, "Optional Steps": false},
        "OS": {"Windows": true, "Linux": true},
        "Arch": {"x64": false, "x32": true},
        "Req. Skills": {"Dynamic Analysis": false, "Static Analysis": true,
            "Scripting": false}
    },
    "Second Challenge": {
        "Type": {"Wargame": false, "Training": true, "Optional Steps": false},
        "OS": {"Windows": true, "Linux": true},
        "Arch": {"x64": true, "x32": true},
        "Req. Skills": {"Dynamic Analysis": false, "Static Analysis": true,
            "Scripting": false}
    }
}
```

Listing 5.1: JSON for morphological box generation

This JSON snippet would then generate two images, Figure 5.1 showing what one might look like.



Ghidra Introduction

Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Figure 5.1: Generated morphological box used for the "Ghidra Introduction" challenge

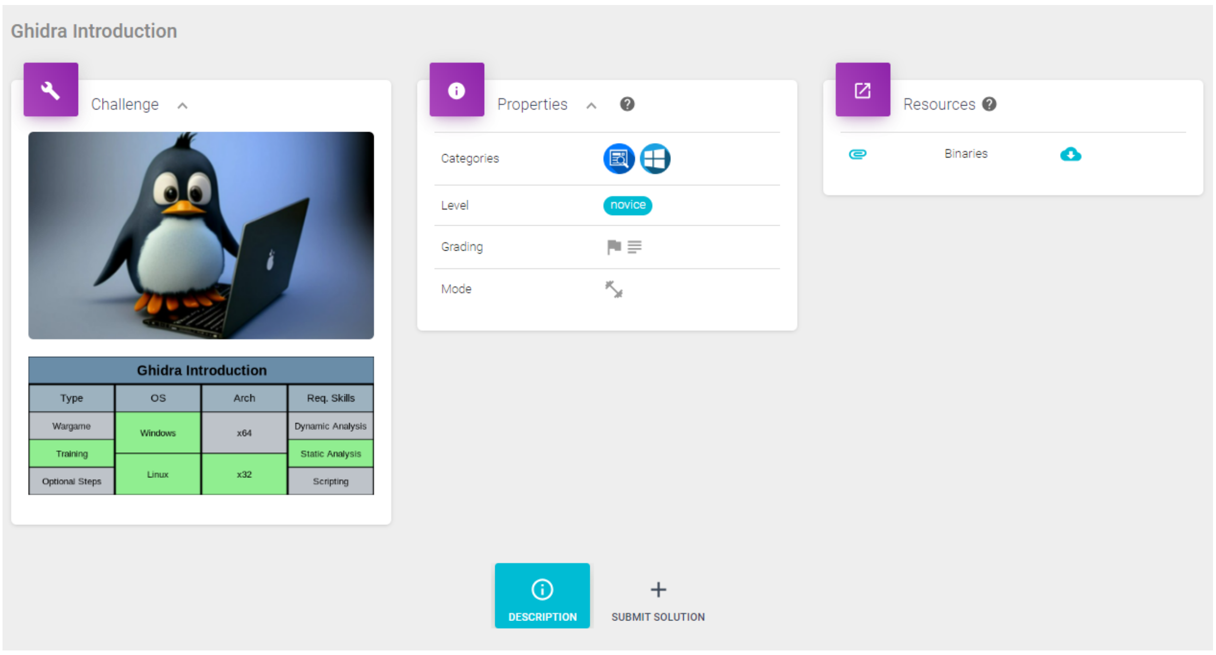
To implement this morphological box generation, the Python library "DrawSVG"[1] was used. The Pillow library was also tested as an alternative, but was quickly discarded as the quality was slightly inferior. As the generated images will be used to give a brief overview of the challenge, colours that are easy to recognise have been selected.

[1]DrawSVG documentation: https://pypi.org/project/drawsvg/

5.3 Challenge Descriptions

This chapter describes all created Hacking-Lab challenges in detail. Images of what each challenge looks like with all its steps collapsed can also be found in the section D "Challenge Sections".

5.3.1 Ghidra Introduction



Ghidra Introduction			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Figure 5.2: Challenge: Ghidra Introduction

Figure 5.2 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Windows, Reverse Engineering
Level:	novice
Grading:	Flag, Writeup
Mode:	Training

Since IDA was introduced in the term project to get the students used to the assembly code, Ghidra, a different disassembler, was chosen for this project. Ghidra offers some advantages over IDA, one of the most important being that it is an open-source software, which means it is freely available to users. It also provides pseudocode, which means that the user no longer has to follow the assembly, but can see the high level code.

Development

Two different C binaries have been written for this challenge, which are used to familiarise the student with Ghidra.

example.exe:

The first binary is very simple; it simply outputs a random number between 0 and 9 when started. This binary is used in this step-by-step explanation of Ghidra.

challenge.exe:

The second binary is a bit more complex and turns an input into an encrypted output. A library was used to encode the message in Base64. The library was authored by Ahmed Elzoughby and released on May 10, 2018 on his public Github repository. [21]
The main function including the encryption calls can be seen in 5.2.

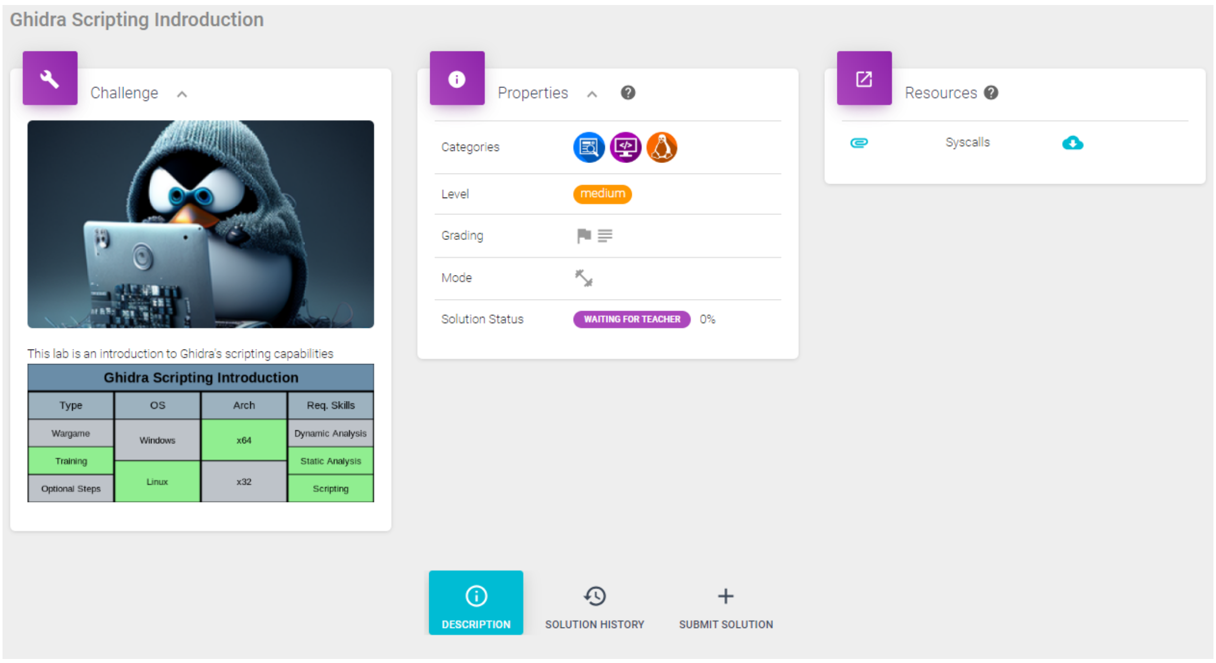
```
const int LINE_LENGTH = 1024;

int main() {
    char line[LINE_LENGTH];
    GetSecretMessage(line);

    const char* ENCRYPTED_MESSAGE = XOR(base64_encode(line), 9);
    DisplaySecretMessage(ENCRYPTED_MESSAGE);
    free(ENCRYPTED_MESSAGE);

    printf("Press any key to continue...\n");
    getchar();

    return 0;
}
```

Listing 5.2: Source code for the first challenge

Summary

As this is an introductory challenge, the main focus is on understanding the necessary features required for the following challenges. After familiarising themselves with Ghidra using the example.exe, the student is given a second, more complex challenge.exe. Along with this binary, they are also given the flag that was encrypted by the challenge.exe. By reverse engineering the challenge.exe, the encrypted flag can be decrypted and the challenge solved.

5.3.2 Ghidra Scripting Introduction



Categories

Level medium

Grading

Mode

Solution Status WAITING FOR TEACHER 0%

This lab is an introduction to Ghidra's scripting capabilities

Ghidra Scripting Introduction			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Figure 5.3: Challenge: Ghidra Scripting Introduction

Figure 5.3 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Linux, Reverse Engineering, Programming
Level:	medium
Grading:	Flag, Writeup
Mode:	Training

With Ghidra being the disassembler of choice, understanding its scripting capabilities will come in handy for the more complex tasks in the future. Ghidra scripts can be written in either Java or Python. In the end, Python was chosen as the programming language because it is a widely used scripting language in the cybersecurity industry, has low overhead and is easy to set up.

Development

This challenge consists of two parts:

Hello World:

The first part teaches the student the basics of Ghidra script development and showcases a simple ”Hello World” script as seen in 5.3.

```
CURRENT_PROGRAM = getCurrentProgram()
PROGRAM_NAME = currentProgram.getName()
print("Hello World from: " + PROGRAM_NAME)
```

Listing 5.3: Ghidra scripting: Hello World

syscalls:

After coding their first script, the student has to analyse a provided binary written in C that invokes several standard library functions and system calls a predefined number of times. The functions called and the number of times they were called can be seen in 5.4.

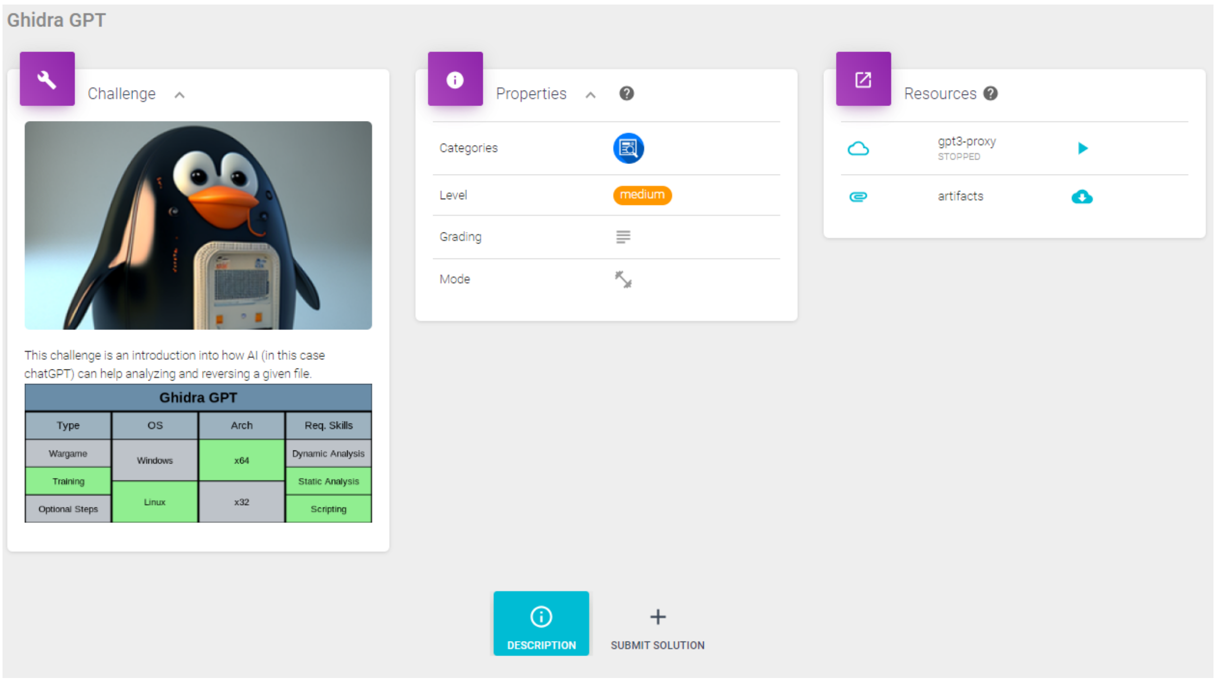
```
strcmp(Str1, Str2);                 // Called 9 times
strncat(Dest, Src, Count)           // Called 3 times
getgid();                           // Called once
gettimeofday(&timeval, NULL);       // Called 14 times
strchr(Str1, Chr);                  // Called 19 times
strlen(Str)                         // Called 3 times
getuid();                           // Called 18 times
system(Command)                     // Called 9 times
strstr(Str, SubStr)                 // Called 16 times
getpid();                           // Called 20 times
```

Listing 5.4: C function calls

Summary

The student is first guided through the development of their first plugin, and then is given increasingly more complex sources to learn from. To test the knowledge gained, the student has to develop a Ghidra script that counts the standard library- and system calls. In order to extract the flag, the number of calls to each function must be mapped to the corresponding letter in the alphabet from top to bottom. To solve the challenge, the developed script that led to the solution and the flag have to be handed in.

5.3.3 Ghidra ChatGPT



This challenge is an introduction into how AI (in this case chatGPT) can help analyzing and reversing a given file.

Ghidra GPT			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Categories	
Level	medium
Grading	
Mode	

gpt3-proxy STOPPED	
artifacts	

Figure 5.4: Challenge: Ghidra ChatGPT

Figure 5.4 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Reverse Engineering, Linux
Level:	medium
Grading:	Writeup
Mode:	Training

Similar to other facets of computer science, AI can help the engineer analyse and process all kinds of data, partially automating and speeding up processes such as reverse engineering. This challenge uses the OpenAI API (section 2.2) together with the newly acquired Ghidra scripting skills (subsection 5.3.2) to assist in all kinds of tasks. The natural language processing models provided by the OpenAI API allow the generation of human-like descriptions of a binary's code, data structures, and algorithms. This combination of tools provides a deeper understanding of the inner workings of the binary and helps identify potential vulnerabilities and security issues, as well as various obfuscation techniques.

Development

During the planning of this challenge, it was decided to develop a Node.js server that acts as a proxy for the student's script to connect to. In this way, is it not necessary for every student to purchase API tokens to solve this challenge, nor for the operator to make his API key public. The Node.js server runs on an idocker and hides direct access to the API key from the student. In addition, the communication between client- and server takes place via a Ghidra script running on the student's machine.

Node.js server:

The following code snippet, 5.5, uses an asynchronous function that generates the text using the OpenAI API with a provided prompt. The `openai.createCompletion` method is used to generate said text with several parameters specified in the object passed as an argument. The `model` parameter specifies the language model to use. In this case, the `gpt-3.5-turbo` model is used, which generates high quality human-like text. The `messages` parameter contains the initial text on which the generated text will be based. The `max_tokens` parameter specifies the maximum number of tokens to generate. Finally, the generated text is accessed and returned using the `data.choices[0].text` property.

```
async function runCompletion(promptText) {
  const completion = await openai.createChatCompletion({
    model: "gpt-3.5-turbo",
    messages: [{"role": "user", "content": promptText}],
    max_tokens: 300,
  });
  return(completion.data.choices[0].message.content);
}
```

Listing 5.5: Completion function for sending text to openAI

Ghidra script:

The Python code shown in Listing 5.6 defines a function that retrieves the decompiled code of the currently selected function in Ghidra. It first gets the currently active program in Ghidra and creates a `DecompInterface` object to provide access to the decompiler functionality. The current program is opened in the decompiler and the function retrieves the decompiled code markup for the current function. The code markup is then returned as a string and stored in the `current_function` variable. This decompiled code is then sent to OpenAI for analysis.

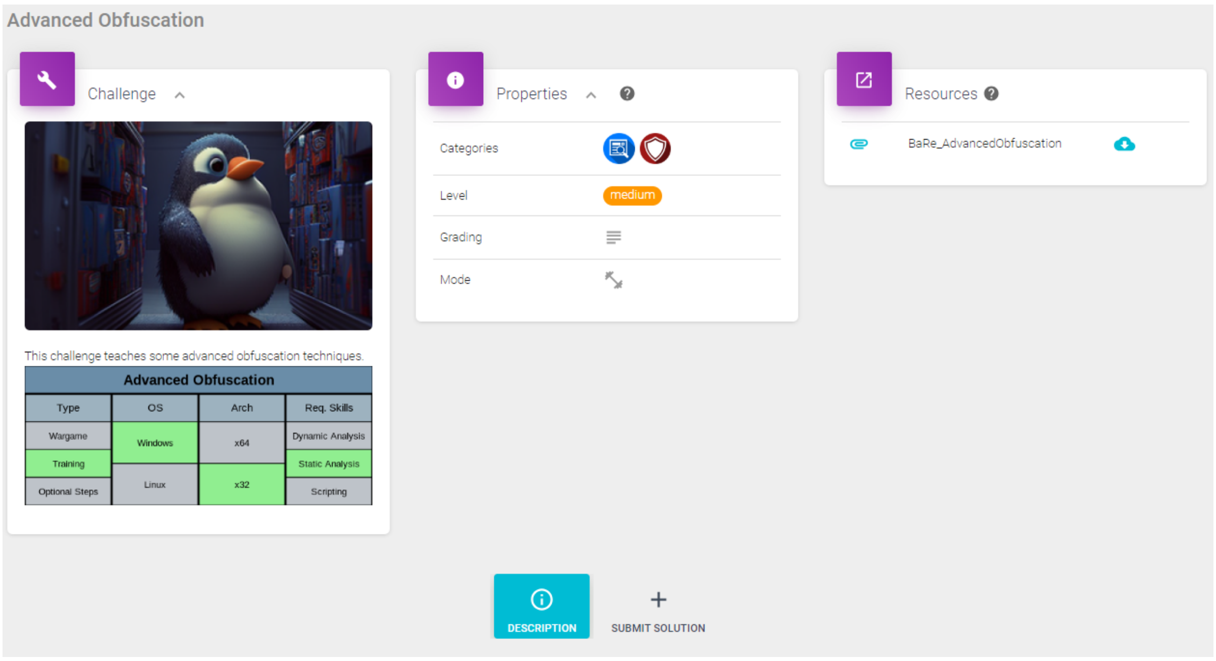
```
def get_current_function():
    currentProgram = getCurrentProgram()
    decompiler = DecompInterface()
    decompiler.openProgram(currentProgram)

    current_function = currentLocation.getDecompile().getCCodeMarkup()
    return current_function
```

Listing 5.6: Python script to get the current function to send to openAI

Summary

This challenge takes the skills learned in the previous challenge and builds on them. As the OpenAI API is a paid service, the student needs to start a Docker and send POST requests to the exposed IP and PORT. The student downloads the script skeleton and adds the functions needed to interact with the OpenAI API. The process of creating the script is guided. Upon completion of the script, the student must modify the question posed so that ChatGPT responds correctly. To complete the challenge, the student must answer the given security questions.

5.3.4 Advanced Obfuscation



This challenge teaches some advanced obfuscation techniques.

Advanced Obfuscation			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Categories	
Level	medium
Grading	
Mode	

BaRe_AdvancedObfuscation

Figure 5.5: Challenge: Advanced Obfuscation

Figure 5.5 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Defense, Reverse Engineering, Windows
Level:	medium
Grading:	Writeup
Mode:	Training

Obfuscation is an important tool for protecting software from malicious reverse engineers. This challenge teaches five different obfuscation techniques and uses them in an example binary.

Development

This challenge uses a two-step process. In the first step, the student is shown the theory behind the obfuscation techniques used. Once they have understood the theory, they are given a binary that has been obfuscated using these techniques.

Theory:

Techniques being taught to the student are:

- Dead Code insertion
- Opaque Predicates
- Indirect jumps
- Function merging
- Method Signature Unification

For an in-depth explanation of these techniques, visit the challenge in hacking-lab[2].

advancedObfuscation.exe

The binary has the functions as shown in Listing 5.7.

```
void print_welcome();
void dead_code(float number);
float addition(void* p[5]);
float function_merged(void* p[5]);
int main();
```

Listing 5.7: Calculator Methods

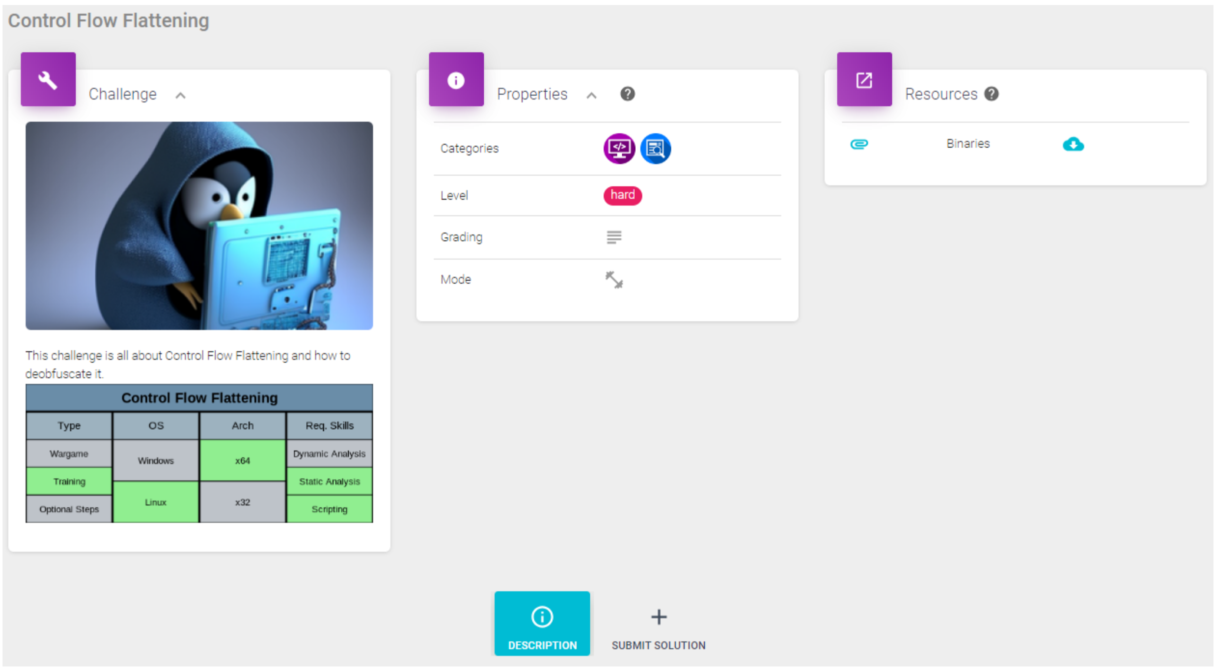
In this example, `addition` and `function_merged` use unified method signatures. `Function_merged` contains the other three mathematical functions and a function to print the whole equation with the result. The binary uses opaque predicates inside the main function and an indirect jump to call the `addition` function. The `dead_code` function is only used to add dead code.
Most of the information on the techniques has been taken from Guardsquare. [22]

Summary

In the first steps, the student learns about the obfuscation techniques used with examples. They will then be given the obfuscated calculator and find out how and where these techniques have been used. A writeup on the findings must be submitted to complete this challenge.

[2]Link to challenge: https://demo.hacking-lab.com/events/98/curriculumevents/126/challenges/378

5.3.5 Control Flow Flattening



This challenge is all about Control Flow Flattening and how to deobfuscate it.

Control Flow Flattening			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Figure 5.6: Challenge: Control Flow Flattening

Figure 5.6 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Programming, Reverse Engineering, Linux
Level:	hard
Grading:	Writeup
Mode:	Training

In addition to the obfuscation techniques described in the previous chapter (subsection 5.3.4), it was decided to tackle a more complex obfuscation technique and construct a challenge around it. Control Flow Flattening (CFF) was chosen because although it is used in commercial software or malware nowadays, no walkthroughs could be found on how to deobfuscate it using Ghidra. This led to the creation of this challenge, which explains CFF and how to unflatten it using Ghidra and Python scripting. To understand the concept behind CFF and the approach used to solve it, the following two sections provide a brief summary.

Control Flow Flattening overview

Control Flow Flattening (CFF) is an obfuscation technique used to hide the flow of a binary by placing its Basic Blocks (BBs) inside a single switch statement. The switch statement itself is inside a while loop and manages the order of execution. This results in a "flattened" appearance of the BBs inside the Control Flow Graph (CFG). Figure 5.7 shows an example of what CFF does when looking at the CFG:

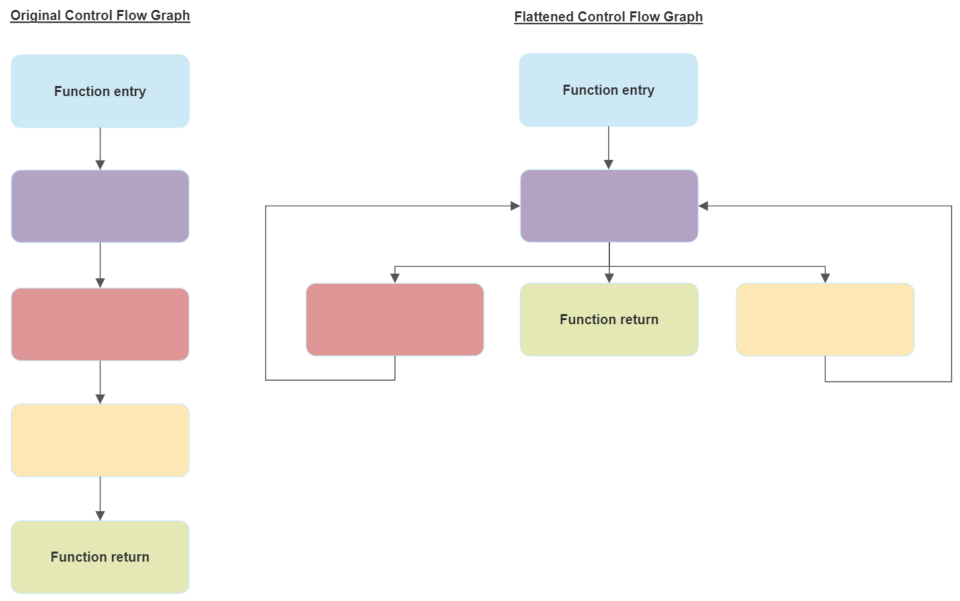


Figure 5.7: Impact of CFF

For further explanation and a sample code, have a look at the challenge in hacking-lab[3].

Symbolic execution

Symbolic execution is a binary analysis technique that evaluates all possible paths of execution. It does this by treating inputs as symbolic variables rather than concrete values. This makes it valuable not only for testing software, but also for understanding the control flow of a binary. Symbolic execution was chosen as the approach for this challenge because it is necessary to examine all possible paths in order to be able to unflatten a binary. Also, since the binaries provided are small, there is no fear of path explosion.
For further explanation, have a look at the challenge in hacking-lab[3].

Development

This challenge begins by explaining what CFF is and how it affects the target binary. Due to the complicated nature of this topic, the student is provided with the necessary theory on how the unflattening process works before they are given the binary exercise to unflatten.

example_flattened:

The theory of unflattening is explained using the example_flattened binary. The unflattening process is divided into four steps:

- Identifying the relevant BBs
- Determining the order of execution of the relevant BBs using symbolic execution
- Undefining the by CFF introduced Basic Block (BB)s
- Patching the binary to make it follow the discovered order

[3]Link to challenge: https://demo.hacking-lab.com/events/98/curriculumevents/126/challenges/379

For all but the second step, it was decided to provide an additional video showing exactly how to apply the written instructions. This approach was chosen because these steps involve sequences of multiple clicks and interactions in Ghidra that can confuse the reader. In addition, video instructions do not bloat the explanation like the equivalent of screenshots.
For the videos, have a look at the challenge in hacking-lab[4].
The only step without a video walkthrough is the actual programming. In step two, the student is given Python code that uses Angr to perform symbolic execution on the example binary. Angr was chosen because it is easy to use, has built-in support for path exploration and constraint solving, which suited our needs perfectly. All the information used to create this challenge has been gathered from a number of sources, including the following: [23], [24], [25], [26], [27], [28], [29].

exercise_flattened:

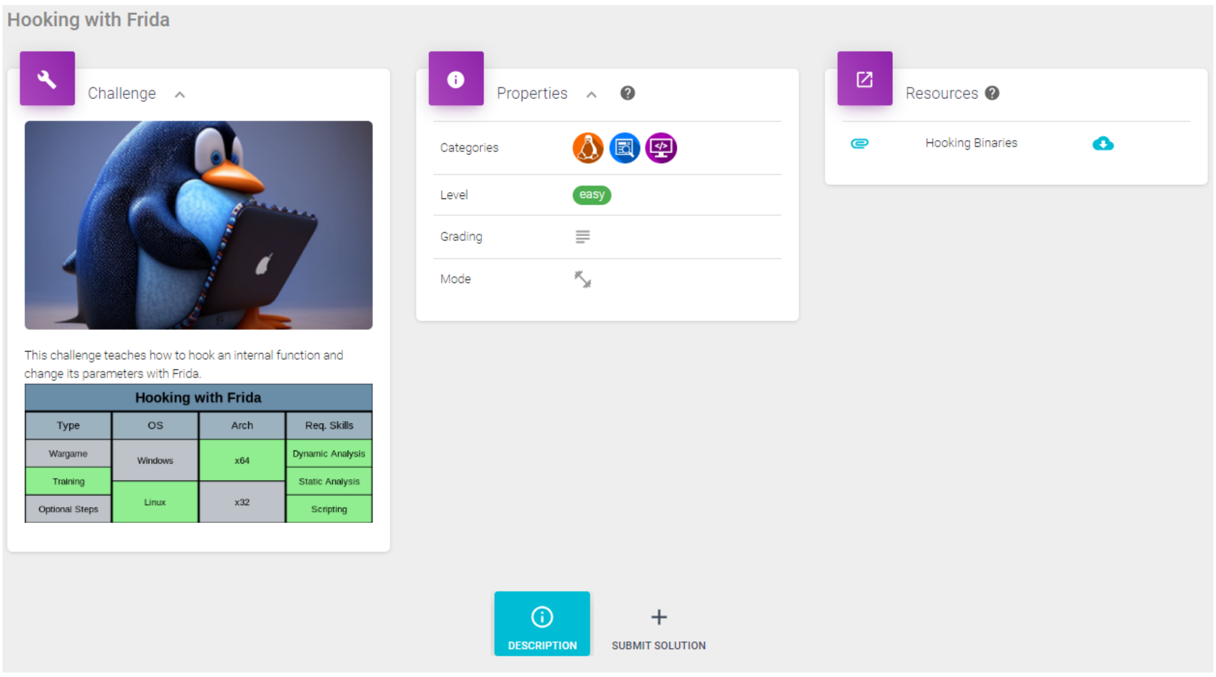
This is the binary that needs to be unflattened by the student. It adds a layer of complexity by having two exit conditions. Each of the two exit conditions results in a different path being taken by Angr and must be manually merged by the student.

Summary

The student learns a more advanced obfuscation technique in theory and then applies it in practice. An example binary is provided, and the student is guided through the entire unflattening process. Next, the student must demonstrate their newfound knowledge of CFF by unflattening another, more complex binary. Finally, the student submits the unflattened CFG of the exercise binary to the Hacking-Lab platform.

[4]Link to challenge: https://demo.hacking-lab.com/events/98/curriculumevents/126/challenges/379

5.3.6 Hooking



This challenge teaches how to hook an internal function and change its parameters with Frida.

Hooking with Frida			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Figure 5.8: Challenge: Hooking

Figure 5.8 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Reverse Engineering, Programming, Linux
Level:	easy
Grading:	Writeup
Mode:	Training

Hooking is an essential part for of reverse engineering. It is used to analyse, intercept and modify the behaviour of an application to understand how it works. A reverse engineer uses hooking to analyse how the functions work and to quickly find vulnerabilities. Frida was recommended by the expert and therefore chosen as the tool to learn about hooking.

Hooking overview

Hooking is essentially a different type of dynamic analysis. The process of intercepting and modifying the behaviour of a program is called hooking. It is possible to place hooks at different levels of the program flow, for example at function calls or API calls. Basically, it allows the user to run custom code that is placed between the original code.
To get a better understanding of this technique, solve the challenge in hacking-lab[5].

[5]Link to challenge: https://demo.hacking-lab.com/events/98/curriculumevents/126/challenges/380

Frida

Frida is an open-source dynamic binary instrumentation tool that allows reverse engineers to inject code using hooks. It provides a framework for instrumenting and analysing software at runtime, without the need to modify or recompile the source.
Frida supports all relevant platforms (Windows, Linux) and even more (macOS, iOS and Android). It provides APIs and bindings for JavaScript, Python and C/C++. The supported platforms combined with the various APIs make Frida approachable for almost everyone.
Frida works by running a server instance inside the target process and creating a client on the host machine. These components then connect to each other and allow data and commands to be exchanged.

Development

The challenge starts with an explanation of Frida, instrumentation and hooking. After installing Frida, the student is ready to follow the guided part of the challenge. For the second part, a custom application was created that has a simple mechanism to get user input and then display that information. After changing the behaviour, it will always display the information injected with Frida.

demo.bin:

This binary is used in the guided part of the challenge, a simple binary asking for name and age. This part explains how to use Ghidra to find the address of the internal function shown in Listing 5.8.

```c
void printInfo (char* n, int age) {
    printf("\n-----------------\nYour name is: %s\n", n);
    printf("and your age is: %d\n", age);
}
```

Listing 5.8: Demo print function

It then shows how to use Frida to place a hook and access the values of the parameters. Finally, it shows how to change these values to change the behaviour of the program.

challenge.bin:

This binary is used in the second part of the challenge. The function to be hooked by the student is shown in Listing 5.9.

```c
int isGuessCorrect(const int GUESS, const int SECRET) {
    if (GUESS > SECRET) {
        printf("Guess was higher than the Secret!\n");
        return 0;
    }
    if (GUESS < SECRET) {
        printf("Guess was lower than the Secret!\n");
        return 0;
    }
    return 1;
}
```

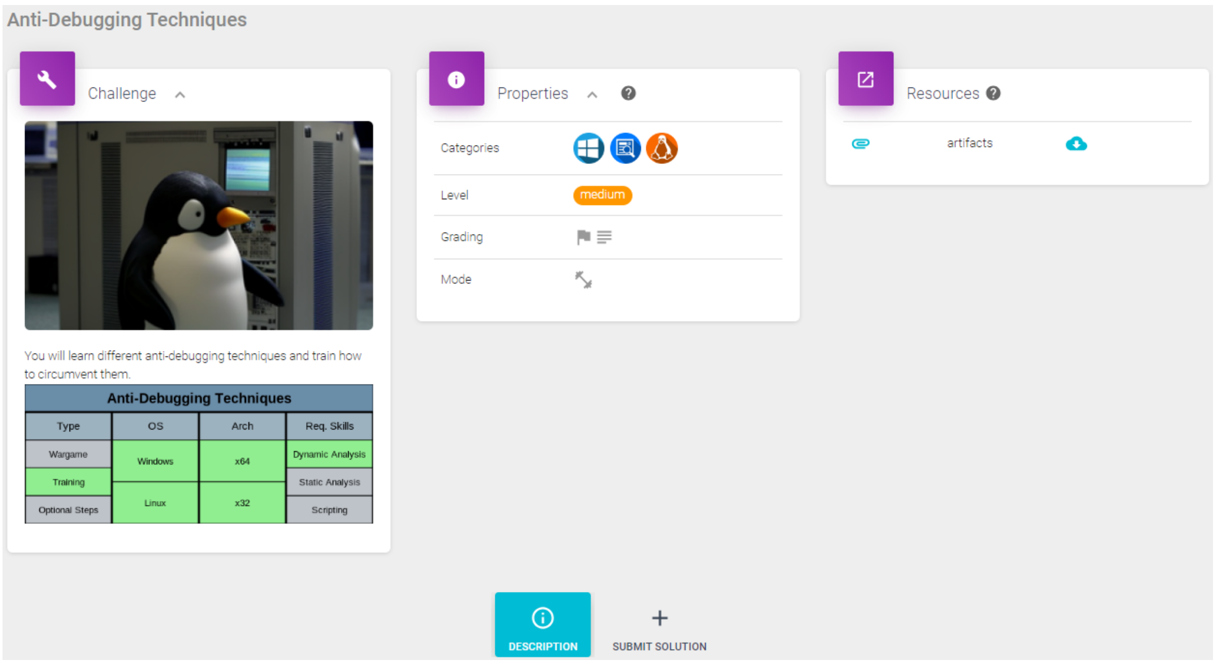
Listing 5.9: Demo print function

All the information used to create this challenge was gathered from Frida's official website. [30]

Summary

The student learns about hooking in theory and then uses Frida in a guided exercise to get a feel for it. They are given a sample binary and information on how to set up Frida to hook an internal function and interact with its parameters. After learning the basics, the student is given the challenge binary, where they have to find a way to always guess the correct random number.
Finally, the student submits their personal solution to the challenge, explaining how it works and what needs to be considered to run their script.

5.3.7 Anti-Debugging Techniques



Anti-Debugging Techniques

Challenge

You will learn different anti-debugging techniques and train how to circumvent them.

Anti-Debugging Techniques			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Properties

Categories	
Level	medium
Grading	
Mode	

Resources

artifacts

Figure 5.9: Challenge: Anti-Debugging Techniques

Figure 5.9 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Reverse Engineering, Windows, Linux
Level:	medium
Grading:	Flag, Writeup
Mode:	Training

Anti-debugging is a set of techniques used in software development to complicate or entirely prevent the debugging process, thereby shielding the code from attackers. Understanding anti-debugging is essential for a reverse engineer, as it can be a significant hurdle to overcome when analysing a binary. Familiarity with these techniques can also help identify and overcome these barriers, allowing for more effective debugging and vulnerability assessment. Ultimately, this knowledge can improve a reverse engineer's skills, which is why this challenge was chosen.

Development

The challenge revolves around the student using dynamic debugging tools to solve problems and bypass implemented anti-debug techniques. Showing all the possible ways in which these barriers can be set up would make the challenge too long. Therefore, five techniques have been chosen that can be used in both Linux and Windows executables. [31] [32] [33]

Selected Techniques
Detecting Software Breakpoints
Ptrace (Linux)
TracerPid (Linux)
Detect Parent Processes
Timing Checks

Table 5.2: Selected Anti-Debug Techniques

Following the techniques shown in Table 5.2, the student will learn how to detect and circumvent them. This includes: How a C binary can detect set software breakpoints, how to use `ptrace` to detect debugging, how to use pseudo file systems such as `/proc` in two different ways (for `TracerPid` and parent processes), and how to set timers that stop the program when a threshold is reached. Information on them has been collected from their man pages[6]. These examples are given as C code for the student to compile. In this way, the challenge provides an open source environment for the student to test different workarounds and bypass the checks. More details about the checks can be found in the challenge[7] itself. For more hands-on experience, two binaries have been created, one for Linux and one for Windows. The Linux binary hasn't been stripped of its symbols to make the process easier, as it is only used for training purposes. On the other hand, the Windows executable, where the challenge flag is placed, also uses the techniques taught, but has its symbols stripped. The goal is to apply the circumvention methods learned and, if used correctly, to read the flag from its memory location.

linux_challenge:

This binary uses the following anti-debugging mechanisms:

- Ptrace
- Timing
- Sofware breakpoint detection

win_challenge:

This is the exercise part of this challenge. It contains the following anti-debug mechanisms:

- `IsDebuggerPresent()`
- Timing
- Checking parent process name

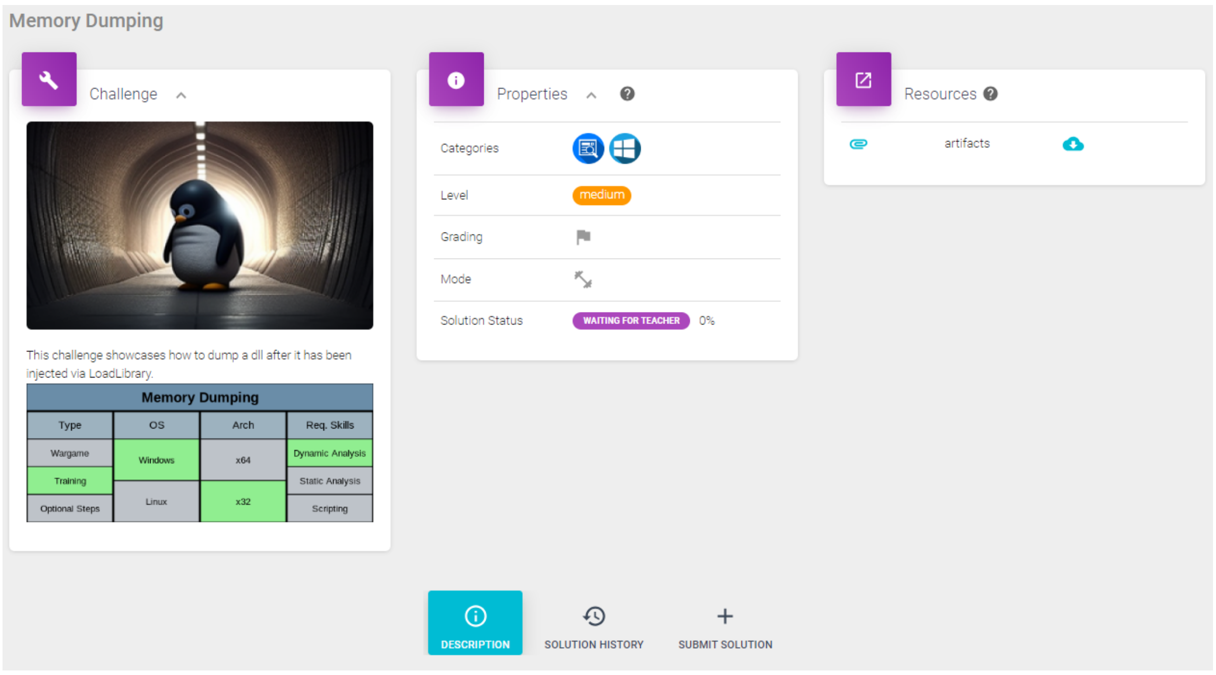
For an in-depth explanation of the above techniques, visit the challenge in hacking-lab.

[6]https://man7.org/linux/man-pages/index.html
[7]https://demo.hacking-lab.com/events/98/curriculumevents/126/challenges/381

Summary

The student is introduced to different types of anti-debugging techniques and then uses the tricks learned in a challenge to obtain the flag. They are first given examples of five techniques that explain basic anti-debugging in a Linux binary. After experimenting with them in the GDB, the student is presented with a Windows executable that employs the same methods but in a different environment. To complete the challenge, the student must both provide the flag hidden in the executable and submit a writeup answering the security questions.

5.3.8 Memory Dumping



This challenge showcases how to dump a dll after it has been injected via LoadLibrary.

Memory Dumping			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Categories	
Level	medium
Grading	
Mode	
Solution Status	WAITING FOR TEACHER 0%

artifacts

Figure 5.10: Challenge: Memory Dumping

Figure 5.10 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Reverse Engineering, Windows
Level:	medium
Grading:	Flag
Mode:	Training

A lot of software, especially malware, uses Dynamic Link Library (DLL) injection to load code into another process at runtime. This makes memory dumping at runtime a crucial part of a reverse engineer's skill set. In addition, memory dumping can help expose information decrypted at runtime. Dumping the decrypted code can be much faster than trying to figure out the encryption algorithm and reverse it statically.

Development

For this challenge, it was decided to focus on Windows systems only, as most of the malware threats target them [34]. Several binaries were developed for this challenge, which are explained in the following sections:

Target.exe:

This binary acts as a dummy executable into which the DLLs are injected. All it does is wait for user input and then close itself. The decision to create this executable was made because this challenge must not violate any EULA by modifying or injecting code into another process.

Injector.exe:

As the name suggests, the purpose of this binary is to inject one of the provided DLLs into the running target.exe. This is achieved by performing a basic `LoadLibrary` injection using `CreateRemoteThread`. There are an unimaginable number of injection techniques of varying degrees of difficulty available on the internet, but `LoadLibrary` combined with `CreateRemoteThread` is one of the easiest for the student to implement and understand, which is why it was chosen for this challenge [35]. Figure 5.11 shows an example of what a `LoadLibrary` injection might look like:

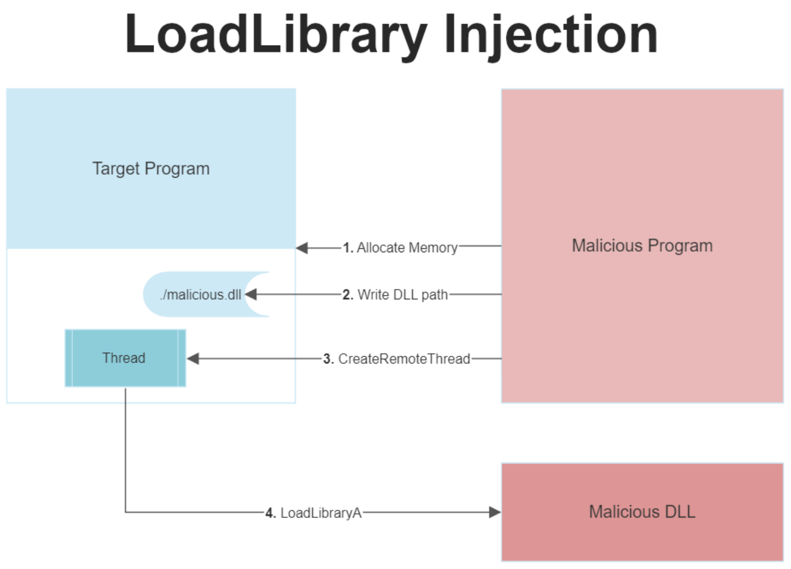


Figure 5.11: Visualisation of LoadLibrary injection

For further explanation, have a look at the challenge in hacking-lab[8].
The decision to use a self-made injector for this challenge was made for two main reasons:

1. Windows does not provide a built-in way to inject binaries into running processes.
2. The target for the injection is hard coded as "target.exe", which should prevent the student from abusing it for other purposes.

Demo.dll:

This DLL was designed to demonstrate memory dumping. It consists of a simple demo flag inside a function that displays a `MessageBox`, signalling which the injection has worked. This function is decrypted at runtime using a simple XOR algorithm as seen in 5.10.

[8]Link to challenge: https://demo.hacking-lab.com/events/98/curriculumevents/126/challenges/382

```c
#define XOR_KEY 0xA
#define HIDDEN_FUNC_LENGTH 0x3E

void DecryptHiddenFunction() {
    DWORD old;
    VirtualProtect(FunctionToHide, HIDDEN_FUNC_LENGTH, PAGE_EXECUTE_READWRITE,
        &old);
    for (int i = 0; i < HIDDEN_FUNC_LENGTH; i++) {
        ((char*)FunctionToHide)[i] ^= XOR_KEY;
    }
    VirtualProtect(FunctionToHide, HIDDEN_FUNC_LENGTH, old, &old);
}
```

Listing 5.10: Function decryption algorithm

The length of the `FunctionToHide` had to be figured out by static analysis, as there was no reliable and easy way to calculate the function size at compile time.

Exercise.dll:

Similar to the Demo.dll, this is a simple DLL, that decrypts parts of its functionality at runtime. But here the function containing the flag is decrypted first and then encrypted again at the end of execution. This should encourage the student to analyse the code more thoughtfully and figure out a suitable moment to dump the DLL before the flag is reencrypted.

Encryptor.py:

Since certain sections of the DLLs need to be encrypted, it was decided to write a small Python script to fulfill that task. This was less tedious than relying on some ancient compile-time encryption headers from the internet. It should be noted that the offsets to these functions had to be figured out using static analysis and hard coded into the script. The script then transforms the previously gathered Relative Virtual Address (RVA) into file offsets.

```python
def VirtualAddressToFileOffset(pe, va):
 for section in pe.sections:
  if section.VirtualAddress <= va < section.VirtualAddress +
      section.Misc_VirtualSize:
   return section.PointerToRawData + va - section.VirtualAddress
 return None
```

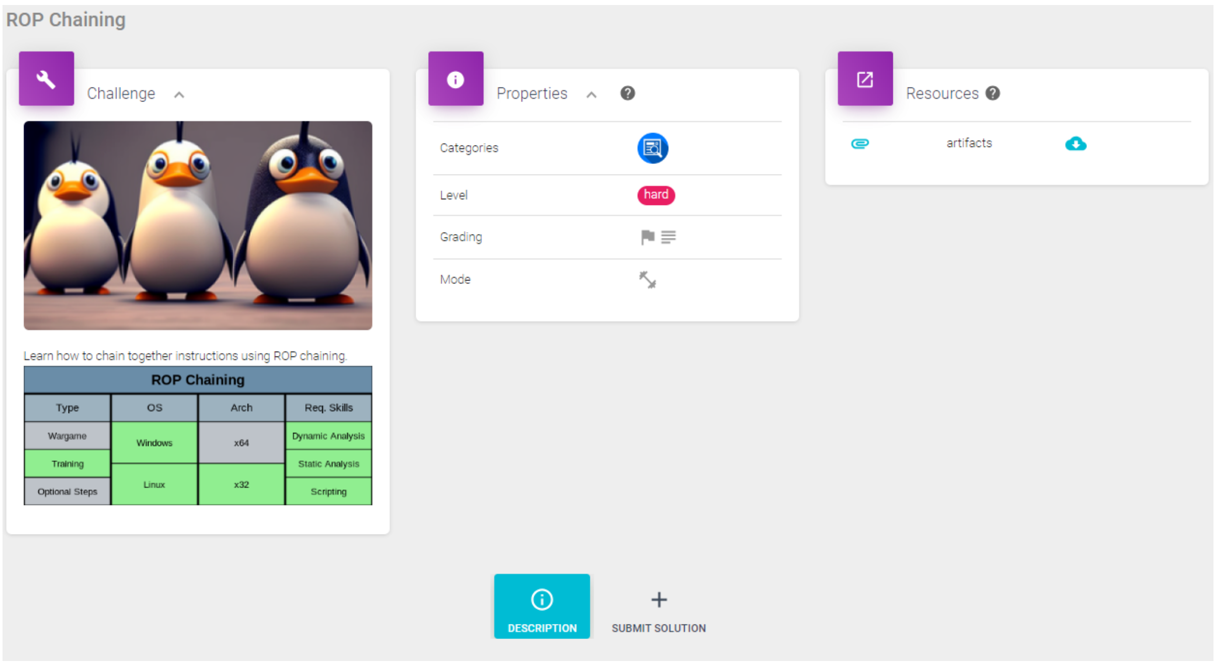
Listing 5.11: Function to transform virtual address to file offset

Using the file offsets, the script encrypts the "to-hide" functions using a simple XOR operation.

Summary

The student is introduced to the concept of `LoadLibrary` injection and memory dumping. They will then be shown how to use x64dbg and its integrated Scylla plugin to extract the loaded binary [36]. After the simple introduction, the student is given an exercise file to dump on their own. Inside the exercise file is a hidden flag that must be submitted to Hacking-Lab to solve this challenge.

5.3.9 ROP Chaining


ROP Chaining

Challenge

Learn how to chain together instructions using ROP chaining.

ROP Chaining			
Type	OS	Arch	Req. Skills
Wargame	Windows	x64	Dynamic Analysis
Training			Static Analysis
Optional Steps	Linux	x32	Scripting

Properties

Categories	
Level	hard
Grading	
Mode	

Resources

artifacts

DESCRIPTION

SUBMIT SOLUTION

Figure 5.12: Challenge: ROP Chaining

Figure 5.12 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Reverse Engineering, Windows, Linux
Level:	hard
Grading:	Flag, Writeup
Mode:	Training

Return Oriented Programming (ROP) is used by many pieces of malicious software [37] to bypass certain security measures, such as non-executable memory segments or runtime protections. Understanding how ROP works is important for a reverse engineer, as it allows the detection of possible attack vectors and helps in the development of countermeasures.
ROP exploits existing instructions (gadgets) within the process's address space. These gadgets can then be chained together, allowing the attacker to do malicious things without ever injecting new code. In short, ROP chains are a means of constructing complex, dynamic behaviour from pre-existing actions. [38]

ROP Gadgets

As mentioned above, the basic idea of ROP is to use small pieces of code called "gadgets" that already exist in the process's address space.
A ROP gadget is essentially a sequence of useful instructions followed by a return instruction, hence the name of the technique. The return / ret instruction can be used to chain several gadgets together, hence the term "ROP chain". It works because a return instruction essentially

pops an address off the stack and jumps to it, allowing control over the programs' execution if the stack can be controlled.
In x86 assembly, registers such as EAX, EBX, ECX are used to store data temporarily at runtime. For example, let's say an attacker finds two ROP gadgets in a program using a tool like ROPgadget [39]:

```
// Gadget 1
0xDEADBEEF  MOV EAX, 5    ; Move the number 5 into the EAX register
            RET           ; Return

// Gadget 2
0xCAFECAFE  ADD EBX, EAX  ; Add the value in the EAX register to the EBX register
            RET           ; Return
```

Listing 5.12: Example for gadgets in a program

In this example, as seen in Listing 5.12, the attacker can use the gadgets to add 5 to the value in the EBX register. This could allow them to bypass some implemented security restrictions. Gadget 1 is invoked first, putting the number 5 into the EAX register. Following this, Gadget 2 is triggered, adding the newly loaded value in the EAX register (now 5) to the existing value in the EBX register. By chaining these two gadgets (Gadget 1 then Gadget 2), the attacker has achieved the desired operation of adding 5 to the EBX register without injecting any new code into the process memory. This is the fundamental concept of ROP and ROP chaining.

Development

The binaries handed out in this challenge are all built for a Linux environment. Kali Linux was chosen to build the challenge binaries because of its widespread use in cybersecurity and its rich environment for learning and implementing hacking techniques. Two binaries were provided to guide the student through the challenge and a final binary containing the flag.

introduction:

The first binary is called "introduction" because it reintroduces the subject of buffer overflows and how to find the correct buffer size using GDB. It also requires the student to correctly chain together two functions that wouldn't be called in a normal program execution, to print out "Hello, World!". Each function typically ends with a return instruction that pops the next address off the stack and jumps to it. A correctly chained sequence would take the program execution from func1 to func2, demonstrating the principle of buffer overflow exploitation.

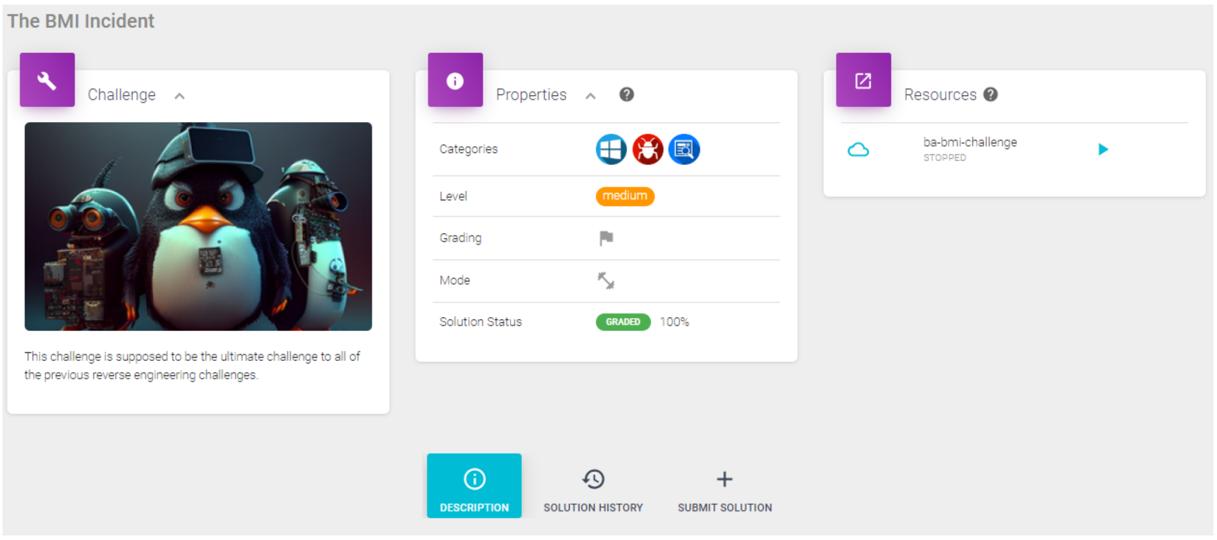
challenge:

To complete the challenge, the student must apply the techniques learned to the second binary. To solve it, the student must use a disassembly tool to find out the correct arguments for the functions and the order in which to call them. The functions use XOR on a flag and, if all are called correctly, print it out in a readable format at the end.

Summary

The student is introduced to the basics of ROP and ROP chaining. They learn how to find and use gadgets to pass arguments to functions that aren't called by default. They first revisit buffer overflows and how to exploit them. Additionally, they learn how to use tools to improve GDB's capabilities, making it easier to solve the challenge. After the introduction, they learn how to find gadgets and chain them together using pwntools. Finally, they are presented with a challenge where they must use the techniques they have learned along with a disassembly tool to find the correct input. To complete the challenge, a writeup, answering two comprehension questions must be handed-in in addition to the flag found in the challenge binary.

5.3.10 BMI Incident



This challenge is supposed to be the ultimate challenge to all of the previous reverse engineering challenges.

Figure 5.13: Challenge: BMI Incident

Figure 5.13 shows the overview of the challenge on the Hacking-Lab platform. The chosen properties for this challenge are:

Categories:	Reverse Engineering, Malware, Windows
Level:	leet
Grading:	Flag
Mode:	Training

This challenge should be the ultimate test for the students. It is a combination of many of the previous challenges and should act as a learning check for them. The experience gained from the following challenges can help.

- IDA Introduction[9]
- x64dbg Introduction[9]
- Ghidra Introduction
- Asm Refresher[9]
- Static Debug[9]
- Dynamic Debug[9]
- Patching[9]

[9]Challenge featured in the term project

The covered techniques are:

- Advanced Obfuscation
- Control Flow Flattening
- Anti Debug techniques
- Memory Dumping

In addition to these topics, a hidden file download from a socket has been added as a new component.

Development

Many smaller binaries had to be created to develop this challenge. These binaries are explained in more detail here.

BMICalc.exe:

This is the main entry point to the challenge for students. It contains an obfuscated BMI calculator that asks for the user's height and weight in order to calculate their BMI. The obfuscation techniques used are indirect jumps and control flow flattening. It also includes anti-debugging techniques to detect a debugger and stop the execution when any is detected. Unbeknownst to the user, it connects to a socket on the Docker and downloads the payload.c as DLL file.

payload.dll:

The DLL contains a function that looks for ”flag.txt” and performs bit-shifts on its content, encrypting it. The content is also encrypted and decrypted at runtime. So again the student has to dump the function at the right time. After analysing this function they will be able to reverse the encryption algorithm used to encrypt ”flag.txt” and get the flag.

solution.py:

Example script to reverse the encryption of the flag provided.

encryptor.py:

Runs on the Docker to do the exact encryption that the received.dll would do. After encrypting the flag the encrypted file is made available for download.

Docker:

This challenge had some new requirements for the Docker that to fulfill. As the challenge is targeted at Windows, the Docker must be able to cross-compile the BMICalc.exe for Windows while running on Alpine Linux. The compilation has to be done on the Docker because the BMICalc.exe needs to know the IP of the Docker in order to connect to it via socket. This direct socket connection needed for this challenge is the reason why a rdocker was chosen.

Summary

The student starts the Docker container and from there they will be able to download their encrypted file (containing the flag) and the BMICalc.exe. They will then have to use many techniques they have learned to see that this calculator does not only calculate the BMI, but also some unexpected things. Following the leads, they will find a function that encrypts a flag.txt file. Armed with this information, they will be able to reverse the encryption on their personal encrypted file. After reversing the algorithm and decrypting the encrypted file, the student submits the UUID to complete the final challenge.

Chapter 6

Results

6.1 Conclusion

During the construction phase, 10 advanced reverse engineering challenges were created. All challenges meet both the requirements set by Hacking-Lab [20] and the additional requirements found in chapter 4. Once created, each challenge was tested by the authors and at least two additional volunteers. This procedure ensured that the level of difficulty was appropriate for the target audience of OST students in their fifth semester. The results of these tests can be found in section 10.2

Upon completion of this lab, students will have a deeper understanding of some of the advanced techniques that can be encountered in reverse engineering. The challenges follow a common theme and increase in difficulty, culminating in a final test.
Additionally, the mentioned lessons learned from section 1.6 were implemented successfully. The time tracking was consistently reported since day one. The testing was improved with additional participants and consequent feedback reporting. A better distribution of the workload across the weeks resulted in more even time spent per team member. The received feedback from the documentation was also heavily considered during the creation of this documentation.

6.2 Future

The challenges created in this project, together with those created in the term project, serve as a solid foundation for students interested in reverse engineering. As they stand, they can be used in any suitable lecture to teach both the basic and more advanced tools and techniques of reverse engineering. In the future, teachers and editors will be able to use this documentation, along with the exercises, to add more topics to the challenges.

Part III

Project Documentation

Chapter 7

Project Plan

Hand-In

The completed report must follow the guidelines set by the Department of Computer Science and the supervisor. The submission process is as follows:

- The PDF version of the report will be sent to the advisor, reviewer and expert and will be archived by OST.
- The printed version will be submitted to the advisor for evaluation and grading.

7.1 Project Management

This following section covers the Time Management and Planning subsections. The former details the allocation of time per author, while the latter outlines the Agile methodology used for planning.

7.1.1 Time Management

The project starts in week 08 of the semester and is scheduled to finish in week 24, giving a total of approximately 16 weeks to complete the hand-In. However, the last week is reserved for poster preparation and printing, leaving a total of 15 weeks for the project. As the module is worth a total of 12 ECTS, students are expected to invest approximately 360 hours on it over the semester. Table 7.1 shows the planned time investment. Consequently, each student should spend approximately 21.3 hours per week on the project. This is calculated as follows: 360 hours total workload for the semester, 15 weeks in a semester plus one week full time, 40 hours, after the normal semester time: (360h - 40h) / 15W = 21.3h/W.

Name	ECTS	Time spent per Week [h]	Total Time spent [h]
Gianluca Nenz	12	21.3	360
Ronny Müller	12	21.3	360
Thomas Kleb	12	21.3	360
Total	36	63.9	1080

Table 7.1: Expected time expenditure per student

7.1.2 Planning

Similar to the previous work on creating reverse engineering challenges for Hacking-Lab, the planning techniques introduced in the Software Engineering Practices (SEP) 1 and 2 modules were used to organise the project. The focus was again on the two main tools discussed: Rational

Unified Process (RUP) and Scrum, which are primarily used in software development but can be adapted for use in other types of projects.
RUP is used to divide the project into four distinct phases - Inception, Elaboration, Construction, and Transition - each of which serves a unique purpose in the development of the project. Inception is used to gain a preliminary understanding of the project and its resolution; Elaboration is used for planning, workload distribution, and concept development; Construction is used to plan, build, and test the project; and Transition acts as a buffer and finalises the product.
To ensure that the project runs smoothly, Scrum's sprints are used to set milestones and tasks that help organise the development process.

7.1.3 Meetings

To ensure efficient progress and a smooth workflow, the team schedules weekly sprint meetings every Monday to discuss and resolve any issues or challenges that may arise. This is also an opportunity to distribute tasks and responsibilities among the team members. In addition, the team meets weekly with their advisor, Ivan Bütler, to provide an update on their progress and to seek advice and guidance as needed.
It is important to note that each meeting is documented and recorded, and the minutes of the advisor meetings are uploaded to the GIT repository for future reference. This ensures that everyone is kept informed of the progress and decisions made during the meetings.

7.1.4 Issue Tracking

Issue tracking is managed using a combination of digital tools and traditional organisational methods. GitLab acts as a central hub for tracking issues, allowing the resolution process to be monitored, classified and coordinated.
The Milestones feature is used to organise these issues, providing a way to prioritise tasks based on their respective due dates and the project timeline. As GitLab has no built-in time tracking, Clockify is chosen to track the time spent on each issue, providing valuable insight into time allocation. A full overview on this can be found in chapter 8.
In addition, sprint meetings are held every Monday to discuss these issues in depth. This routine facilitates the review and reorganisation of tasks, encourages team collaboration and ensures the timely and efficient resolution of tracked issues.

7.2 Roles and Responsibilities

The following section examines how the roles and responsibilities are defined, assigned and communicated within the project team. The section also examines strategies for managing and communicating roles and responsibilities to ensure a successful project completion.

7.2.1 Roles

Authors

Authors conduct research, develop ideas for the different challenges, and write the thesis, adhering to academic standards and guidelines and working collaboratively with other stakeholders.

Advisor

The advisor guides and supports the authors through the research process, ensuring adherence to academic standards and guidelines, and providing expertise on both Hacking-Lab and reverse engineering in general.

Expert and Reviewer

A bachelor thesis needs external people to prevent unfair favouritism. These external people are the expert and the reviewer.
Expert: The expert is proposed by the advisor. The expert must have at least a Master's degree in computer science or similar. The expert must not benefit from the results of the work and must not have a business or family relationship with any of the people involved. The expert assesses the work independently of the supervisor on the basis of the submitted documents and the presentation of the Bachelor thesis. He should also include the practical suitability in the assessment.
Reviewer: The reviewer is appointed by the head of studies. Normally, the reviewer is an advisor of another Bachelor thesis in the same semester. The reviewer does not have to give his or her own assessment like the expert, but only has to contribute another opinion on the grade. [40]

Other Stakeholders

These include the Hacking-Lab staff, who keep the working environment up to date and ensure the stability of the platform, and the academic institution, the Ostschweizer Fachhochschule.

7.2.2 Division of Labour

This section presents the division of labour between the project members as shown in 7.2. It outlines the specific tasks assigned to each member and highlights their responsibilities.

Gianluca Nenz	**Ronny Müller**	**Thomas Kleb**
Scrum Master	Meeting Minutes	GitLab Setup
Ghidra Scripting Introduction	Ghidra Introduction	Documentation Setup
Control Flow Flattening	Advanced Obfuscation	Advisor Meeting Presentation
Memory Dumping	Hooking with Frida	Ghidra GPT
The BMI Incident	The BMI Incident	Anti Debugging Techniques
		ROP Chaining

Table 7.2: Division of labour

7.3 Timeline and Milestones

This chapter provides a comprehensive overview of the project's progress, detailing the various sprints and milestones throughout the project and explaining the time tracking.

7.3.1 Phases and Iterations

The project consists of the four phases of RUP. Each of these phases has several iterations that create the different sprints for the project. Meetings with the advisor are held either on Wednesday or Friday, while team meetings are held weekly on Monday. Each iteration / sprint lasts seven days.
As shown in the inception table 7.3, the project starts before the semester in order to establish a strong foundation on which to build. The other iterations in this phase are used to set up the repositories and the skeleton for the documentation.

Inception			
Iteration	Start	End	Description
0	13.02.2023	19.02.2023	Setting up GitLab for the project.
1	20.02.2023	26.02.2023	Creating CI/CD pipeline and defining documentation structure, searching for ideas for subjects.
2	27.02.2023	05.03.2023	Finishing up base structure for documentation and GitLab, defining base concepts for the subjects.

Table 7.3: RUP: Inception Phase Planning

The elaboration phase 7.4 focuses on planning and assessing potential risks in the project and defining the scope of the project. This includes developing a project plan, implementing risk management strategies and defining the scope of the project to ensure that the construction phase proceed without significant disruption.

Elaboration			
Iteration	Start	End	Description
3	06.03.2023	12.03.2023	Creating project plan and risk management
4	13.03.2023	19.03.2023	Defining labs and subjects

Table 7.4: RUP: Elaboration Phase Planning

The construction phase 7.5 is used to create the different challenges. Each challenge is created in a streamlined manner and went through several stages before being uploaded to Hacking-Lab: The process starts with the creation and discussion of a concept; once the concept is finalised, the challenge is built based on it and eventually tested. For the term project, a limit of one week per challenge is sufficient. In this project, the maximum time allocated per challenge is defined in the concepts, but can not exceed two weeks. This planned time does not include the testing of the challenges.

Construction			
Iteration	Start	End	Description
5	20.03.2023	26.03.2023	Creating "Ghidra Intro" and preparing feedback forms
6	27.03.2023	02.04.2023	Testing "Ghidra Intro" and creating "Ghidra Scripting"
7	03.04.2023	09.04.2023	Testing "Ghidra Scripting" and planning "Ghidra GPT"
8	10.04.2023	16.04.2023	Finishing "Ghidra GPT" and planning "Obfuscation" challenges (CFF and Advanced Obfuscation)
9	17.04.2023	23.04.2023	Testing "Ghidra GPT", finishing "Advanced Obfuscation" and planning "Anti Techniques"
10	24.04.2023	30.04.2023	Testing "Advanced Obfuscation" challenge, planning "Hooking" and "Anti-Debug"
11	01.05.2023	07.05.2023	Finishing and testing of the "Control Flow Flattening"
12	08.05.2023	14.05.2023	Creating "Memory Dumping", finishing "Hooking" and "Anti-Debugging Techniques" and planning the last challenges
13	15.05.2023	21.05.2023	Finish "Memory Dumping", planning and creation of "ROP Chaining"
14	22.05.2023	28.05.2023	Finish "ROP Chaining" and planning last challenge

Table 7.5: RUP: Construction Phase Planning

The transition phase 7.6 is mainly used to finish up the documentation, create the poster for the presentation and finalise all the files needed for the hand-in. A buffer week is added to the transition phase to ensure that enough time is available.

Transition			
Iteration	Start	End	Description
15	29.05.2023	04.06.2023	Finishing last challenge and start finishing the documentation
16	05.06.2023	11.06.2023	Finishing abstract, finishing documentation text, starting with poster
17	12.06.2023	16.06.2023	Buffer

Table 7.6: RUP: Transition Phase Planning

Gantt Diagram

A Gantt chart is an effective tool for visualising and managing the project timeline. For this reason, a Gantt chart is used to display each phase, sprint and advisor meeting. This allows for easy adjustments to be made during the semester, while having an overview of the whole project. To make the chart easier to read, it is divided into the different phases: Iteration and Elaboration grouped (Figure 7.1), Construction (Figure 7.2) and Transition (Figure 7.3) and listed below. Appendix E displays these diagrams at a larger scale.

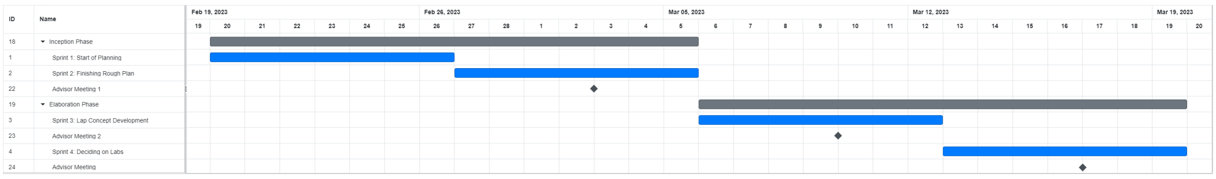


Figure 7.1: Gantt chart: Inception and Elaboration

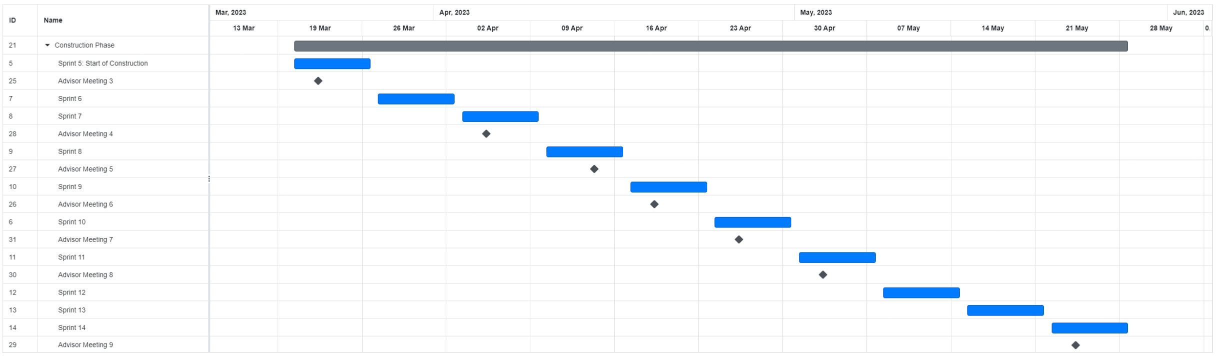


Figure 7.2: Gantt chart: Construction

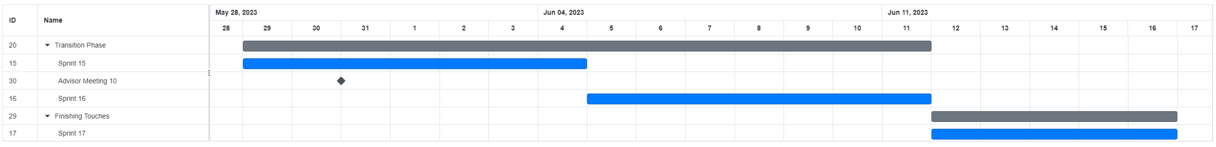


Figure 7.3: Gantt chart: Transition

7.3.2 Milestones

Milestones		
Title	Deadline	Description
M1 - Review of feedback	20.02.2023	Review SA feedback from the advisor and update the structure of the GitLab and documentation accordingly
M2 - Defining problem domain and project scope	10.03.2023	Problem domain and project scope are defined and discussed with the advisor
M3 - First concepts defined, and project plan base finished	20.03.2023	Base for construction phase is finished and prepared
M4 - Expert and reviewer presentation	14.04.2023	Presentation to inform the expert and reviewer about the project is held
M5 - Setup labs	28.05.2023	Labs are set up and tested
M6 - Hand-In	16.06.2023	Document and poster are handed in to the advisor and OST
M7 - Final presentation	08.08.2023	Presentation of Bachelor thesis and its defense

Table 7.7: Milestones set for the project

Chapter 8

Project Monitoring

8.1 Overview

This chapter is used to display and monitor the different stages of the project. It includes a review of each milestone as well as the time spent by each student in total and per category.

8.2 Milestone Review

Table 8.1 provides a structured overview of the key milestones set for the project. It details each milestone, its completion date and any associated notes displaying any complications had.

Milestone	Completion Date	Notes
M1 - Review of feedback	20.02.2023	Milestone finished without complications
M2 - Defining problem domain and project scope	10.03.2023	Milestone finished without complications
M3 - First concepts defined, and project plan base finished	20.03.2023	Milestone finished without complications
M4 - Expert and reviewer presentation	14.04.2023	Milestone finished without complications
M5 - Setup labs	02.06.2023	Milestone wasn't achieved in time but because of the planned buffer no complications arrived
M6 - Hand-In	16.06.2023	Milestone finished without complications
M7 - Final presentation	08.08.2023	Presentation held after hand-in of documentation

Table 8.1: Monitoring Notes for the Milestones

Looking at the notes for each milestone, it can be said that the project was largely carried out according to the original plan. The experience gained during the term project allowed a clean start to this project, as evidenced by the fact that there were no complications during the initial phases. The presentation to the expert and reviewer was considered successful, indicating effective communication and presentation of the project objectives and progress. There was a slight hiccup during the 'set up labs' phase. However, the delay did not cause significant problems as a buffer had been built in during the planning stages. Overall, the milestones were achieved without any significant problems.

8.3 Time Tracking

To ensure accurate and reliable time tracking, Clockify[1] was used throughout the project. Clockify was chosen for its user-friendly interface, robust reporting features and ability to accurately track hours spent on different tasks.

8.3.1 Time Tracked per Student

As calculated in subsection 7.1.1, each student should have spent an average of 21.33 hours per week or a total of 360 hours. In order to complete the documentation in time, it was decided to exclude the last week from the monitoring.

Name	Average Time spent per Week [h]	Total Time spent [h]
Gianluca Nenz	22.96	367.33
Ronny Müller	22.07	353.08
Thomas Kleb	24.81	397.07

Table 8.2: Recorded Time Investments (excluding the last sprint)

Table 8.2 shows how much time each of the students put into this project. As mentioned before, the bachelor thesis is worth 12 ETCS, which means about 360 hours spent in the semester. Taking into account that the time spent in the last week was not monitored, each of the students has reached this target.
The pie charts in Figure 8.1 show how each of the students distributed their hours across each of the categories set up in Clockify. Each category was chosen for its importance in the project. Whenever a student spent time on the project, they used one of these categories to track it. At the end of the semester, the data was collected and presented in the charts below. With three students working on the project, responsibilities could be shared according to each student's strengths. This resulted in a structured working environment and a clean workflow.

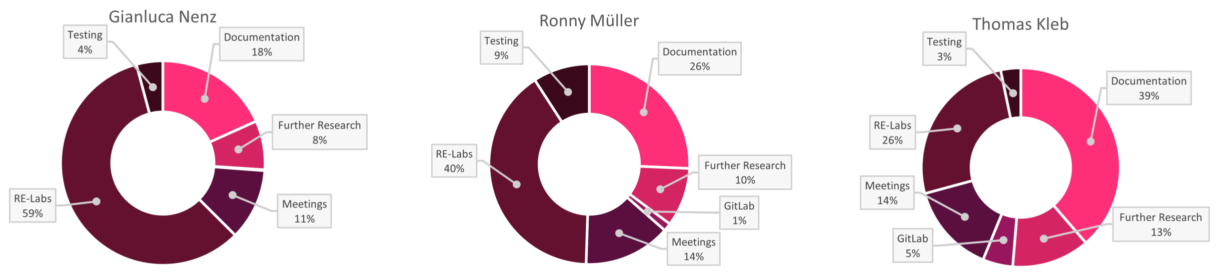


Figure 8.1: Time distribution of each student

[1]Clockify Website: https://clockify.me/

8.3.2 Overall Time Tracked

The bar graph in Figure 8.2 shows the time spent on each sprint of the project. Above each bar is the corresponding numerical value. It can be seen that the early stages of the project took less time on average than the rest of the project because of the previous knowledge gained from the term project. This graph also shows that some time was invested before the start of the semester (Sprint 0) to prepare the platforms to be used.
During the construction phase (Sprint 5 - Sprint 14) an average of about 70 hours per week was spent. This amount is slightly higher than the calculated average (64 hours per week), but as this was the most important phase of the whole project, it was expected to take more time.

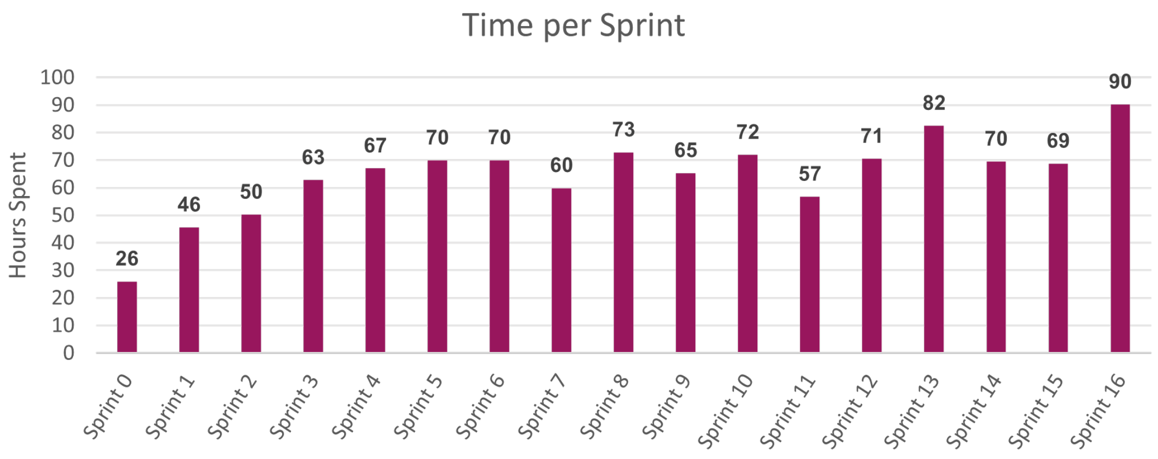


Figure 8.2: Time investment per sprint

As with the personal time distribution in Figure 8.1, a pie chart has been created for the overall time monitoring. Figure 8.3 shows the overall time distribution. As expected, most of the time was spent creating challenges, followed by writing documentation. The next largest category is "Meetings", where the time spent on preparing for and holding meetings and presentations was tracked. The remaining the categories were used to ensure the high quality of the challenges.

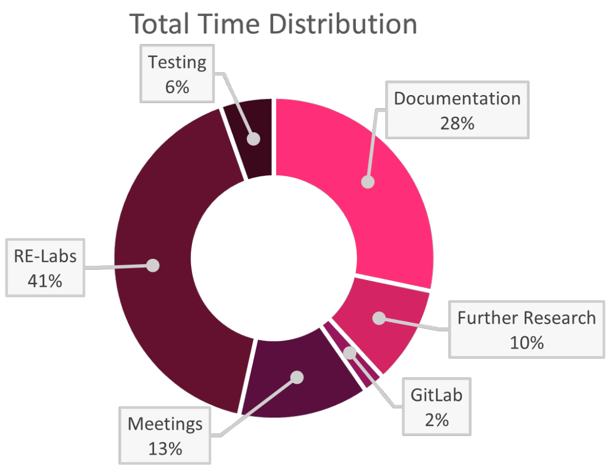


Figure 8.3: Total time investment per category

Chapter 9

Risk Management

9.1 Risks

The following chapter provides a comprehensive analysis of the project risks and the measures taken to manage them. It includes a risk matrix 9.1, risk analysis table 9.2 (see table 9.1 for an overview), a breakdown of risks by project milestone in section 9.2, and a section listing all updates made to the risk analysis table throughout the semester (section 9.3). The risk analysis table identifies and assesses potential risks and mitigating actions. The update history section records the changes made to the risk analysis table to keep it relevant and effective.

9.1.1 Risk Analysis

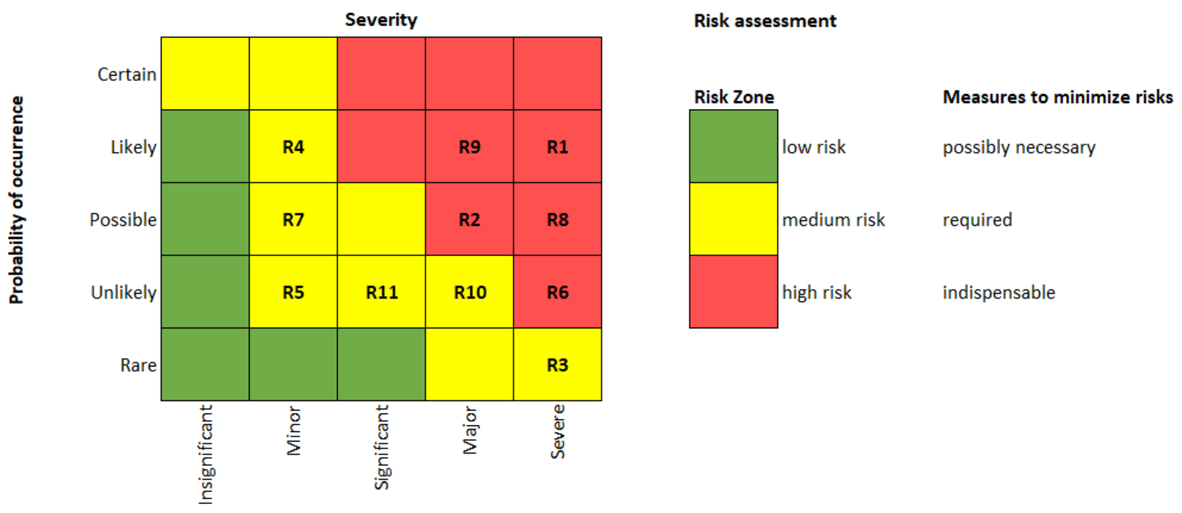


Figure 9.1: Risk matrix

ID	Identifier used for the risk
Description	Explanation of the risk
Probability	The probability of the risk occurring at least once during the project in a qualitative way.
Time Loss	Estimated time loss if risk occurs
Mitigation	How the risk is planned to be mitigated
Severity	The extent of damage the risk can have on the project (measured qualitatively)

Table 9.1: Explanation of the different titles used in the following page

ID	Description	Probability	Time Loss	Mitigation	Severity
R1	Not enough testing done	likely	1-2h	Already got participation confirmation from testers	severe
R2	Not being able to create reversible programs with additional difficulties	possible	2-4h	Assured the advisor is available for consultation	major
R3	Irreparable corruption of git server	rare	4h	Weekly off-site git server backups	severe
R4	Work lost due to work not pushed to source control	likely	0.5-1h	Frequent reminders to push changes	minor
R5	Demo Tenant / Editor down due to maintenance	unlikely	1-24h	No mitigation possible	minor
R6	License problems with used software	unlikely	0.5-1h	Avoid the use of paid software	severe
R7	Missing documentation / support on a tool	possible	1-2h	No mitigation possible	minor
R8	Not enough time for the actual challenges because of too much programming etc.	possible	24h	Creating challenges in chronological order and in an iterating fashion	severe
R9	Work time estimate is too low	likely	5-24h	Precisely define estimations and talk about them with advisor. Add a buffer to the estimate.	major
R10	New and changing requirements	unlikely	5-10h	Define scope in the planning phase together with advisor	major
R11	Not enough communication inside between team-members	unlikely	0.5-1h	Weekly meetings: Sprint and Retrospective	significant

Table 9.2: Risk analysis for whole project

9.2 Encountered Risks

9.2.1 R9 during CFF challenge development

Researching and creating the CFF challenge took more time than expected. This was because no resources were found for learning how to unflatten a binary using Ghidra. As a result, the creator of the challenge had to come up with his own working solution, which took more time.

9.2.2 R9 and R2 during ROP challenge development

While testing the ROP challenge internally, the team discovered that there was a misunderstanding of ROP within the challenge binary. This led to further research on the topic until every aspect was understood. Combined with a recoding of the challenge binary, the creation of this challenge took more time than originally planned.

9.3 Risks Update History

After encountering R9 for the first time, it was decided to change its possibility from ”possible” to ”likely”.

Chapter 10

Quality Measures

10.1 Testing

In this bachelor thesis, quality assurance through testing is an important component. Testing has one of the biggest influences on the structure and quality of the thesis, as the challenges are created to teach reverse engineering to the students. In order to get comparable feedback from the testers, a survey form is used that has to be filled in for each challenge. These forms all have a similar structure, regardless of the specific challenge for which they are completed. More information about the form can be found in subsection 10.1.2.

10.1.1 Process

Challenges are also tracked in GitLab, and are therefore subject to the same requirements as documentation. This means that new challenges are created as a new branch, and when they are ready, a merge request needs to be created. Once this request exists, it is the responsibility of unaffiliated team members to review any changes made and provide feedback using the form before the branch is merged. It has also been decided that at least three people must test a challenge. Therefore, the challenge must be tested by at least one other person after it has been merged.

10.1.2 Feedback forms

In order to get comparable feedback at the end, we rely mainly on closed questions and ratings on a linear scale. This subsection briefly explains the questions we used and why we chose them. The questions are in bold and the reasons are given in the following paragraph.

Did you have previous knowledge on the topic of the challenge?

This question was chosen to be able to compare answers from different levels of knowledge. It helps to see if the tester could have completed the task more quickly due to prior knowledge of the topic.

Did the challenge meet your expectations?

This question checks that the name of the challenge is consistent with its content. It also helps us to understand whether we have covered all the relevant parts of a particular topic.

How would you rate the difficulty of the challenge?

Knowing the level of difficulty is important to compare the time taken and to know if more hints are needed to solve the challenge.

How much time did you spend to solve the challenge?

Estimating the time is very difficult. It is therefore important to get feedback so that the individual challenges do not take too much time. As the aim is to use them in class, they should not take more than an hour.

How would you rate the structure of the challenge?

The term project is used as the basis for the challenges. This question is asked to ensure that it still meets the standards and to see if there is any improvement in the general structure of the challenges.

How would you rate the learning experience?

This question measures the educational impact of the challenge. A high rating indicates effective transfer of knowledge and skills to the examiner. Low ratings may indicate a need for more intuitive problem statements or additional resources.

How much did you enjoy the challenge as a whole?

The level of enjoyment influences motivation and engagement, which in turn influences how much a student learns from the challenge. A high rating indicates that the challenge was engaging and rewarding. Low ratings could indicate a need for more gamification or interactivity.

10.2 Testing Feedback Evaluation

The purpose of this chapter is to analyse the feedback from the volunteer testers. In general, up to 6 people tested the challenges and gave feedback and suggestions for improvement. The following subsection 10.2.1 shows the results for each of the questions asked. For a better overview, the data is presented as a graph, where the x-axis represents the number (ID) of the challenges and the y-axis represents the percentage of responses relative to the total number of responses received for that challenge.

10.2.1 Evaluation

NR	Name of Challenge	Tester	NR	Name of Challenge	Tester
1	Ghidra Introduction	6	6	Hooking with Frida	5
2	Ghidra Scripting Introduction	5	7	Anti-Debugging Techniques	4
3	Ghidra GPT	5	8	Memory Dumping	4
4	Advanced Obfuscation	5	9	ROP Chaining	4
5	Control Flow Flattening	4	10	The BMI Incident	4

Table 10.1: Legend for the feedback tables

Each of the following graphs shows the feedback form data collected for each of the challenges. Table 10.1 is used as the legend for these graphs. The numbers represent the challenge number as listed on the live tenant[1]. This table also shows how many people have tried each challenge. As not all challenges have the same number of testers, a percentage has been chosen. This allows a better comparison between challenges.

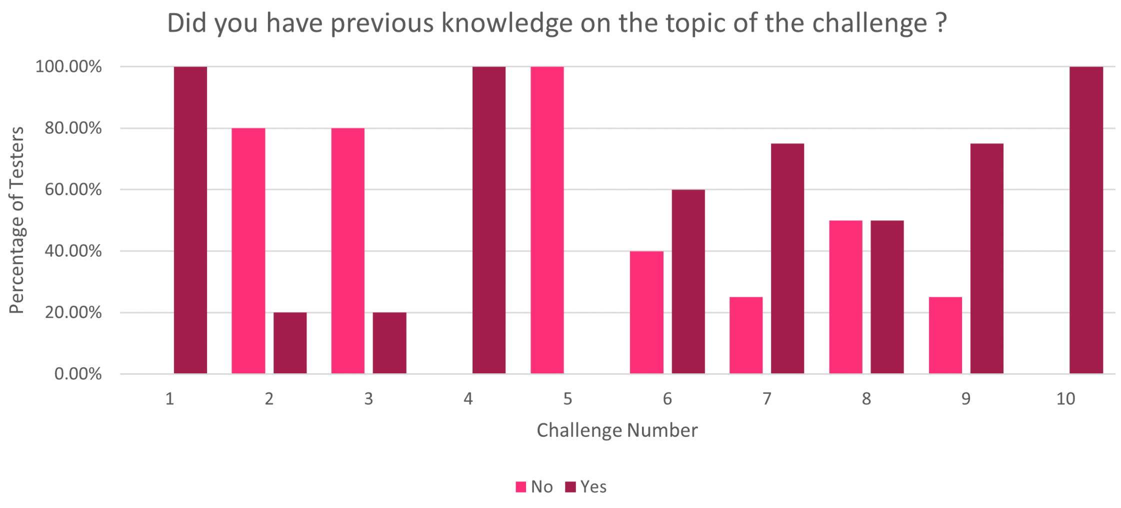


Figure 10.1: Feedback evaluation: Previous Knowledge

Figure 10.1 is the graphical representation of how much prior knowledge the testers had. It shows that most of the challenge topics were new to at least one of the testers. As expected, they all knew about Ghidra, but didn't have much experience writing scripts for it. The rest of the data also represents the expected value of some people knowing about the topic. One anomaly in this data is the number of people with experience in the memory dumping topic. With more data, the graphs will look different, perhaps bringing the result closer to the expected value. Overall, the analysis of this data shows that, as planned, most of the topics were new to the participants. Therefore, the goal of teaching the students new topics was achieved.

[1]Link to tenant: https://demo.hacking-lab.com/events/98/curriculumevents/126

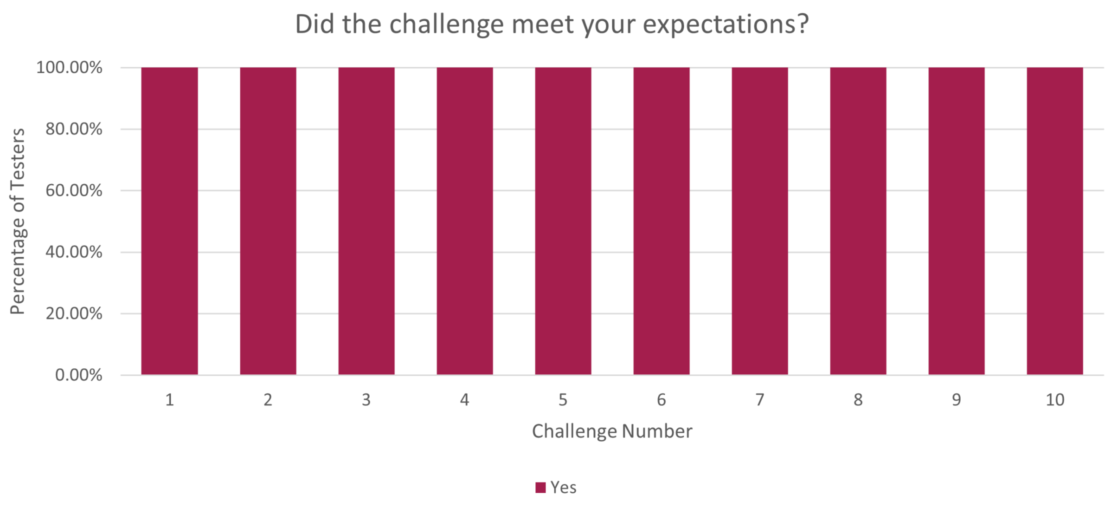


Figure 10.2: Feedback evaluation: Expectations

Figure 10.2 shows whether the challenges met the testers' expectations. Each time a student reads the introduction and title of a challenge, they set certain expectations for it. Looking at the graph, each of the challenges scored, received the highest score for that category. Therefore, the challenges that meet all the expectations of all the testers are optimal for this project. This means that, this project can be used as a reference for future labs and challenges.

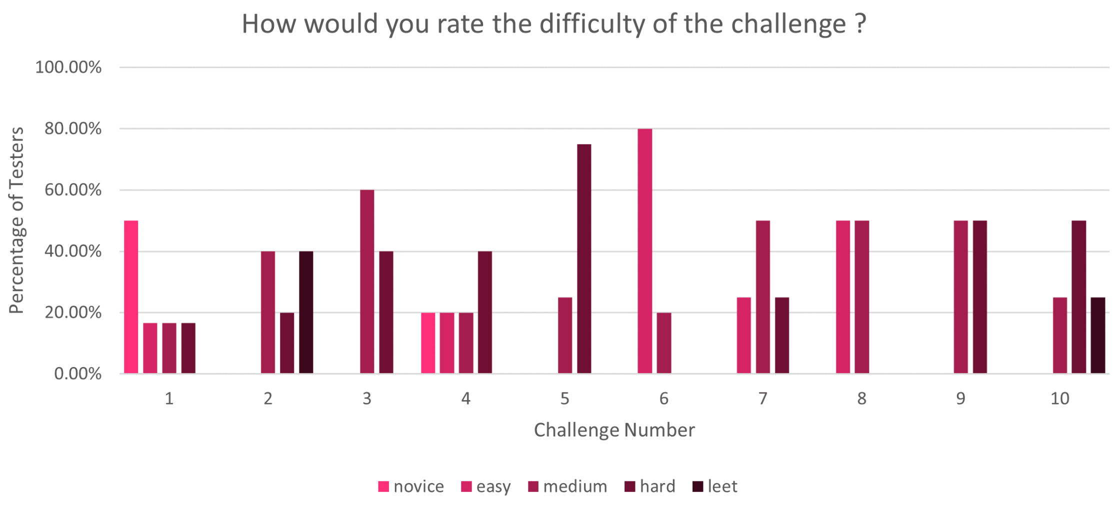


Figure 10.3: Feedback evaluation: Difficulty of Challenges

Figure 10.3 shows how tester would rate the difficulty of each challenge. This graph can be compared to Figure 10.1 as the challenges tend to be harder the less prior knowledge a student has. An anomaly to this rule is Challenge 8, "Memory Dumping". Even though half of the testers had no knowledge of memory dumping, the difficulty was perceived as easy to medium. This is also reflected in the written feedback, as the general response from the testers was that this process was easier than expected. The feedback on difficulty also led to some adjustments, as some challenges were thought to be easier than they were. This evaluation can help future creators of Hacking Lab exercises and challenges to have a reference on which what level of difficulty to choose.

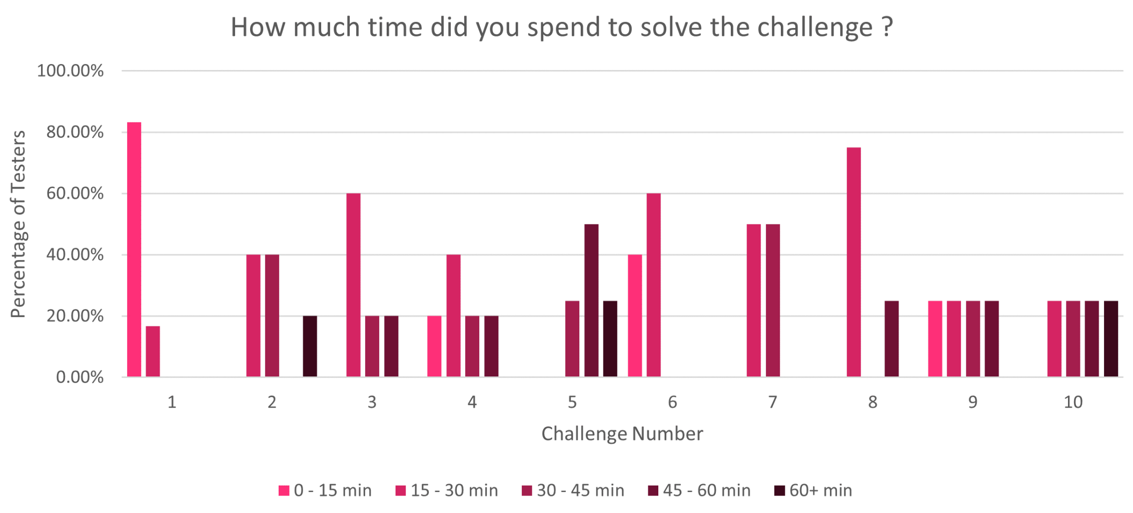


Figure 10.4: Feedback evaluation: Time Spent per Challenge

The target for each challenge was that it should take a student approximately 30-45 minutes to complete. Figure 10.4 shows how much time the testers spent on each challenge. Comparing the results with the target shows that Challenge 6, "Hooking with Frida", doesn't meet the requirements. As this challenge is an introduction to the tool Frida, it would be expected to take less time than the other challenges, but it shows that there would have been space to add more detailed information about the tool. The same goes for the introduction to Ghidra. It is also shorter than the target, but this was expected as it is a basic guide to Ghidra. Another reason why it took the testers less time to complete is Figure 10.1. Most of the testers already had previous knowledge of this tool.

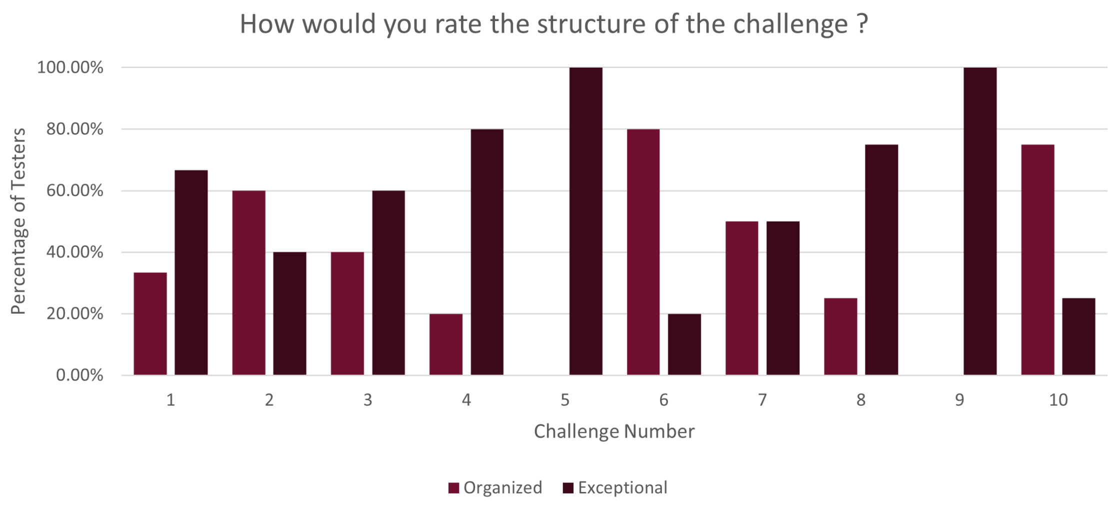


Figure 10.5: Feedback evaluation: Rating of Structure

Figure 10.5 shows how the structure of each challenge was rated by the testers. The better the structure (common thread), the easier it is for a student to understand the steps required to complete a challenge. Each tester was asked to rate the structure between 1 and 5. None of the challenges were rated below a 4, which means that the overall structure is up to standard. This feedback, together with the written notes, was used to correct minor problems.

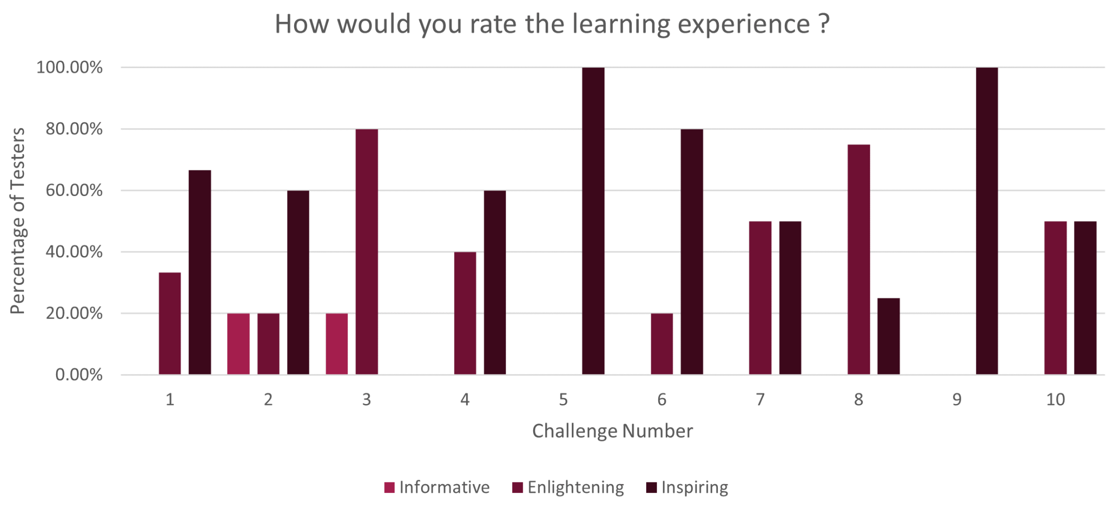


Figure 10.6: Feedback evaluation: Rating of Learning Experience

Figure 10.6 shows how a student would rate the way the topic of each challenge is taught. As in the previous question, the tester can give a rating between 1 and 5. Most of the challenges were rated between 4 and 5, which was the target. Challenges 2 and 3, "Ghidra Scripting Introduction" and "Ghidra GPT" were both rated below the target. Thanks to the written feedback from the testers, these two challenges were adjusted and improved. The graph shows that the challenges created are up to standard and can be used as a reference for creating a learning environment for students.

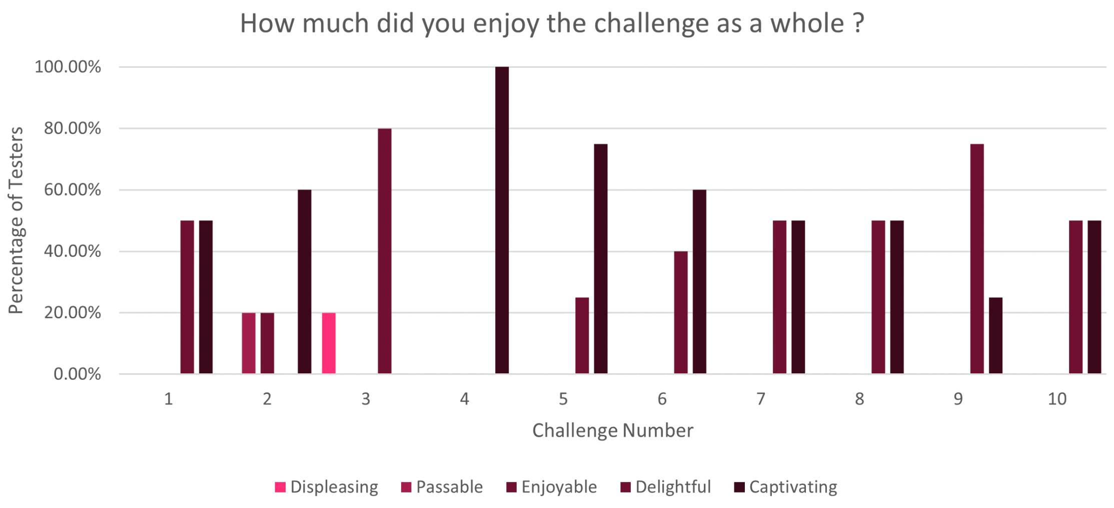


Figure 10.7: Feedback evaluation: General Enjoyment

Figure 10.7 shows how much the testers enjoyed each challenge on a scale of 1 to 5. A comparison with Figure 10.6 shows similarities: the better the learning experience, the better the overall enjoyment. Again, the challenges were improved after receiving the constructive feedback, which means that Challenge 2, "Introduction to Ghidra Scripting", now has a higher enjoyment factor. To ensure high quality, fellow students were asked to re-evaluate the challenge. Another anomaly is Challenge 3, "Ghidra GPT". This particular feedback showed a general dislike of the "AI hype", but no dislike of the challenge itself.

Part IV

Directories

Bibliography

[1] palaksinghal9903. "Basic blocks in compiler design." (2022), [Online]. Available: https://www.geeksforgeeks.org/basic-blocks-in-compiler-design/. (accessed: 5.05.2023).

[2] InfiniteSkills. "Reverse engineering and exploit development." (2017), [Online]. Available: https://www.udemy.com/share/101Zbi/. (accessed: 14.06.2023).

[3] J. Stroschein. "Reverse engineering malware with ghidra." (2020), [Online]. Available: https://www.pluralsight.com/courses/reverse-engineering-malware-ghidra. (accessed: 14.06.2023).

[4] Ghidra. "Ghidra." (2023), [Online]. Available: https://ghidra-sre.org. (accessed: 27.03.2023).

[5] NationalSecurityAgency. "Ghidra." (2023), [Online]. Available: https://github.com/NationalSecurityAgency/ghidra. (accessed: 27.03.2023).

[6] OpenAI. "Openai api reference." (2023), [Online]. Available: https://platform.openai.com/docs/api-reference/. (accessed: 11.04.2023).

[7] OpenAI. "Openai api documentation." (2023), [Online]. Available: https://platform.openai.com/docs/introduction. (accessed: 11.04.2023).

[8] OpenAI. "Api models." (2023), [Online]. Available: https://platform.openai.com/docs/models/overview. (accessed: 19.04.2023).

[9] OpenAI. "Api tokenizer." (2023), [Online]. Available: https://platform.openai.com/tokenizer. (accessed: 19.04.2023).

[10] x64dbg. "X64dbg." (2023), [Online]. Available: https://x64dbg.com/. (accessed: 15.06.2023).

[11] D. Ogilvie. "X64dbg." (2023), [Online]. Available: https://github.com/x64dbg/x64dbg. (accessed: 16.06.2023).

[12] R. Stallman. "Gdb: The gnu project debugger." (2023), [Online]. Available: https://sourceware.org/gdb/. (accessed: 16.06.2023).

[13] R. Stallman. "Gdb git." (2023), [Online]. Available: https://sourceware.org/gdb/current/. (accessed: 16.06.2023).

[14] OST. "Studienschwerpunkte." (2023), [Online]. Available: https://www.ost.ch/de/studium/informatik/bachelor-informatik/studieninhalt-und-aufbau/studienschwerpunkte. (accessed: 05.03.2023).

[15] Hacking-Lab. "Register and login." (2023), [Online]. Available: https://help.ost-dc.hacking-lab.com/login/. (accessed: 05.03.2023).

[16] Hacking-Lab. "Ost info and help." (2023), [Online]. Available: https://help.ost-dc.hacking-lab.com/. (accessed: 23.03.2023).

[17] SWITCH. "Switch edu-id." (2023), [Online]. Available: https://www.switch.ch/edu-id/. (accessed: 24.03.2023).

[18] Hacking-Lab. "Idocker challenge developer." (2023), [Online]. Available: https://www.hacking-lab.com/blog/idocker-challenge-developer. (accessed: 10.04.2023).

[19] Hacking-Lab. "Rdocker challenge developer." (2023), [Online]. Available: https://www.hacking-lab.com/blog/rdocker-challenge-developer. (accessed: 10.04.2023).

[20] I. Bütler. "Challenge requirements." (2023), [Online]. Available: https://hacking-lab.atlassian.net/wiki/spaces/HLSD/pages/234881025/Challenge+Requirements. (accessed: 05.06.2023).

[21] elzoughby. "Base64." (2018), [Online]. Available: https://github.com/elzoughby/Base64. (accessed: 05.04.2023).

[22] G. Goodes. "Beyond control flow flattening: Advanced software obfuscation techniques." (2022), [Online]. Available: https://www.guardsquare.com/blog/beyond-control-flow-flattening-advanced-software-obfuscation-techniques. (accessed: 21.04.2023).

[23] A. Klopsch. "Attacking emotet's control flow flattening." (2022), [Online]. Available: https://news.sophos.com/en-us/2022/05/04/attacking-emotets-control-flow-flattening/. (accessed: 5.05.2023).

[24] eshard. "D810: A journey into control flow unflattening." (2022), [Online]. Available: https://eshard.com/posts/D810-a-journey-into-control-flow-unflattening. (accessed: 5.05.2023).

[25] jscrambler. "Jscrambler 101 — control flow flattening." (2017), [Online]. Available: https://blog.jscrambler.com/jscrambler-101-control-flow-flattening. (accessed: 5.05.2023).

[26] S. Frankoff. "Angr control flow deobfuscation." (2022), [Online]. Available: https://research.openanalysis.net/angr/symbolic%20execution/deobfuscation/research/2022/03/26/angr_notes.html. (accessed: 5.05.2023).

[27] Y. Shoshitaishvili, R. Wang, C. Salls, *et al.*, "Sok: (state of) the art of war: Offensive techniques in binary analysis," 2016.

[28] N. Stephens, J. Grosen, C. Salls, *et al.*, "Driller: Augmenting fuzzing through selective symbolic execution," 2016.

[29] Y. Shoshitaishvili, R. Wang, C. Hauser, C. Kruegel, and G. Vigna, "Firmalice - automatic detection of authentication bypass vulnerabilities in binary firmware," 2015.

[30] Frida. "Frida website." (2023), [Online]. Available: frida.re. (accessed: 12.05.2023).

[31] O. @Apriorit. "Anti debugging protection techniques with examples." (2019), [Online]. Available: https://www.apriorit.com/dev-blog/367-anti-reverse-engineering-protection-techniques-to-use-before-releasing-software. (accessed: 12.05.2023).

[32] J. Olekks. "Nine anti-debugging techniques for application security." (2020), [Online]. Available: https://securityboulevard.com/2020/09/nine-anti-debugging-techniques-for-application-security/. (accessed: 12.05.2023).

[33] cp¡r¿. "Anti-debug tricks." (2022), [Online]. Available: https://anti-debug.checkpoint.com/. (accessed: 12.05.2023).

[34] R. C. "Over 95 percent of all new malware threats discovered in 2022 are aimed at windows." (2022), [Online]. Available: https://www.guardsquare.com/blog/beyond-control-flow-flattening-advanced-software-obfuscation-techniques. (accessed: 22.05.2023).

[35] A. Hosseini. "Ten process injection techniques: A technical survey of common and trending process injection techniques." (2017), [Online]. Available: https://www.elastic.co/de/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process. (accessed: 24.05.2023).

[36] NTQuery, *Scylla*, version 0.9.8, May 2015. [Online]. Available: `https://github.com/NtQuery/Scylla`.

[37] C. Ntantogian, G. Poulios, G. Karopoulos, and C. Xenakis, "Transforming malicious code to rop gadgets for antivirus evasion," *IET Information Security*, vol. 13, no. 6, pp. 570–578, 2019. [Online]. Available: `https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-ifs.2018.5386`.

[38] H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," S. De Capitani di Vimercati and P. Syverson, Eds., pp. 552–61, 2007.

[39] JonathanSalwan, *Ropgadget*, version 7.3. [Online]. Available: `https://github.com/JonathanSalwan/ROPgadget`, (accessed: 08.06.2023).

[40] S. R. M. Stocker. "Leitfaden für bachelor- und studienarbeiten." (2022), [Online]. Available: `https://ostch.sharepoint.com/:b:/r/teams/TS-StudiengangInformatik/Freigegebene%20Dokumente/Studieninformationen/Studien-%20und%20Bachelorarbeiten/Leitfaden%20BA%20SA%20v1.1.pdf?csf=1&web=1&e=hZPfbx`. (accessed: 10.04.2023).

List of Figures

List of Tables

Listings

Part V

Appendix

Appendix A

Advisor Meeting Protocols

The table below gives an overview of the decisions made at each advisor meeting, along with the date and duration. The full minutes are available on the GitLab[1].

Nr	Phase	Date	Description	Duration [min]
1	Elaboration	21.02.2023	Review Term Project and planning the first steps of the bachelor thesis	90
2	Elaboration	03.03.2023	Look over defined challenge concepts and explain them	60
3	Elaboration	10.03.2023	Review our priorisation of challenges and concepts	60
4	Elaboration	17.03.2023	Communicate current state and present plan for construction phase	45
5	Construction	22.03.2023	Present the morphological box generator and questions about documentation	30
6	Construction	05.04.2023	Defined further meetings, update on current state and questions for the mid presentation	80
7	Construction	03.05.2023	Define GPT Docker, update on current state and room reservation	100
8	Construction	24.05.2023	Update project status and plan next steps	60
9	Transition	31.05.2023	Discuss grading instructions and hand-ins	35

Meetings held with advisor

[1] Gitlab - Meeting Protocols: https://gitlab.ost.ch/ba23-reverseengineering/ba23re-documentation/-/tree/main/Meetings

Appendix B

Personal Reports

B.1 Thomas Kleb

This semester I delved deeper into reverse engineering, an area of cybersecurity that continues to pique my interest. The start was less stressful than previous projects, as I was now armed with experience from the term project. Regular contact with our advisor, Ivan, provided clarity and structure, and I'm still very grateful for his professional guidance. My responsibilities included designing different challenges and identifying unique problems for each student to solve, an aspect of the project that I found particularly rewarding. Our team remained organised and effective, with tasks distributed evenly after numerous meetings. Unexpected periods of downtime during the creation of the challenges were managed effectively, as I had learned to plan for such events more accurately. I faced a variety of technical challenges, such as Docker and LaTeX not working as expected, and difficulties in finding subjects, which ultimately served as valuable learning experiences. I was able to apply and improve upon the lessons learned from the previous semester, particularly in the area of pre-planning to streamline our process. Reflecting on my experience, I remain grateful for the opportunity to work on such a significant project and for the support of our supervisor. The excitement of working in this team and on this project has not waned, and I look forward to my future endeavours in cybersecurity.

B.2 Ronny Müller

Being a part of this project brought me immense joy and satisfaction. However, it is essential to note that tensions began to rise dramatically towards the project's conclusion due to meeting deadlines, miscommunication, and misunderstanding. Nevertheless, through effective problem-solving and hard work, we successfully completed the project.

The most fun part was the brainstorming phase. We had the freedom to come up with our own challenge ideas and their topics. Every team member was very invested in the discussion and the potential challenges. It was exhilarating to witness the different ideas come together to create the finished challenge concepts.

The second best part was creating the challenges. I really like the idea of teaching to interested students. This way my interest in reverse engineering combined with my joy in teaching. Overall I felt a sense of pride when overcoming a difficult problem during the project. I had to push my boundaries of my knowledge and skills to find the best possible challenges.

This project taught me a lot about the importance of teamwork, resilience, and adaptability in a group. I also learned a lot about communication and think it will help me in future projects.

B.3 Gianluca Nenz

Being able to write my bachelor thesis on reverse engineering was a huge blessing for me. I was already very interested in the topic before this thesis, so it was a perfect opportunity to deepen my knowledge. As a result, I became very invested in this project and wanted to understand everything down to the last detail.

Thanks to the previous experience from the term project, the whole initial set-up part went really smoothly. We were able to continue with the established routine of creating challenges and dividing our work efficiently. Testing also went well because we started looking for testers from the very beginning and kept on asking for feedback when none came through.

The most enjoyable part for me was definitely researching the challenge topics and coming up with a suitable challenge idea. This was the part where I learned the most. I also enjoyed creating the challenges in Hacking-Lab, which meant teaching the topics we had researched to an interested audience. Although it also showed me once again how hard it is to address all the issues a student might have when solving the challenges.

All in all, I have really enjoyed working on this project, even with the tension building as the deadline approached. It taught me the importance of communication and transparency within the team when faced with difficulties. Ivan was also a great advisor and was always willing to help us with any questions we had. Thanks to the hard work put in by each member of the team, we now have a catalogue of 10 challenges that I am proud to look at.

Appendix C

Sprint Monitoring

Sprint 1: Start of Planning Phase

This sprint is about preparing the base for the project and the semester in general. The goal is to finish the GitLab base, schedule the workdays and prepare the test template.

Duration:	20.02 - 26.02
RUP Phase:	Inception Phase
Open Issues:	0

Figure C.1 shows that most of the time in this sprint was used to prepare and plan for the whole semester. At the end there were no open issues and the sprint was completed without any problems.

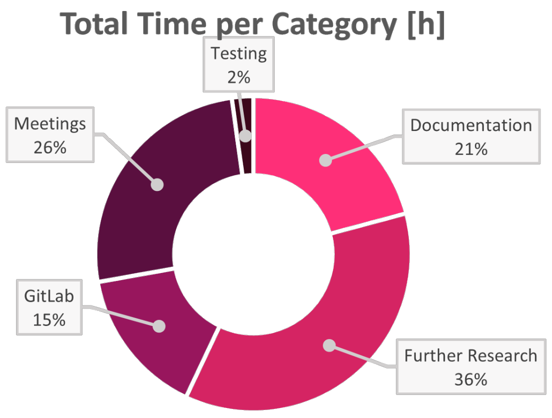


Figure C.1: Chart of the sprint 1 time distribution

Sprint 2: Planning continues...

The purpose of this sprint is to complete the inception phase and to prepare for decisions on which topics will be turned into challenges for the Hacking Lab. Several templates will be created for different purposes, including Labs, Challenges, Graphics and Coding in general.

Start Date:	27.02 - 05.03
RUP Phase:	Inception Phase
Open Issues:	0

As shown in Figure C.2, most of this sprint was used for further planning and setting up to properly complete the inception phase. Most of the time was spent in meetings and preparing the documentation.

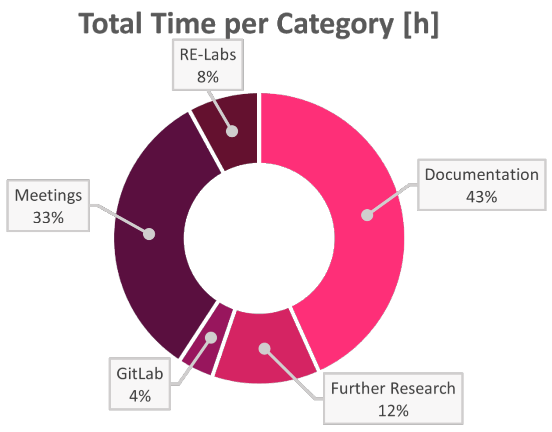


Figure C.2: Chart of the sprint 2 time distribution

Sprint 3: Lab concept development

This is the first sprint of the elaboration phase. The aim of this phase is to find out which labs will be presented on the platform and to finalise the basic structure of the documentation, together with the first sketch of the project plan.

Start Date:	06.03 - 12.03
RUP Phase:	Elaboration Phase
Open Issues:	”Docker Description”

Figure C.3 shows that the documentation required the most attention for this sprint. Further research was done for the labs and several meetings were held to ensure the transition to the new phase of the project.

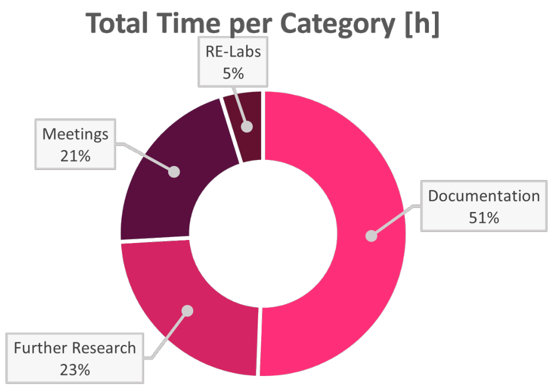


Figure C.3: Chart of the sprint 3 time distribution

Sprint 4: The planning returns

This sprint is used to prepare the first lab concepts, which will facilitate the start of the construction phase. It is also used to finalise the preconstruction documentation.

Start Date:	13.03 - 19.03
RUP Phase:	Elaboration Phase
Open Issues:	0

Figure C.4 shows, that most of it was spent on finishing the documentation. The rest of the time was spent on preparing each other category for the start of the construction phase.

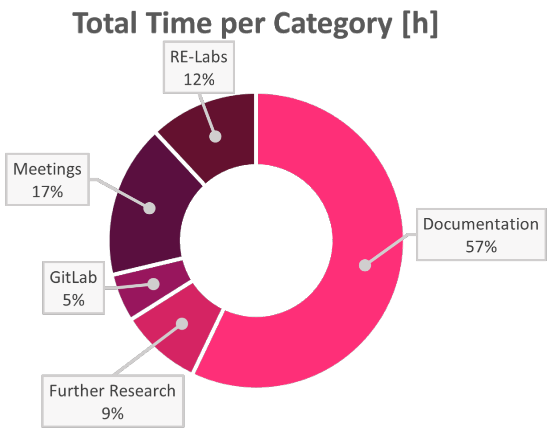


Figure C.4: Chart of the sprint 4 time distribution

Sprint 5: Start of Construction Phase

This sprint kicks off the build phase. It will be used to finish the basis for the documentation and to decide how the development will be documented. This sprint is also used to finish the first version of the morphological box scripts.

Start Date:	20.03 - 26.03
RUP Phase:	Construction Phase
Open Issues:	0

Due to the setup required for both documentation and challenges, most of the time was spent in these categories. Figure C.5 reflects this.

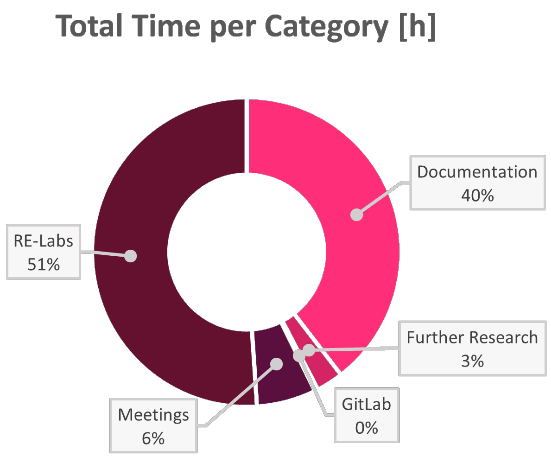


Figure C.5: Chart of the sprint 5 time distribution

Sprint 6: Next Step of Construction

This sprint, similar to Sprint 5, is used to work on both the documentation and the challenges. The first challenges to be created are not as time consuming as the rest. The first challenge "Ghidra Introduction" will be tested, and the next two will be created.

Start Date:	27.03 - 02.04
RUP Phase:	Construction Phase
Open Issues:	0

As with the previous sprint, most of the time was spent creating the challenges and writing the documentation. This ensured a clean start and optimal setup for the coming weeks. Figure C.6 shows an overview of the time distribution.

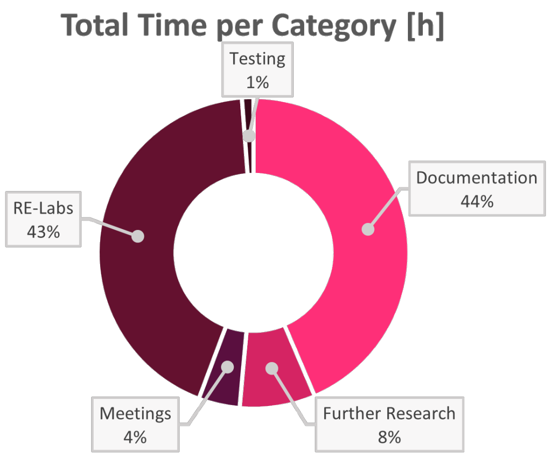


Figure C.6: Chart of the sprint 6 time distribution

Sprint 7: Pre Mid Presentation

This sprint is planned to start planning the more advanced challenges and to finish the GPT challenge. It is also the last sprint before the presentation for the reviewer and expert.

Start Date:	03.04 - 09.04
RUP Phase:	Construction Phase
Open Issues:	0

Figure C.7 shows that again most of the time was spent on creating and managing challenges. It also shows that longer meetings were held with both the consultant and the authors to prepare for the following week's presentations.

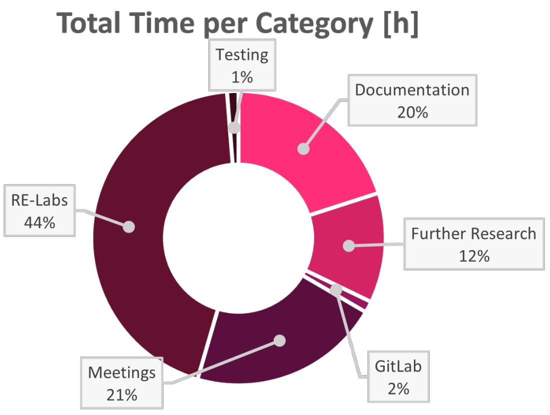


Figure C.7: Chart of the sprint 7 time distribution

Sprint 8: Mid Presentation

As well as creating the challenges, this sprint will focus on the presentation at the end of the week. This sprint should finish the Advanced Obfuscation and Ghidra GPT challenges.

Start Date:	10.04 - 16.04
RUP Phase:	Construction Phase
Open Issues:	0

Figure C.8 shows that most of the time in this sprint was spent on planning the meetings and presentation, along with creating the challenges.

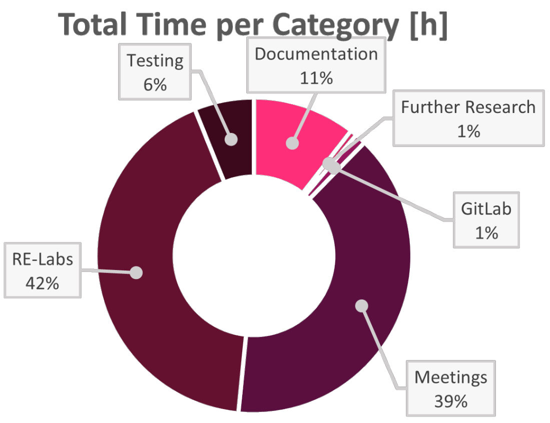


Figure C.8: Chart of the sprint 8 time distribution

Sprint 9: Constructing Challenges

This sprint is used to test the ”Ghidra GPT” challenge and to create the ”Advanced Obfuscation” challenge. A product of this sprint is the decision to split the obfuscation subject into two separate challenges.

Start Date:	17.04 - 23.04
RUP Phase:	Construction Phase
Open Issues:	0

This is reflected in Figure C.9. It shows that creating the challenge takes the most time, followed by documentation and further research.

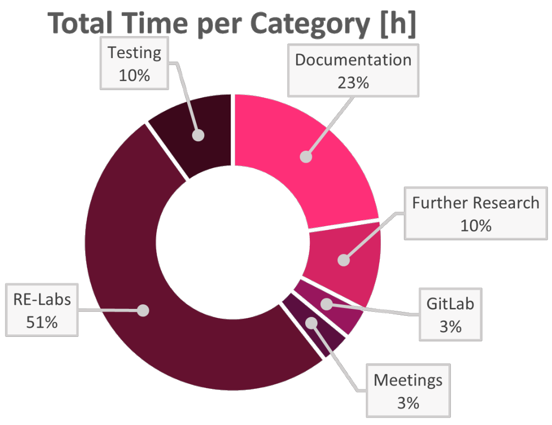


Figure C.9: Chart of the sprint 9 time distribution

Sprint 10: Constructing Challenges

Sprint 10 is used to create more challenges. In this case the ”CFF” and the ”Hooking with Frida” challenges. In addition, some time is planned to focus on the following topics and to fix mistakes in previous challenges.

Start Date:	24.04 - 30.04
RUP Phase:	Construction Phase
Open Issues:	0

Figure C.10 shows that nearly three quarters of the time were used to improve the previous challenges and to create the new ones.

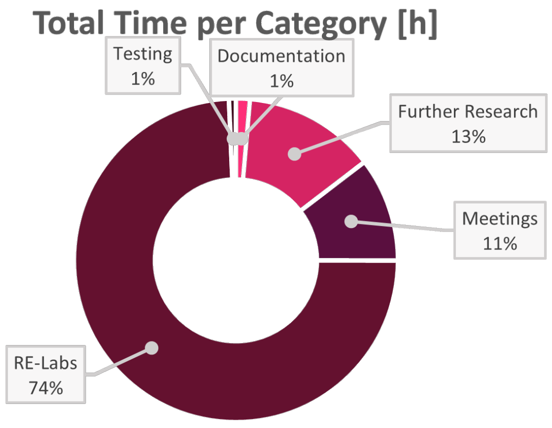


Figure C.10: Chart of the sprint 10 time distribution

Sprint 11: Update and Fix Challenges

This sprint, like the other sprints in the construction phase, is used to plan and construct challenges. More time is also spent in this sprint in meetings, as the advisor meeting discusses the concept of project testing.

Start Date:	01.05 - 07.05
RUP Phase:	Construction Phase
Open Issues:	0

Figure C.11 looks as expected for a construction sprint: Most of the time is spent working on the challenges and in meetings to discuss the future.

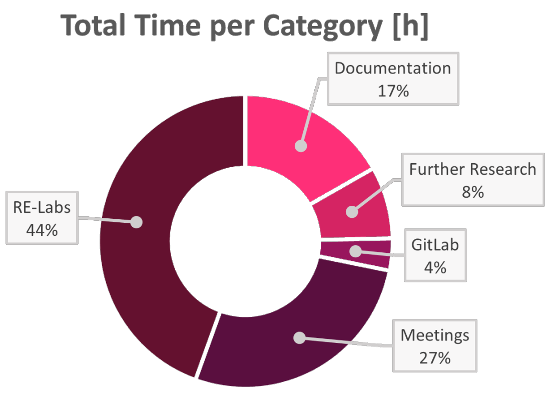


Figure C.11: Chart of the sprint 11 time distribution

Sprint 12: Hooking and Anti Debug

This sprint is used to complete the two challenges ”Hooking with Frida” and ”Anti-Debugging Techniques”. In addition, it is used to plan ahead and decide on the final challenges to be created.

Start Date:	08.05 - 14.05
RUP Phase:	Construction Phase
Open Issues:	0

This sprint is more balanced across the categories, as shown in Figure C.12. This is due to the multiple facets that need to be organised and planned in this sprint.

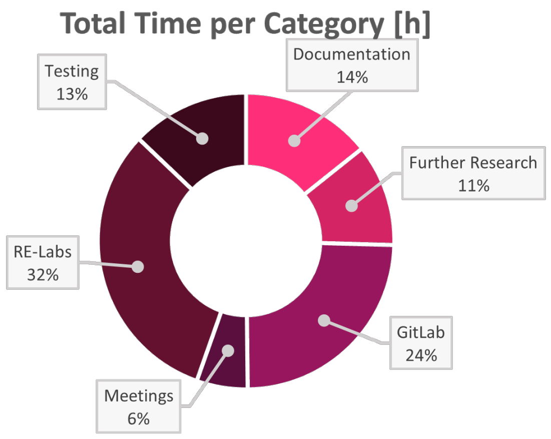


Figure C.12: Chart of the sprint 12 time distribution

Sprint 13: Testing Challenges

This sprint is used to finish testing all the challenges which aren't tested yet. In addition to that, the creation of the "ROP Chaining" challenge started and the planning of the final combination challenge started.

Start Date: 15.05 - 21.05

RUP Phase: Construction Phase

Open Issues: 0

Compared to other sprints, less time was spent building challenges and more time was spent finishing merge requests and other aspects of documentation and challenge creation. This is shown in Figure C.13.

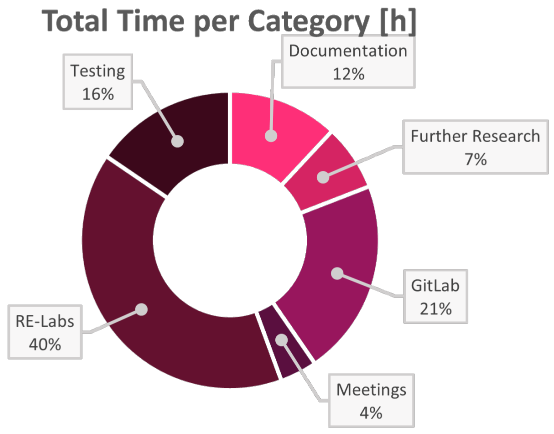


Figure C.13: Chart of the sprint 13 time distribution

Sprint 14: End of Construction Phase

This is the final sprint of the construction phase. It is used to finish the problems of the previous challenges and to start the creation of the final challenge.

Start Date: 22.05 - 28.05

RUP Phase: Construction Phase

Open Issues: 0

In this sprint, a bit more than half of the time was spent on the challenges. This is also shown in Figure C.14. The students which weren't occupied with challenge creation worked on preparing the documentation and GitLab for the Transition phase.

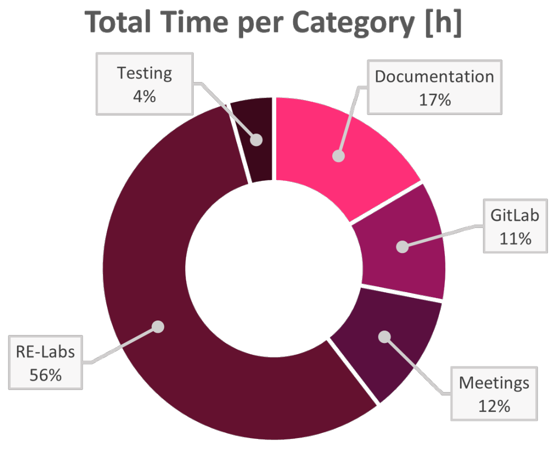


Figure C.14: Chart of the sprint 14 time distribution

Sprint 15: Start of Transition Phase

This sprint is used to complete the challenges and internal testing. The finished challenges are also given to students and other volunteer testers. To ensure that the final challenge is up to standard, this sprint is also used to finish it.

Start Date:	29.05 - 04.06
RUP Phase:	Transition Phase
Open Issues:	1

The start of the transition phase was not intended to have a high amount of challenge work to be done. To ensure that there were as few problems as possible, focus was set on updating according to the testers' feedback. In addition, work had to be completed for the final challenge to go live. This resulted in most time being spent in the challenge category as shown in Figure C.15.

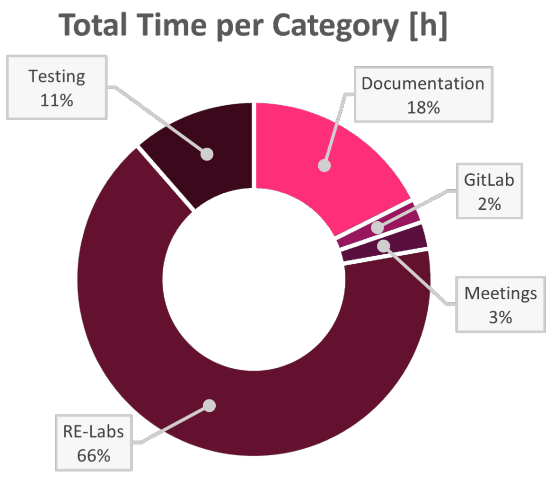


Figure C.15: Chart of the sprint 15 time distribution

Sprint 16: Finish Documentation

This sprint is the last full week of the project. As such, it will be used to finalise much of the documentation, including the analysis of the test feedback.

Start Date:	05.06 - 11.06
RUP Phase:	Transition Phase
Open Issues:	0

The last sprint before the hand-in week was mainly used to test and update all the challenges and to finish the documentation. This distribution is also shown in Figure C.16.

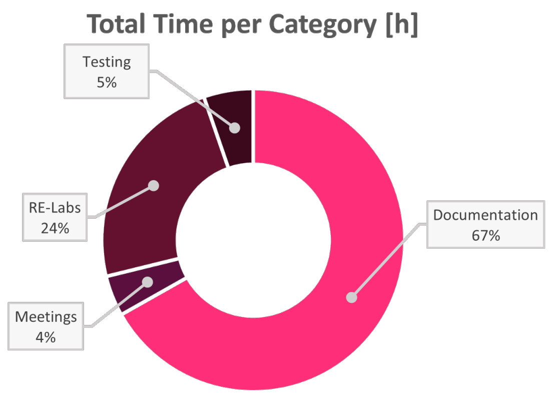
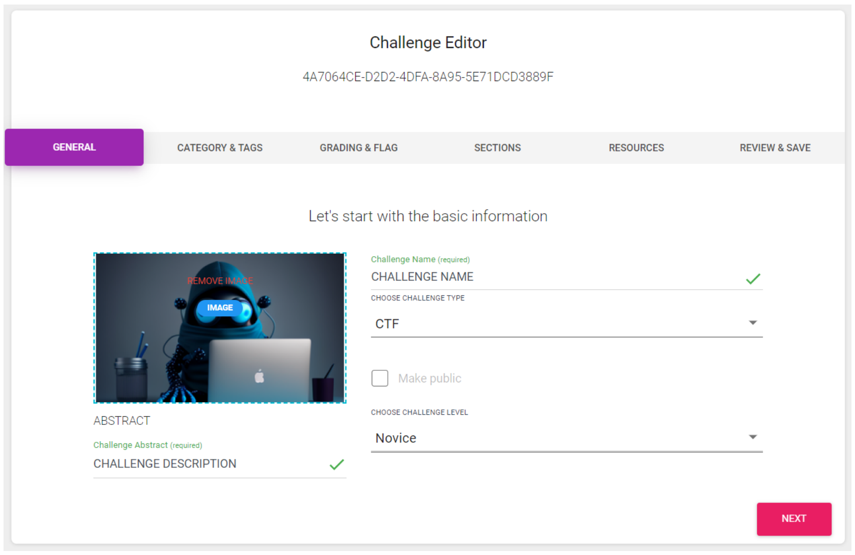


Figure C.16: Chart of the sprint 16 time distribution

Appendix D

Screenshots

Hacking-Lab


Challenge Editor

4A7064CE-D2D2-4DFA-8A95-5E71DCD3889F

GENERAL | CATEGORY & TAGS | GRADING & FLAG | SECTIONS | RESOURCES | REVIEW & SAVE

Let's start with the basic information

Challenge Name (required): CHALLENGE NAME

CHOOSE CHALLENGE TYPE: CTF

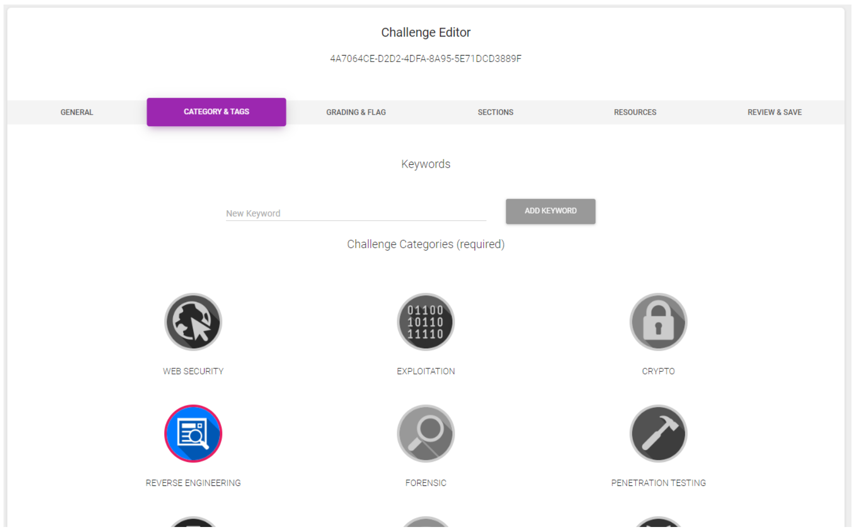
Make public

CHOOSE CHALLENGE LEVEL: Novice

ABSTRACT

Challenge Abstract (required): CHALLENGE DESCRIPTION

Figure D.1: Challenge Editor - General


Challenge Editor

4A7064CE-D2D2-4DFA-8A95-5E71DCD3889F

GENERAL | CATEGORY & TAGS | GRADING & FLAG | SECTIONS | RESOURCES | REVIEW & SAVE

Keywords

New Keyword

Challenge Categories (required)

Figure D.2: Challenge Editor - Categories

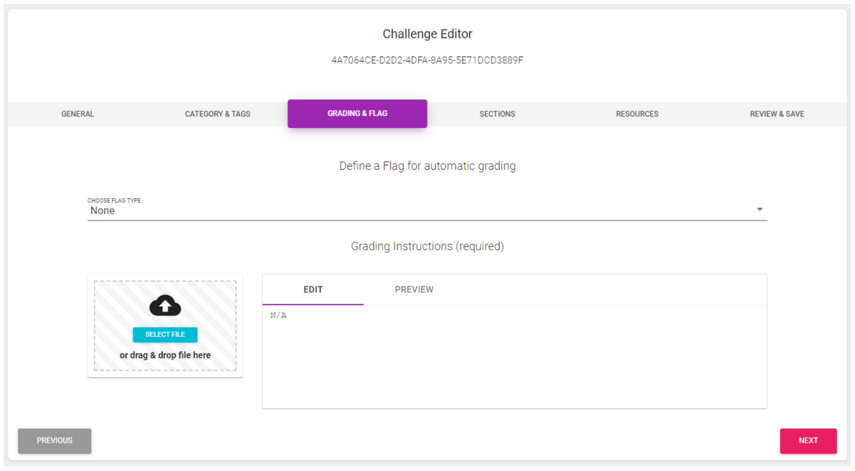


Figure D.3: Challenge Editor - Grading

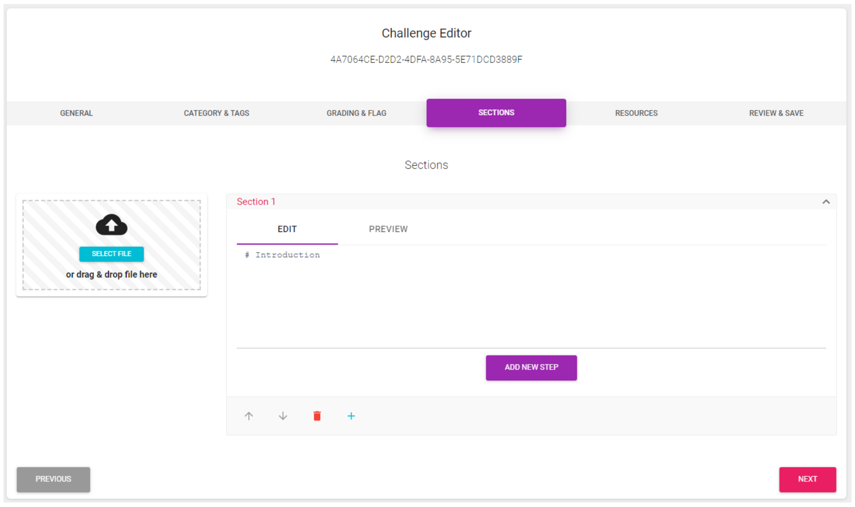


Figure D.4: Challenge Editor - Sections

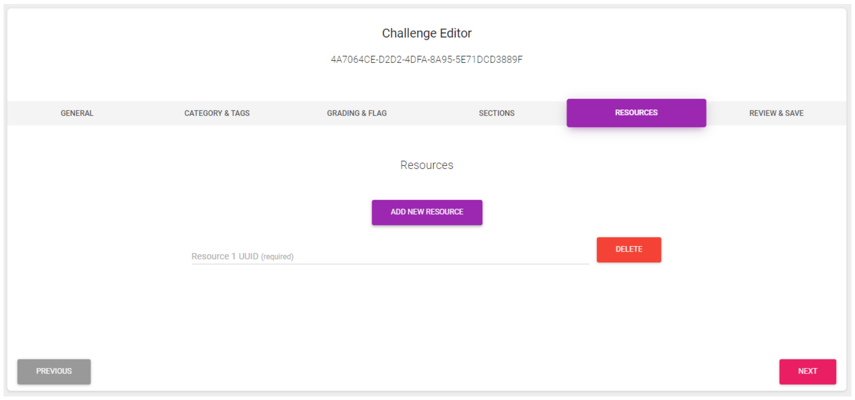
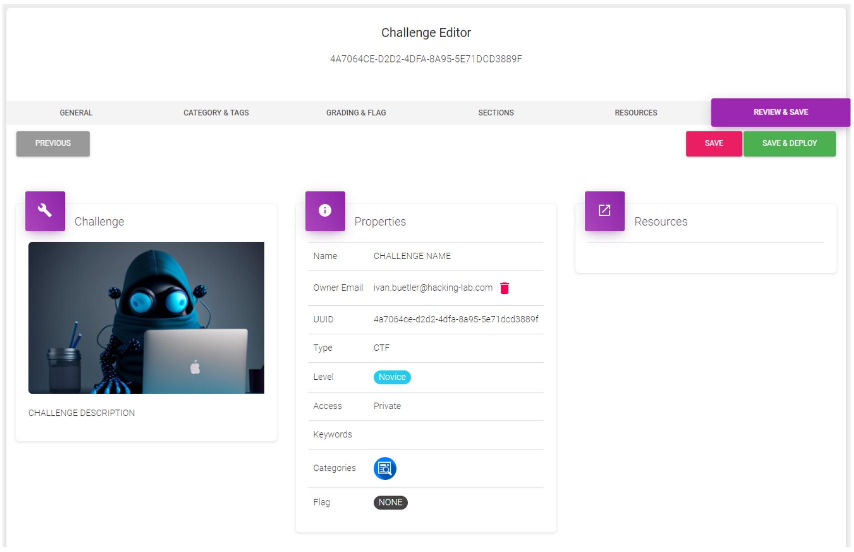


Figure D.5: Challenge Editor - Resources

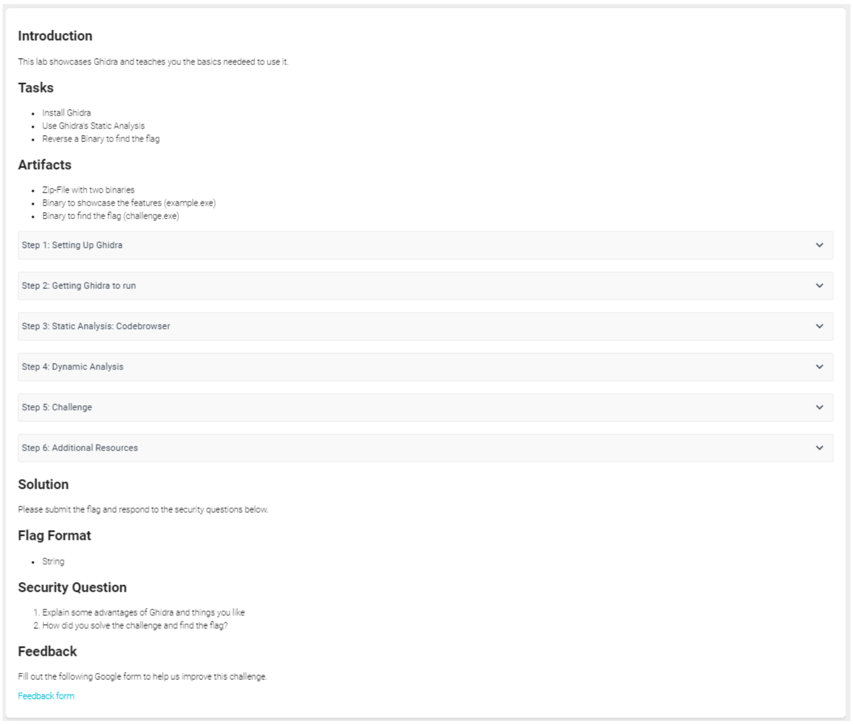


Name	CHALLENGE NAME
Owner Email	ivan.buetler@hacking-lab.com
UUID	4a7064ce-d2d2-4dfa-8a95-5e71dcd3889f
Type	CTF
Level	Novice
Access	Private
Keywords	
Categories	
Flag	NONE

Figure D.6: Challenge Editor - Review

Challenge Sections

Ghidra Introduction



Introduction

This lab showcases Ghidra and teaches you the basics needeed to use it.

Tasks

- Install Ghidra
- Use Ghidra's Static Analysis
- Reverse a Binary to find the flag

Artifacts

- Zip-File with two binaries
- Binary to showcase the features (example.exe)
- Binary to find the flag (challenge.exe)

Step 1: Setting Up Ghidra

Step 2: Getting Ghidra to run

Step 3: Static Analysis: Codebrowser

Step 4: Dynamic Analysis

Step 5: Challenge

Step 6: Additional Resources

Solution

Please submit the flag and respond to the security questions below.

Flag Format

- String

Security Question

1. Explain some advantages of Ghidra and things you like
2. How did you solve the challenge and find the flag?

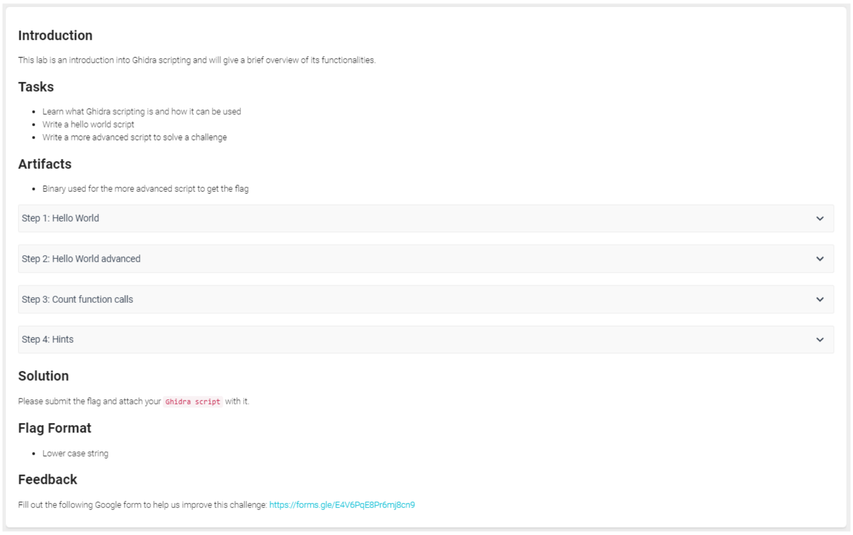
Feedback

Fill out the following Google form to help us improve this challenge.

Feedback form

Figure D.7: Challenge: Ghidra Introduction - Sections

Ghidra Scripting Introduction



Introduction

This lab is an introduction into Ghidra scripting and will give a brief overview of its functionalities.

Tasks

- Learn what Ghidra scripting is and how it can be used
- Write a hello world script
- Write a more advanced script to solve a challenge

Artifacts

- Binary used for the more advanced script to get the flag

Step 1: Hello World

Step 2: Hello World advanced

Step 3: Count function calls

Step 4: Hints

Solution

Please submit the flag and attach your `Ghidra script` with it.

Flag Format

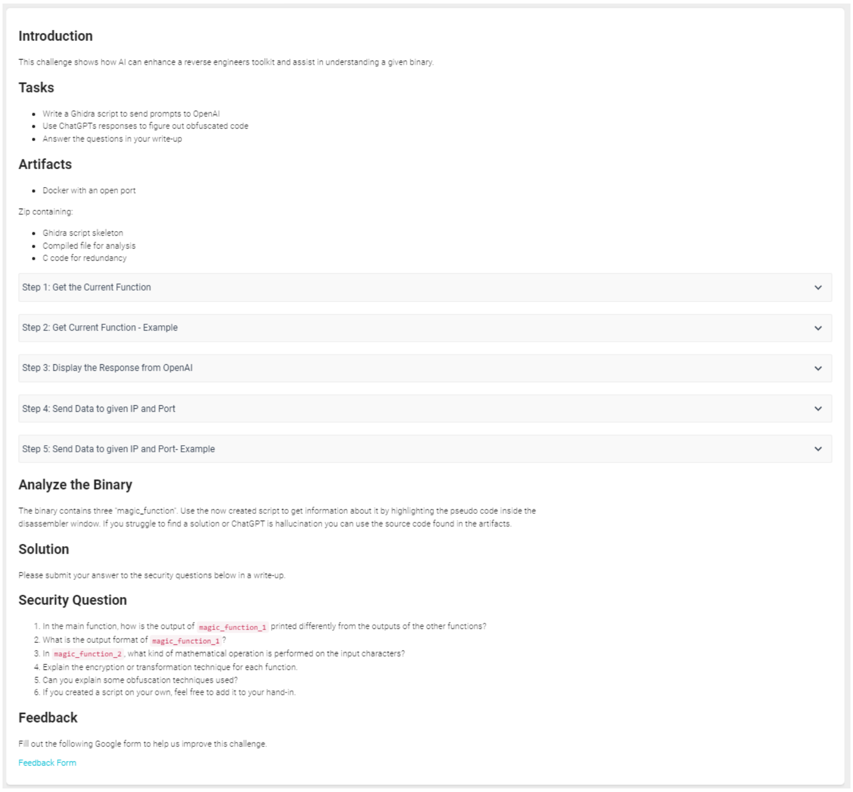
- Lower case string

Feedback

Fill out the following Google form to help us improve this challenge: https://forms.gle/E4V6PqE8Pr6mj8cn9

Figure D.8: Challenge: Ghidra Scripting Introduction - Sections

Ghidra GPT



Introduction

This challenge shows how AI can enhance a reverse engineers toolkit and assist in understanding a given binary.

Tasks

- Write a Ghidra script to send prompts to OpenAI
- Use ChatGPTs responses to figure out obfuscated code
- Answer the questions in your write-up

Artifacts

- Docker with an open port

Zip containing:

- Ghidra script skeleton
- Compiled file for analysis
- C code for redundancy

Step 1: Get the Current Function

Step 2: Get Current Function - Example

Step 3: Display the Response from OpenAI

Step 4: Send Data to given IP and Port

Step 5: Send Data to given IP and Port- Example

Analyze the Binary

The binary contains three "magic_function". Use the now created script to get information about it by highlighting the pseudo code inside the disassembler window. If you struggle to find a solution or ChatGPT is hallucination you can use the source code found in the artifacts.

Solution

Please submit your answer to the security questions below in a write-up.

Security Question

1. In the main function, how is the output of `magic_function_1` printed differently from the outputs of the other functions?
2. What is the output format of `magic_function_1` ?
3. In `magic_function_2`, what kind of mathematical operation is performed on the input characters?
4. Explain the encryption or transformation technique for each function.
5. Can you explain some obfuscation techniques used?
6. If you created a script on your own, feel free to add it to your hand-in.

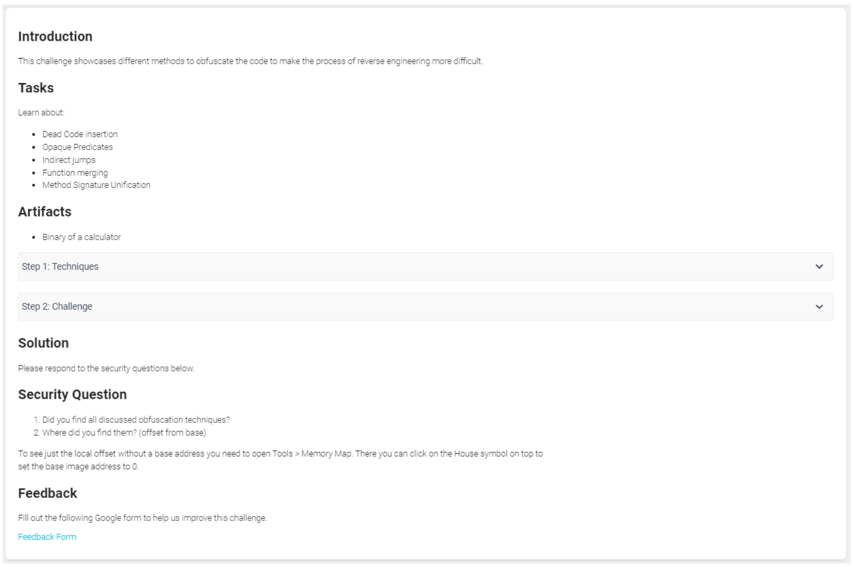
Feedback

Fill out the following Google form to help us improve this challenge.

Feedback Form

Figure D.9: Challenge: Ghidra GPT - Sections

Advanced Obfuscation



Introduction

This challenge showcases different methods to obfuscate the code to make the process of reverse engineering more difficult.

Tasks

Learn about:

- Dead Code insertion
- Opaque Predicates
- Indirect jumps
- Function merging
- Method Signature Unification

Artifacts

- Binary of a calculator

Step 1: Techniques

Step 2: Challenge

Solution

Please respond to the security questions below.

Security Question

1. Did you find all discussed obfuscation techniques?
2. Where did you find them? (offset from base)

To see just the local offset without a base address you need to open Tools > Memory Map. There you can click on the House symbol on top to set the base image address to 0.

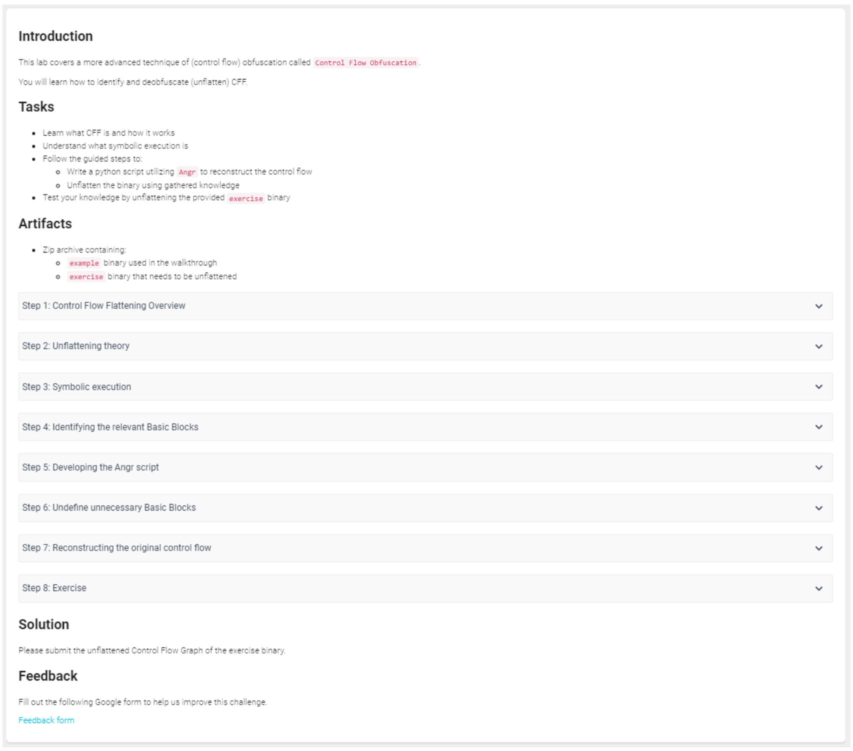
Feedback

Fill out the following Google form to help us improve this challenge.

Feedback Form

Figure D.10: Challenge: Advanced Obfuscation - Sections

Control Flow Flattening



Introduction

This lab covers a more advanced technique of (control flow) obfuscation called `Control Flow Obfuscation`.

You will learn how to identify and deobfuscate (unflatten) CFF.

Tasks

- Learn what CFF is and how it works
- Understand what symbolic execution is
- Follow the guided steps to:
  - Write a python script utilizing `Angr` to reconstruct the control flow
  - Unflatten the binary using gathered knowledge
- Test your knowledge by unflattening the provided `exercise` binary

Artifacts

- Zip archive containing:
  - `example` binary used in the walkthrough
  - `exercise` binary that needs to be unflattened

Step 1: Control Flow Flattening Overview
Step 2: Unflattening theory
Step 3: Symbolic execution
Step 4: Identifying the relevant Basic Blocks
Step 5: Developing the Angr script
Step 6: Undefine unnecessary Basic Blocks
Step 7: Reconstructing the original control flow
Step 8: Exercise

Solution

Please submit the unflattened Control Flow Graph of the exercise binary.

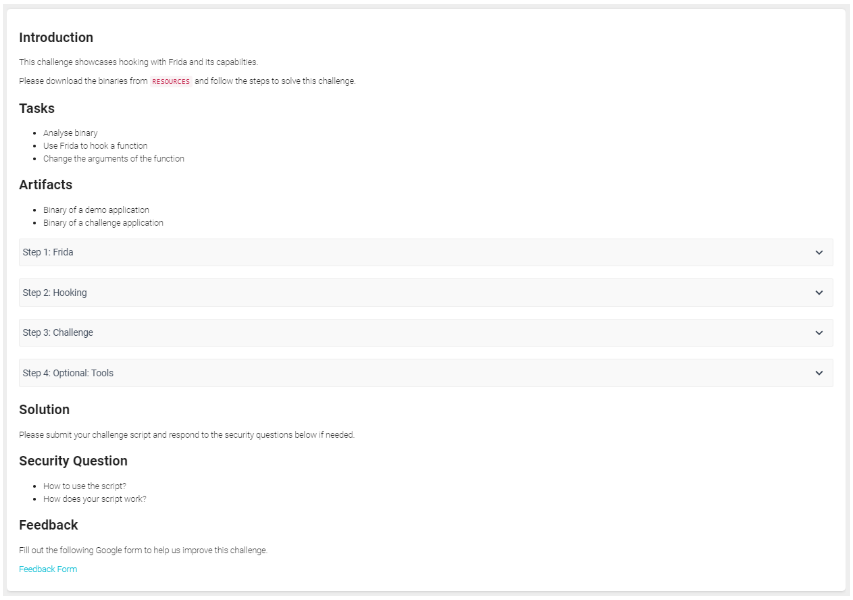
Feedback

Fill out the following Google form to help us improve this challenge.

Feedback form

Figure D.11: Challenge: Control Flow Flattening - Sections

Hooking with Frida



Introduction

This challenge showcases hooking with Frida and its capabilties.

Please download the binaries from `RESOURCES` and follow the steps to solve this challenge.

Tasks

- Analyse binary
- Use Frida to hook a function
- Change the arguments of the function

Artifacts

- Binary of a demo application
- Binary of a challenge application

Solution

Please submit your challenge script and respond to the security questions below if needed.

Security Question

- How to use the script?
- How does your script work?

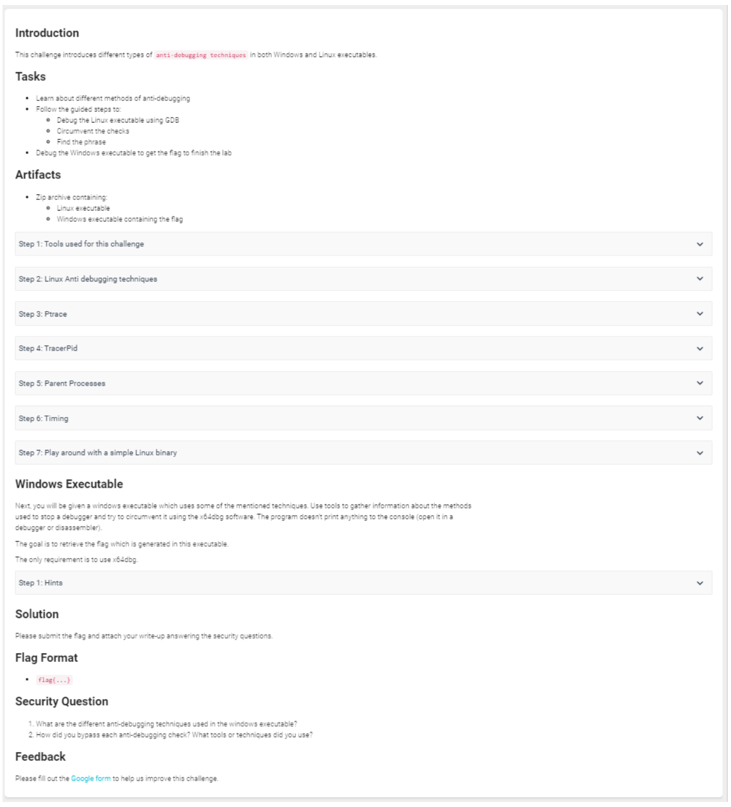
Feedback

Fill out the following Google form to help us improve this challenge.

Feedback Form

Figure D.12: Challenge: Hooking with Frida - Sections

Anti-Debugging Techniques



Introduction

This challenge introduces different types of `anti-debugging techniques` in both Windows and Linux executables.

Tasks

- Learn about different methods of anti-debugging
- Follow the guided steps to:
    - Debug the Linux executable using GDB
    - Circumvent the checks
    - Find the phrase
- Debug the Windows executable to get the flag to finish the lab

Artifacts

- Zip archive containing:
    - Linux executable
    - Windows executable containing the flag

Step 1: Tools used for this challenge

Step 2: Linux Anti debugging techniques

Step 3: Ptrace

Step 4: TracerPid

Step 5: Parent Processes

Step 6: Timing

Step 7: Play around with a simple Linux binary

Windows Executable

Next, you will be given a windows executable which uses some of the mentioned techniques. Use tools to gather information about the methods used to stop a debugger and try to circumvent it using the x64dbg software. The program doesn't print anything to the console (open it in a debugger or disassembler).

The goal is to retrieve the flag which is generated in this executable.

The only requirement is to use x64dbg.

Step 1: Hints

Solution

Please submit the flag and attach your write-up answering the security questions.

Flag Format

- `flag{...}`

Security Question

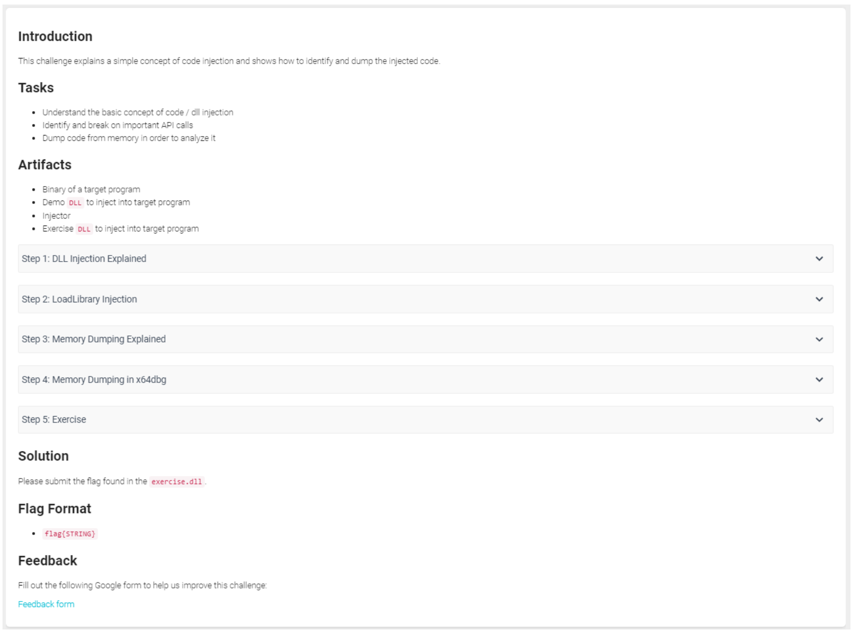
1. What are the different anti-debugging techniques used in the windows executable?
2. How did you bypass each anti-debugging check? What tools or techniques did you use?

Feedback

Please fill out the Google form to help us improve this challenge.

Figure D.13: Challenge: Anti-Debugging Techniques - Sections

Memory Dumping



Introduction

This challenge explains a simple concept of code injection and shows how to identify and dump the injected code.

Tasks

- Understand the basic concept of code / dll injection
- Identify and break on important API calls
- Dump code from memory in order to analyze it

Artifacts

- Binary of a target program
- Demo `DLL` to inject into target program
- Injector
- Exercise `DLL` to inject into target program

Step 1: DLL Injection Explained

Step 2: LoadLibrary Injection

Step 3: Memory Dumping Explained

Step 4: Memory Dumping in x64dbg

Step 5: Exercise

Solution

Please submit the flag found in the `exercise.dll`

Flag Format

- `flag{STRING}`

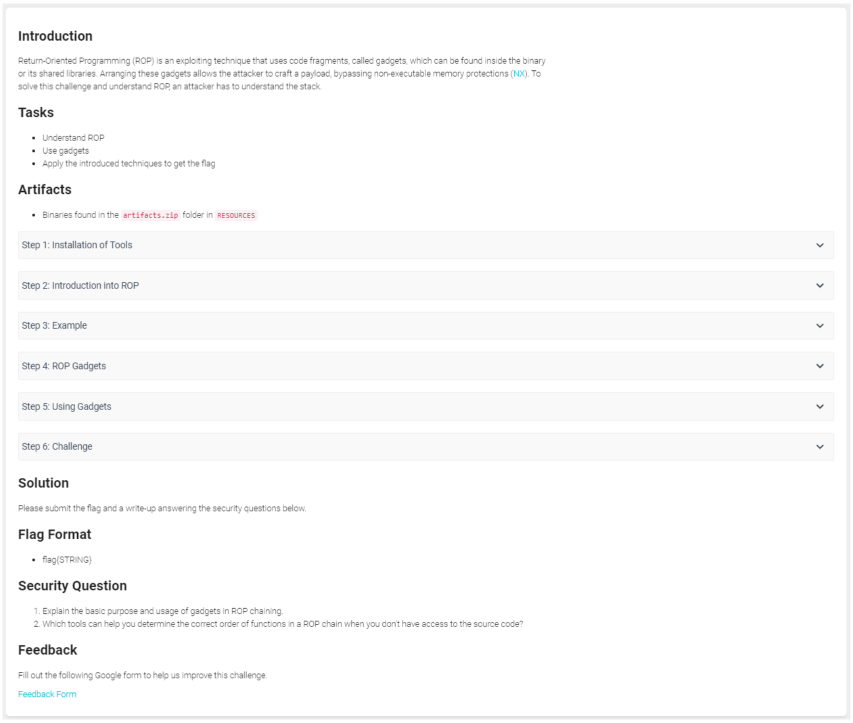
Feedback

Fill out the following Google form to help us improve this challenge:

Feedback form

Figure D.14: Challenge: Memory Dumping - Sections

ROP Chaining



Introduction

Return-Oriented Programming (ROP) is an exploiting technique that uses code fragments, called gadgets, which can be found inside the binary or its shared libraries. Arranging these gadgets allows the attacker to craft a payload, bypassing non-executable memory protections (NX). To solve this challenge and understand ROP, an attacker has to understand the stack.

Tasks

- Understand ROP
- Use gadgets
- Apply the introduced techniques to get the flag

Artifacts

- Binaries found in the `artifacts.zip` folder in `RESOURCES`

Step 1: Installation of Tools

Step 2: Introduction into ROP

Step 3: Example

Step 4: ROP Gadgets

Step 5: Using Gadgets

Step 6: Challenge

Solution

Please submit the flag and a write-up answering the security questions below.

Flag Format

- flag{STRING}

Security Question

1. Explain the basic purpose and usage of gadgets in ROP chaining.
2. Which tools can help you determine the correct order of functions in a ROP chain when you don't have access to the source code?

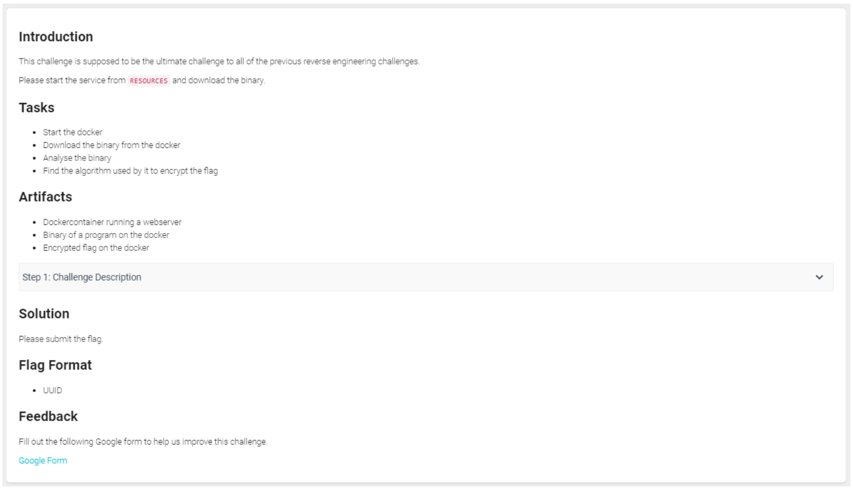
Feedback

Fill out the following Google form to help us improve this challenge.

Feedback Form

Figure D.15: Challenge: ROP Chaining - Sections

The BMI Incident



Introduction

This challenge is supposed to be the ultimate challenge to all of the previous reverse engineering challenges.

Please start the service from `RESOURCES` and download the binary.

Tasks

- Start the docker
- Download the binary from the docker
- Analyse the binary
- Find the algorithm used by it to encrypt the flag

Artifacts

- Dockercontainer running a webserver
- Binary of a program on the docker
- Encrypted flag on the docker

Step 1: Challenge Description

Solution

Please submit the flag.

Flag Format

- UUID

Feedback

Fill out the following Google form to help us improve this challenge.

Google Form

Figure D.16: Challenge: The BMI Incident - Sections

Appendix E

Gantt Diagram

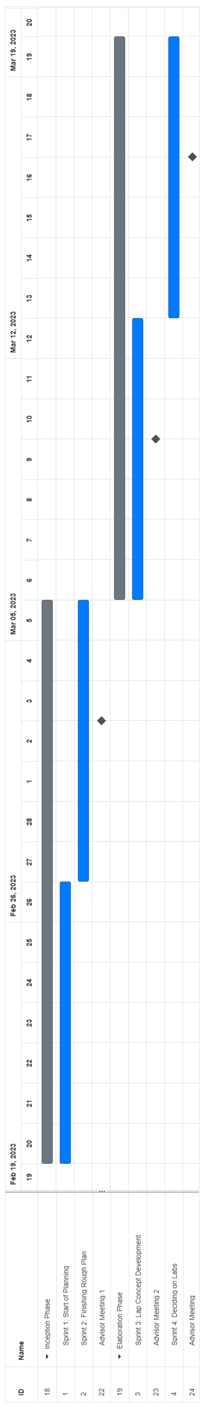


Figure E.1: Whole Gantt diagram - Part 1

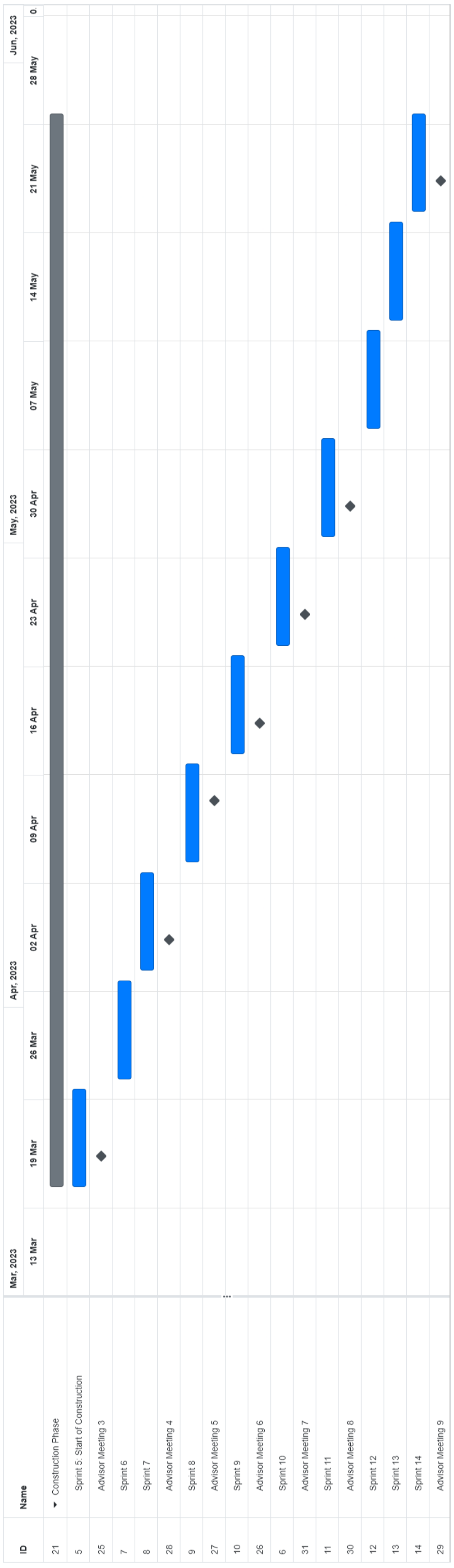


Figure E.2: Whole Gantt diagram - Part 2

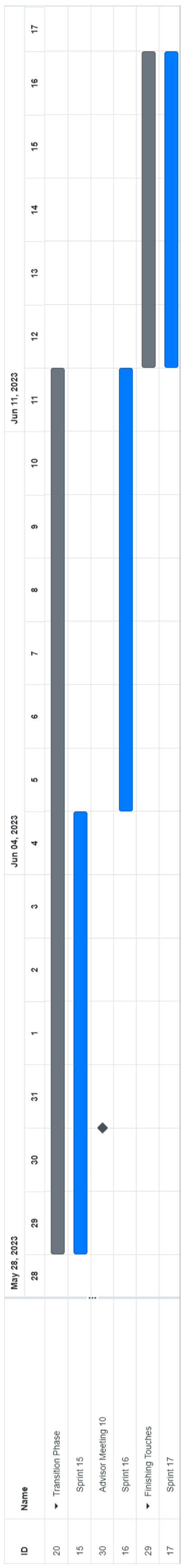
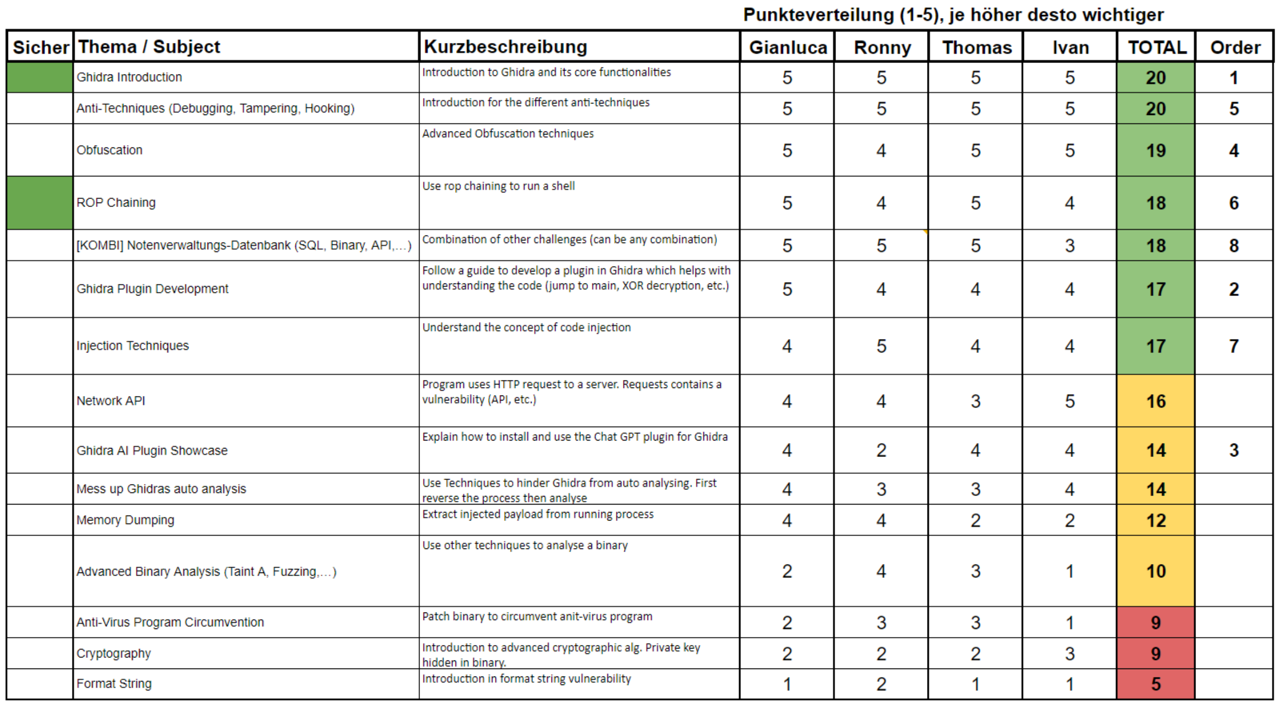


Figure E.3: Whole Gantt diagram - Part 3

Appendix F

Challenge Identification



Punkteverteilung (1-5), je höher desto wichtiger

Sicher	Thema / Subject	Kurzbeschreibung	Gianluca	Ronny	Thomas	Ivan	TOTAL	Order
🟩	Ghidra Introduction	Introduction to Ghidra and its core functionalities	5	5	5	5	20	1
	Anti-Techniques (Debugging, Tampering, Hooking)	Introduction for the different anti-techniques	5	5	5	5	20	5
	Obfuscation	Advanced Obfuscation techniques	5	4	5	5	19	4
🟩	ROP Chaining	Use rop chaining to run a shell	5	4	5	4	18	6
	[KOMBI] Notenverwaltungs-Datenbank (SQL, Binary, API,...)	Combination of other challenges (can be any combination)	5	5	5	3	18	8
	Ghidra Plugin Development	Follow a guide to develop a plugin in Ghidra which helps with understanding the code (jump to main, XOR decryption, etc.)	5	4	4	4	17	2
	Injection Techniques	Understand the concept of code injection	4	5	4	4	17	7
	Network API	Program uses HTTP request to a server. Requests contains a vulnerability (API, etc.)	4	4	3	5	16	
	Ghidra AI Plugin Showcase	Explain how to install and use the Chat GPT plugin for Ghidra	4	2	4	4	14	3
	Mess up Ghidras auto analysis	Use Techniques to hinder Ghidra from auto analysing. First reverse the process then analyse	4	3	3	4	14	
	Memory Dumping	Extract injected payload from running process	4	4	2	2	12	
	Advanced Binary Analysis (Taint A, Fuzzing,...)	Use other techniques to analyse a binary	2	4	3	1	10	
	Anti-Virus Program Circumvention	Patch binary to circumvent anit-virus program	2	3	3	1	9	
	Cryptography	Introduction to advanced cryptographic alg. Private key hidden in binary.	2	2	2	3	9	
	Format String	Introduction in format string vulnerability	1	2	1	1	5	

Figure F.1: Brainstormed challenge topics