CYBERSECURITY PEDAGOGY & PRACTICE JOURNAL

Volume 2, No. 2
September 2023
ISSN: 2832-1006

In this issue:

iscap
Information Systems & Computing Academic Professionals

The **Cybersecurity Pedagogy and Practice Journal** (**CPPJ**) is a double-blind peer-reviewed academic journal published by **ISCAP** (Information Systems and Computing Academic Professionals). Publishing frequency is two times per year. The first year of publication was 2022.

CPPJ is published online (https://cppj.info). Our sister publication, the proceedings of the ISCAP Conference (https://proc.iscap.info) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the ISCAP conference. At that point, papers are divided into award papers (top 15%), and other accepted proceedings papers. The other accepted proceedings papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the CPPJ journal.

While the primary path to journal publication is through the ISCAP conference, CPPJ does accept direct submissions at https://iscap.us/papers. Direct submissions are subjected to a double-blind peer review process, where reviewers do not know the names and affiliations of paper authors, and paper authors do not know the names and affiliations of reviewers. All submissions (articles, teaching tips, and teaching cases & notes) to the journal will be refereed by a rigorous evaluation process involving at least three blind reviews by qualified academic, industrial, or governmental computing professionals. Submissions will be judged not only on the suitability of the content but also on the readability and clarity of the prose.

Currently, the acceptance rate for the journal is under 35%.

Questions should be addressed to the editor at editorcppj@iscap.us or the publisher at publisher@iscap.us. Special thanks to members of ISCAP who perform the editorial and review processes for CPPJ.

CYBERSECURITY PEDAGOGY AND PRACTICE JOURNAL

Editors

2023 Review Board

Utilizing Discord-based Projects to Reinforce Cybersecurity Concepts

Marc Waldman
marc.waldman@manhattan.edu

Patricia Sheridan
patricia.sheridan@manhattan.edu

Computer Information Systems and Business Analytics Department
O'Malley School of Business
Manhattan College
Riverdale, NY 10471, USA

Abstract

This paper describes two Discord-based undergraduate student projects that were created to reinforce cybersecurity concepts covered in a data privacy course and an introduction to cloud computing course. The first project, created for the data privacy course, is used to provide students with a practical and "hands-on" project to utilize role-based access control (RBAC). This course was developed primarily for business students who do not necessarily have the information technology background possessed by computer information systems or computer science students. The second project, created for an introduction to cloud computing course, is used to illustrate the basic function of a botnet and incorporates a few other cybersecurity topics. We believe these projects represent just the tip of the iceberg of how Discord can be utilized as a source of course projects. Additional Discord-based project ideas are described at the end of the paper.

Keywords: information systems education, cybersecurity projects, discord, data privacy, role-based access control (RBAC), bot programming, botnets

1. INTRODUCTION

Discord is a free voice, video, and text communication platform that was initially designed to facilitate communication between video game players (Discord, 2023). Since its launch in 2015, the platform has also been widely adopted by non-gamers seeking to create shared-interest virtual communities. Since the COVID-19 pandemic, it has also been used as a tool in the delivery of online courses - being one of the few easy-to-use free platforms that combines text-based chat, direct messaging, screen sharing, and both voice and video conferencing for multiple concurrent users.

Much has been written about utilizing Discord's sharing and communication features as a tool to facilitate online education (Grimbsy, 2019; Wiles & Simmons, 2022). However, the Discord ecosystem (software, community, content) can also be used as a source of course projects. We describe two cybersecurity-related Discord-based student projects that we created for a data privacy course and an introduction to cloud computing course. Both courses are offered in our Computer Information Systems (CIS) Department, which is housed within the School of Business. All project-related materials are available for download (https://github.com/marcwaldman/discord_rbac_botnet).

2. DISCORD

The Discord platform allows users to navigate in and out of virtual communities. Each community is referred to as a "server" or a "guild". Each server consists of one or more multimedia-rich discussion "channels" that allow participants to talk, in real-time, using voice, text chat, or video conferencing.

Channels are voice or text-based "virtual rooms" set aside for the specified type of communication. Voice channels also support video and screen sharing. Multiple named channels can exist within a single server and can optionally be grouped under an umbrella name called a "category". For example, a Discord server for individuals interested in computer programming might have a "Java" and a "Python" category, each containing a group of channels related to topics within the particular programming language.

Figure 1 shows a very basic Discord client user screen. Individual text and voice channels can be selected from the left-hand side listing. In this case, the "welcome" channel has been selected. The two categories appearing on this screen are "TEXT CHANNELS" and "VOICE CHANNELS". The chat area is located to the right of the channel listing - messages from users that have posted to this channel will appear here. Older messages can be read by scrolling up; newer messages appear at the bottom of the channel. This channel contains only two messages. Both text and multimedia (emojis, pictures, and video clips) can be posted to text channels.

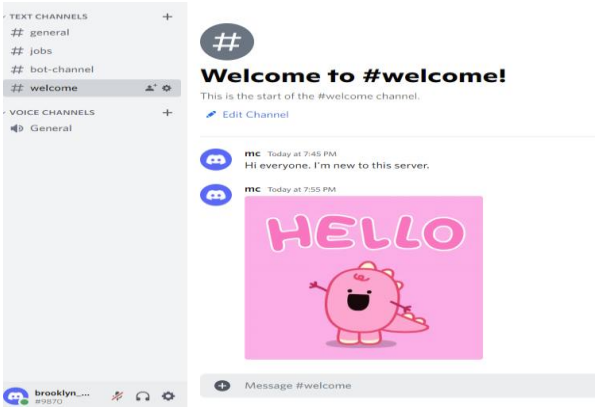

Figure 1: Discord Client User Screen

The Discord software, referred to as the Discord client, is available for all major desktop and phone-based operating systems. In addition, one can access Discord via a web-based front-end. Note that the Discord platform itself is centralized and hosted on Discord company hardware. Users and developers simply interact with the platform via the client software or its application programmer interface (API).

The Discord client provides a relatively intuitive graphical user interface that allows one to join, create, or administer servers. No programming or system administration experience is needed. With the click of a button, a server can be created and pre-populated with default text and voice channels. All channels can be changed or removed by the server's creator.

In addition to creating servers, one can join other pre-existing servers. Servers can be configured to allow anyone to join or require an invite to join. An invite is essentially a server-specific URL generated by Discord. These options are controlled by the server's creator, who is, by default, considered the server's administrator.

A Discord-provided search engine allows individuals to find servers catering to specific interests. Web-accessible third-party search services and listings are also available.

Permissions and Roles

Popular Discord servers may have many thousands of users. Administering and policing members in channels and other Discord objects is facilitated by permissions and roles. Permissions are assigned to grant access to Discord objects (e.g. a text channel) or to allow specific actions such as the ability to delete any post within a channel. Roles can be thought of as a named alias for a group of permissions. Anyone who joins a server can be assigned to one or more roles. The default role is the "everybody" role - permissions granted to this role apply to all users of the server. Permissions or roles can be assigned to users by the server's administrator.

Certain permissions, such as removing a message posted by another user, are typically reserved solely for the administrator. However, the administrator can assign these permissions, or a role incorporating these permissions, to other users - allowing the moderation workload to be shared among special "deputized" members.

Why Discord?

Several Discord competitors exist, including the well-known Slack and Microsoft Teams applications. When evaluating the alternatives, we were looking for an application that had the following:

- Wide adoption and utilization.
- Freely available for all operating systems.

- Allowed the creation of "virtual communities" that kept a persistent record of public text-based messages that could later be viewed by community members who were not present at the time the messages were posted.
- Allowed the creation of "bots" -programs that interact with the communications platform.
- Provided the ability to grant application-specific permissions or roles to different users of the system.

While the free-to-use tier of several of Discord's competitors, including Slack, do satisfy these requirements, Discord was chosen due to its perceived ease of use, popularity among college students (D'Agostino, 2023; Sukhanova, 2023), and generous limits on its free services.

The two projects described in this paper do not require any Discord-specific features. Therefore, the projects can be adapted to any platform offering the last three features outlined above.

Although originally developed for use by gamers, Discord has been widely (well over 100 million users) utilized and adopted as a platform to host virtual communities in areas as diverse as music appreciation and cryptocurrencies (Browning, 2021). There exists a rich support community for this platform as evidenced by the large collection of quality resources that can be readily accessed on YouTube or otherwise found via a search engine.

The authors, during the Spring 2022 and Spring 2023 semesters, utilized Discord-based student projects in two courses offered in our CIS department. The courses and the associated projects are described below.

3. DATA PRIVACY COURSE

This is a CIS elective course that is open to all students in the School of Business. Due to the importance of this subject matter, CIS majors are strongly encouraged to enroll. The course has a law-oriented focus, examining data privacy laws, policies, and issues related to data protection. There are no course prerequisites, but junior or senior standing is expected.

As the material covered in this course will likely be of interest to students in other majors, no formal prerequisites were added. While this lack of prerequisites expanded the audience for the course, it also limited the type of computer-based projects that could be assigned. Projects involving computer programming, system administration, or database development were not an option. Nevertheless, applied "hands-on" projects can greatly enhance student learning and show how in-course concepts can be applied in the "real world".

The important concepts of "roles" and "permissions" are covered in the course. These concepts are relatively easy to understand and are found in many aspects of life (society, school, work, etc.). However, it is only when one tries to apply these concepts to non-trivial problems does one realize the complexity of the underlying implementation issues (identity management, conflicting permissions, role revocation, etc.). Students majoring in information technology-related fields (Computer Science, Business Analytics, Information Systems, etc.) will likely be able to apply these concepts to implementable projects in areas such as database management or system administration. Fewer opportunities to practically apply these concepts exist for those with non-information technology backgrounds. A Discord-based project was developed to help fill this gap.

Role-Based Access Control

One of the reasons we chose the Discord platform was its ability to allow server owners to establish and grant roles to specific users. This provided us with the opportunity to create a student project that incorporated the concept of Role-Based Access Control (RBAC), a topic that is covered in the course. As defined in (Ferraiolo et al., 1995), "the central notion of Role-Based Access Control (RBAC) is that users do not have discretionary access to enterprise objects. Instead, access permissions are administratively associated with roles, and users are administratively made members of appropriate roles. Users can be made members of roles as determined by their responsibilities and qualifications and can be easily reassigned from one role to another without modifying the underlying access structure."

In more simplistic terms, a role can be thought of as an alias for a set of permissions, and each role can be assigned to, or revoked from, one or more users. Roles can be combined and built into hierarchies, forming new roles that provide incremental addition of new permissions. For example, within a company, everyone may be assigned the role of employee. The employee role has a limited set of access permissions to the company's Human Resources (HR) system. The manager role incorporates the employee role and includes additional access permissions for the HR system.

Multiple roles can also be assigned to a single user. Within RBAC systems there exist mechanisms that allow, possibly conflicting, permissions within different roles to be disambiguated and then applied. Discord, which supports the assignment and combination of multiple roles, has a rich set of permissions that can be applied to various objects including communication channels, messages, and users. The full set of permissions and disambiguation rules are described on the Discord Developer website (Discord Permissions, 2023).

Preparation and Project

Before being assigned the project, the students were required to complete a PowerPoint-based tutorial/exercise that led them through Discord setup and usage. The tutorial also covered the basics of server creation, permissions, and roles. In a subsequent class, the following project was assigned:

1) For this project you will create a Discord Server for a physical therapy office so that staff members and patients can exchange general health and wellness advice. Participants may include the office manager, receptionist, patients, etc.

    - The group wants to share nutrition tips, recipes, and ideas for healthy eating. They also want to recommend fitness workouts and outdoor trails to each other and suggest local shops and restaurants that cater to health-conscious individuals.

    - Some members of the office staff want to be able to monitor the server, delete inappropriate content or inadvertent disclosures of personal health information, and remove any members if necessary.

2) Create a Discord server for this purpose and submit a writeup that includes the following:

    - List of at least 6 roles that will be utilized by your server.

    - List of at least 6 channels that will be available to your members. Briefly describe the purpose of each channel. Some may be private – but specify which ones and which roles can access them.

    - List of at least three categories and which channels they will contain.

    - List of permissions assigned to each role. Briefly explain the reason for assigning such permissions.

    - Email the URL for your server to the instructor and add the instructor as an administrator.

The physical therapy scenario can be replaced by a similar one – doctor's office, law office, day-care center, etc. Another alternative is to offer students the opportunity to choose one of many scenarios or create their own scenario. Section 6 describes student feedback and survey results.

4. INTRODUCTION TO CLOUD COMPUTING COURSE

This is an upper sophomore/junior level course that is meant to introduce cloud-based applications and programming to students. It is required for all CIS majors. The only prerequisite is our one-term introduction to programming course which covers the Python programming language.

Over the past few years, communications platforms such as Microsoft Teams, Slack, and Discord have increasingly been adopted by organizations of all sizes. Students in our program will very likely utilize one or more of these platforms during an internship, part-time job, or as a member of a student club. These systems are cloud-based and allow the creation of programs, called bots, that can interact with users of the platform.

Discord API

Discord allows developers to create programs, usually referred to as bots, that can interact independently (don't need constant human monitoring) with the platform. Bots typically provide utility functionality such as generating a nightly server activity report, monitoring a chat stream for inadmissible language, or replying to simple questions posed by users. Bots that receive user-generated text and then reply, in real-time, with a meaningful text response, are frequently referred to as chatbots (Adamopoulou & Moussiades, 2020).

Discord, Slack, and Teams all provide an API that can be used to create bots. One does not need a programming background to add a bot to a server. Many types of pre-built bots exist, and they can be easily added by a server's administrator. Some of the most popular bots are those that automatically perform routine operations such as greeting new users or

reposting content from social media (Twitter, Instagram, etc.) to a specific channel. These bots are typically created by third parties and are not necessarily vetted or pre-approved by the Discord company. When a server administrator requests a bot be added to their server, the Discord server will reveal which permissions the bot is requesting. The administrator must use their discretion to decide if the bot should be allowed to run with the requested permissions.

The Discord API is based on REST (Fielding & Taylor, 2002) and WebSockets (Melnikov & Fette, 2011). However, Discord does not recommend using the API directly, as it is "complex and fairly unforgiving" (Discord API, 2023). Instead, bot developers typically utilize code libraries that provide an easier-to-use class and/or function-based interface to the API. The Discord company does not provide code libraries to utilize the API. Nevertheless, several well-supported, third-party, free, and open-source libraries are available for many programming languages. These code libraries abstract away many of the low-level details of the underlying communication between the user's program (the bot) and the server. Discord does provide a list of code libraries (Discord Code, 2023) that it has vetted for valid API rate-limit implementations. These limits, which place a cap on the number of API requests a bot can make per second, will likely not be of concern to students as the quotas are generous and are essentially meant to prevent abuse or service overloads by bots with a large user base.

Project Description

This course contains a cybersecurity component that provides a high-level overview of some of the cloud-relevant secure computing concepts outlined in the IS2020 Competency Model (Leidig & Salmela, 2020). To help reinforce and apply this material, a cybersecurity-related bot project was chosen instead of a chatbot. This project was inspired by a class lesson/discussion on cybercrime and botnets.

A botnet is a network of compromised computers, called "bots", that are under the remote control of a human operator who is referred to as the "botmaster" (Feily et al., 2009). So as not to confuse the reader with the two uses of the term "bot" (innocuous program interfacing with the Discord server vs. a compromised computer), the term "zombie" (Cooke et al., 2005) will be used to refer to the compromised computer that is part of a botnet.

There exists a wealth of literature on botnet history, uses, architecture, defense, and detection (Bailey et al., 2009; Cooke et al., 2005; Feily et al., 2009; Vormayr et al., 2017). Therefore, only the project-relevant aspects of botnets will be discussed in this paper.

A botnet can be thought of as a form of distributed computing. The botmaster issues commands to all participating computers (zombies) via some command and control (C&C) mechanism. The actual C&C mechanism depends on the level of sophistication of the botnet but can be as simple as an internet relay chat (IRC) channel or social media account (Pantic & Husain, 2015) or as complex as a peer-to-peer network hidden via the TOR anonymizing network (Casenove & Miraglia, 2014). Although this description of a botnet may make it appear to be rather benign, the term botnet has a strong association with cybercrime and other malicious activity.

Probably the most publicized and well-known use of a botnet is for distributed denial of service (DDOS) attacks against websites or other Internet-based services (Hoque et al., 2015). These attacks inundate a set of IP addresses, belonging to a targeted organization, with network traffic in an attempt to prevent or slow legitimate access to the services provided by the target. DDOS attacks have been utilized for things as trivial as eliminating an opponent from an online game to serious crimes such as online blackmail and critical service disruption. Other uses of botnets include sending spam and crypto-currency mining (Sood & Enbody, 2013).

A victim computer (zombie) almost always joins a botnet involuntarily. This usually occurs when the user of the victim computer is tricked into executing some malicious code via a phishing e-mail, instant message hyperlink, or other form of infection. Once infected with the malicious code the zombie typically contacts the botmaster via the C&C mechanism or other prearranged communication channel, and then awaits commands from the botmaster.

For this project, each student designed a Discord bot that functioned as a zombie – accepting commands issued by the botmaster (the student entering commands into Discord). Upon startup, the zombie first connects to Discord. This connection allows the zombie to contact the botmaster – Discord acts as the C&C mechanism. On initial connection to Discord, the bot sends its "ID" (a small randomly generated sequence of

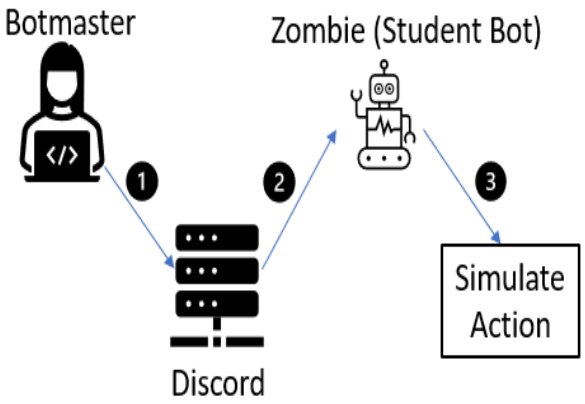

Figure 2: Botmaster to Zombie Communication

characters and numbers), an IP address, and operating system name (e.g. "Windows 10", "Android", etc.) to the botmaster. Both the IP address and operating system name are randomly generated – they are sent to Discord to illustrate the type of information collected by botmasters upon initial infection of the zombie. The zombie then awaits botmaster commands from the botmaster.

Figure 2 illustrates the botmaster to zombie communication process. The student (botmaster) issues commands. Each command is forwarded by Discord to the zombie (student-created bot), which is listening for botmaster commands in a specific channel (step 2). The zombie parses the command and then simulates the corresponding action (step 3).

Some of the botnet commands the students were required to simulate are listed in Table 1 (all commands can be found in the project documentation). Each command has one or more required arguments that are specified along with the command – just like the specification of arguments to shell commands. Arguments are separated from the command and each other by one or more spaces. For example, the botnet command "Traffic victim.com:80" instructs the zombie to simulate the sending of traffic to victim.com on port 80 (as part of a DDOS attack). The student-implemented zombie does not send network traffic (spam, etc.) to externally targeted services. Instead, to mimic the generation of malicious traffic, HTTP requests are sent to a local (on the student's computer) running instance of Python's bundled HTTP server. By default, all requests received by the HTTP server are displayed in the HTTP server's terminal window – allowing the student to view the zombie-initiated requests as they are generated.

A sample botmaster and zombie interaction is shown in Figure 3. Upon startup, the zombie sends its randomly generated bot ID, IP Address, and operating system to the botmaster via Discord. The botmaster requests the zombie to send network traffic to victim.com on port 80. This is simulated by sending requests to a locally running HTTP server. Requests received by the HTTP server are displayed in a terminal window (right-hand side of Figure 3). A true DDOS attack would utilize many such zombies to inundate the targeted host with network traffic. The Findhash command, also shown on the left-hand side of Figure 3, asks the bot to search its locally cached list of commonly used passwords to "crack" (find a match for) the supplied SHA256 hash. In this case, the zombie found a match on the string "password123". Real botnets would likely utilize more advanced forms of hash cracking/matching (Dev, 2013).

Based on student programming and computer security background, the project can be expanded to include more sophisticated zombie behavior, coordination, and simulation of botnet

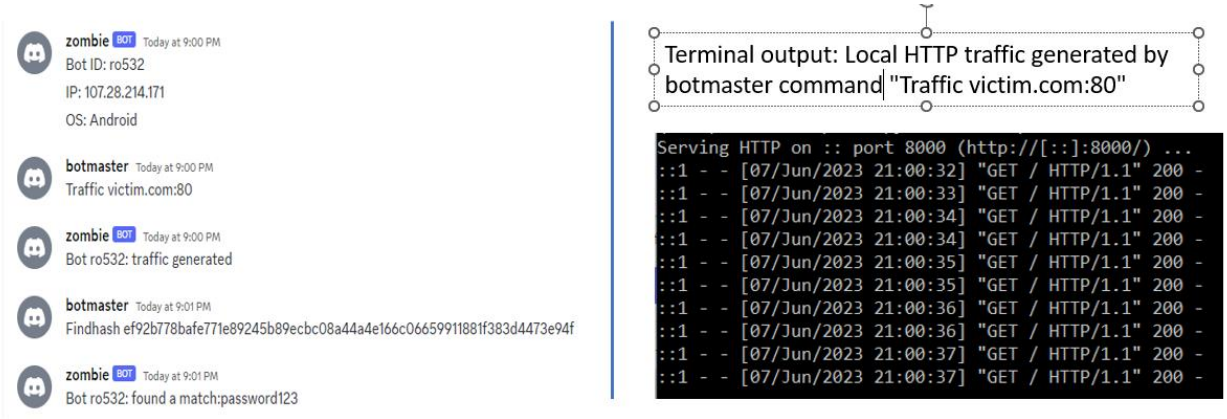

Figure 3:Sample botmaster and zombie interaction via Discord

Command	Argument(s)	Description
Update	URL	Simulate updating the zombie executable with the one specified by the URL.
Traffic	IP_Address:port or Hostname:port	Simulate a DDOS attack against the specified IP_Address (or hostname) and associated port.
Findhash	Cryptographic_Hash	Simulate the behavior of a zombie being assigned to password cracking or cryptographic hash matching.
Mail	Email_type, Email_address	Simulate a zombie sending phishing email of the specified type (banking, payment_request, password_reset, etc.) to the user with the specified Email_address.

Table 1: Sample Botnet Commands

commands. Possible enhancements, inspired by the botnet literature, include encrypted botmaster and zombie communication, using the TOR anonymizing network for botnet communication, channel or server hopping based on prearranged functions, and generation of "cover" traffic to hinder C&C mechanism discovery.

Although this project utilized only one zombie it is certainly possible to have multiple bots (zombies) join a Discord server, allowing for project additions such as those described in the previous paragraph.

Preparation

One course session, an hour and fifteen minutes in length, was devoted to introducing the topics and sample code needed for Discord bot creation. Before the start of this session, the students were required to complete, as a homework assignment, readings and exercises that introduced Discord. This is the same intro material that was utilized for the Data Privacy course project.

It is best to access the Discord API by utilizing one of the freely available, open-source, rate-limit vetted code libraries. For the Python programming language, several such libraries exist. By far the most popular (as measured by GitHub "stars") is discord.py (Rapptz, Danny, 2015/2023).

Figure 4 shows the code for a very basic bot that utilizes discord.py. This bot, once connected to a Discord server, will display a "Bot logged in as" message in the terminal window. The bot will then receive messages that users send to the Discord server. If the bot receives, via Discord, the string "$hello", it will respond to the message's author with a "Bot welcomes" greeting. Note that each

```
import discord

TOKEN="Copy Your Bot Token Here"

client = discord.Client()

@client.event
async def on_ready():                          # executed when bot logs into Discord
    print("Bot logged in as ",client.user)

@client.event
async def on_message(message):                 # executed when a message is received
    if message.author == client.user:          # don't respond to bot's own messages
        return

    if message.content=="$hello":              # welcome user who sent a $hello message
        await message.channel.send("Bot welcomes "+message.author.name)

client.run(TOKEN)
```

Figure 4: Slightly modified version of the basic bot code that appears on the discord.py documentation webpage (Rapptz, Danny, 2015/2023).

message sent to the bot is automatically accompanied by associated metadata, allowing the bot to incorporate the message author's username.

As part of a series of in-class exercises, students were asked to make incremental changes to this code to expand the bot functionality. Each exercise required the student to modify and then test the code via Discord server interaction. The exercises culminated in the creation of a bot that performed basic calculations (addition, subtraction, etc.) at the request of a user. The botnet project was assigned after the preparation materials and associated exercises were completed.

5. RELATED WORK

This section provides a literature-based survey of the methods (assignments, exercises, or projects) utilized to introduce or reinforce the topics of access control and botnets.

Courses requiring software development skills, such as software engineering or operating systems design, typically approach access control projects from an implementation angle – requiring students to either audit or write the code that implements a particular access control mechanism (Luburic et al., 2019; Nieh & Vaill, 2005; Petullo et al., 2016). A similar approach has been taken by database management and design courses (Yang, 2009). In this case, access control features (users, policies, roles, etc.) are implemented or audited using the database management system's tools.

Lab-based access control exercises are frequently found in system administration and cybersecurity courses (Du et al., 2010; Hay et al., 2008; Hu & Wang, 2008). These lab exercises require the student to implement, via operating system tools or features, various forms of access control on system resources such as networks or external storage. The use of these tools may be reinforced by team-based capture-the-flag competitions (Eagle & Clark, 2004) in which students attempt to either breach or defend computing resources (Chothia & Novakovic, 2015; Mirkovic & Peterson, 2014).

Two freely-available software-based tools have been developed to allow an individual to create, visualize and query RBAC (Wang et al., 2015) and MLS (Wang et al., 2014) access control policies. The student uses a simple point-and-click interface to create, interact with, and modify a graph-based visualization of the policy. The software can be used to verify policy constraints.

A less formal approach to access control training can be found in CyberCIEGE, a video game intended to support education and training in computer and network security (Cone et al., 2007). The game employs resource management and simulation to illustrate information assurance concepts for education and training (Thompson & Irvine, 2014). Access control is only one of the topics covered by the game (*CyberCIEGE Scenario Listing*, 2023).

Botnet projects and lab exercises run the gamut from passive to active. Lab exercises on the more passive end (Dias et al., 2017; Hay et al., 2008) focus on watching a "harmless" botnet (which doesn't contain exploit code) in action by utilizing packet sniffers and network visualization tools. Due to the distributed nature of botnets, the lab exercises are conducted on virtual machines and bots are prevented from spreading to the external network. In these lab exercises the student plays the role of the botmaster issuing commands to the zombies via an IRC server that is distributed as part of the virtual machine images. Another variation of this lab exercise replaces the "harmless" botnet code with "real" botnet code (Hannah & Gianvecchio, 2015).

The second type of botnet lab exercises and projects focuses on modifying the botnet codebase to introduce new features such as additional infection methods or new client information capture (DeCusatis et al., 2021; Vergos, 2011).

6. ADDITIONAL DISCORD-RELATED PROJECT IDEAS

We believe that Discord's popularity, availability, and content make it a rich source for student projects. Since it is fundamentally a communications platform, chatbots are the most obvious type of project that can be assigned (Cerezo et al., 2019; Raglianti et al., 2022).

The desktop Discord client caches, on disk, much information about a user's session (Iqbal et al., 2021). This is done to minimize the amount of bandwidth utilized during a Discord session. This cached information can form the basis of digital forensics-based projects.

Discord's popularity among the Non-Fungible Token (NFT) and cryptocurrency community (Sharma et al., 2022) may provide a good source of data to analyze what might be involved in

launching and maintaining a successful NFT project. This community is frequently the target of scammers and cybercriminals (TRM Labs, 2022)– this malicious activity generates Discord-based text and links that can also be extracted and analyzed by students.

7. STUDENT FEEDBACK AND DISCUSSION

We have been utilizing Discord-based projects since the Spring 2022 term. Overall, about half the students report having previously used Discord. This number is slightly higher for students taking the cloud computing course. Nevertheless, none of the students reported any difficulty completing the exercises/tutorial that introduced Discord and server creation.

The Data Privacy project was initially assigned as a group project to be completed during one in-class session. However, based on post-project student feedback, it was converted to a single student project and split over two class sessions. This gave the students more time to think about the project and they did not have to worry about delegating tasks to different members of the group. After these changes were implemented, most students strongly agreed that the project improved their understanding of how role-based permissions are an important part of an organization's privacy protection program.

Cloud Computing course students were given one week to complete the project. Each student had to complete the project on their own - this was not a team-based project. Students were encouraged to contact the instructor if they experienced any code errors that prevented them from making progress on the project. Only about a quarter of the students contacted the instructor with code-related error issues. Most of these errors had to do with the improper use of the Python hashlib module. This module provides an interface to several secure hash and message digest algorithms. Based on this experience, and student feedback, additional course time will be devoted to proper use of this module. In a post-project survey slightly under 75% of the students reported that they liked working on the project – using words such as "fun", "enjoyed" and "great" in their response. Slightly under 80% of the students agreed with the statement that the project helped them improve their understanding of the project-relevant computer security topics.

All project-related materials are available for download (https://github.com/marcwaldman/discord_rbac_botnet). The scenarios utilized in, and the components of, both projects can be tailored based on student background.

8. REFERENCES

Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II 16*, 373–383. https://doi.org/10.1007/978-3-030-49186-4_31

Bailey, M., Cooke, E., Jahanian, F., Xu, Y., & Karir, M. (2009). A survey of botnet technology and defenses. *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, 299–304. https://doi.org/10.1109/CATCH.2009.40

Browning, K. (2021, December 29). How Discord, Born From an Obscure Game, Became a Social Hub for Young People. *The New York Times*. https://www.nytimes.com/2021/12/29/business/discord-server-social-media.html

Casenove, M., & Miraglia, A. (2014). Botnet over Tor: The illusion of hiding. *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, 273–282. https://doi.org/10.1109/CYCON.2014.6916408

Cerezo, J., Kubelka, J., Robbes, R., & Bergel, A. (2019). Building an Expert Recommender Chatbot. *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*, 59–63. https://doi.org/10.1109/BotSE.2019.00022

Chothia, T., & Novakovic, C. (2015). An Offline Capture The {Flag-Style} Virtual Machine and an Assessment of Its Value for Cybersecurity Education. *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*. https://www.usenix.org/system/files/conference/3gse15/3gse15-chothia.pdf

Cone, B. D., Irvine, C. E., Thompson, M. F., & Nguyen, T. D. (2007). A video game for cyber security training and awareness. *Computers and Security*, *26*(1), 63–72. https://doi.org/10.1016/j.cose.2006.10.005

Cooke, E., Jahanian, F., & McPherson, D. (2005). The Zombie Roundup: Understanding,

Detecting, and Disrupting Botnets. *Usenix SRUTI*, *5*. https://www.usenix.org/legacy/event/sruti05/tech/full_papers/cooke/cooke_html/

CyberCIEGE Scenario Listing. (2023). https://nps.edu/web/c3o/scenarios

D'Agostino, S. (2023, April 26). *Discord for Leaking Military Files—And Exam Questions*. Inside Higher Ed. https://www.insidehighered.com/news/tech-innovation/digital-teaching-learning/2023/04/26/discord-leaking-military-files-and-exam

DeCusatis, C. M., Bavaro, J., Cannistraci, T., Griffin, B., Jenkins, J., & Ronan, M. (2021). Red-blue team exercises for cybersecurity training during a pandemic. *11th IEEE Annual Computing and Communication Workshop and Conference, CCWC 2021, Las Vegas, NV, USA, January 27-30, 2021*, 1055–1060. https://doi.org/10.1109/CCWC51732.2021.9376016

Dev, J. A. (2013). Usage of botnets for high speed md5 hash cracking. *Third International Conference on Innovative Computing Technology (INTECH 2013)*, 314–320. https://doi.org/10.1109/INTECH.2013.6653658

Dias, J. P., Pinto, J. P., & Cruz, J. M. (2017). A Hands-on Approach on Botnets for Behavior Exploration. In M. Ramachandran, V. M. Muñoz, V. Kantere, G. B. Wills, R. J. Walters, & V. Chang (Eds.), *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security, IoTBDS 2017, Porto, Portugal, April 24-26, 2017* (pp. 463–469). SciTePress. https://doi.org/10.5220/0006392404630469

Discord. (2023, March 8). *About Discord*. About Discord. https://discord.com/company

Discord API. (2023, April 23). *Discord Gateways API*. Discord Developer Portal. https://discord.com/developers/docs/topics/gateway

Discord Code. (2023, May 3). *Discord Vetted Code Libraries*. Discord Developer Portal. https://discord.com/developers/docs/topics/community-resources#libraries

Discord Permissions. (2023, April 24). *Discord Permissions Documentation*. Discord Developer Portal. https://discord.com/developers/docs/topics/permissions

Du, W., Jayaraman, K., & Gaubatz, N. B. (2010). Enhancing security education with hands-on laboratory exercises. *Proceedings of the 5th Annual Symposium on Information Assurance (ASIA'10)*, 56–61. https://www.academia.edu/download/74371140/ASIA10Proceedings.pdf#page=65

Eagle, C., & Clark, J. L. (2004). Capture-the-flag: Learning computer security under fire. *Proceedings from the Sixth Workshop on Education in Computer Security (WECS6). Monterey, CA*. https://apps.dtic.mil/sti/citations/tr/ADA435319

Feily, M., Shahrestani, A., & Ramadass, S. (2009). A survey of botnet and botnet detection. *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, 268–273. https://doi.org/10.1109/SECURWARE.2009.48

Ferraiolo, D., Cugini, J., & Kuhn, D. R. (1995). Role-Based Access Control (RBAC): Features and Motivations. *Proceedings of 11th Annual Computer Security Application Conference*, 241–248. https://csrc.nist.gov/pubs/conference/1995/12/15/rolebased-access-control-rbac-features-and-motivat/final

Fielding, R. T., & Taylor, R. N. (2002). Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology (TOIT)*, *2*(2), 115–150. https://doi.org/10.1145/514183.514185

Grimbsy, G. (2019). Using Discord To Foster A Learning Community. *Innovations in Teaching & Learning Conference Proceedings*, *11*, Hall-Hall. https://doi.org/10.13021/itlcp.2019.2520

Hannah, K., & Gianvecchio, S. (2015). Zeuslite: A tool for botnet analysis in the classroom. *Journal of Computing Sciences in Colleges*, *30*(3), 109–116. https://doi.org/10.1145/2714576.2714579

Hay, B., Dodge, R., & Nance, K. L. (2008). Using Virtualization to Create and Deploy Computer Security Lab Exercises. In S. Jajodia, P. Samarati, & S. Cimato (Eds.), *Proceedings of The IFIP TC-11 23rd International Information Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008, September 7-10, 2008, Milano, Italy* (Vol. 278, pp. 621–635). Springer. https://doi.org/10.1007/978-0-387-09699-5_40

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2015). Botnet in DDoS attacks: Trends and challenges. *IEEE Communications Surveys & Tutorials*, *17*(4), 2242–2270. https://doi.org/10.1109/COMST.2015.2457491

Hu, D., & Wang, Y. (2008). Teaching computer security using xen in a virtual environment. *2008 International Conference on Information Security and Assurance (Isa 2008)*, 389–392. https://doi.org/10.1109/ISA.2008.18

Iqbal, F., Motyliński, M., & MacDermott, Á. (2021). Discord Server Forensics: Analysis and Extraction of Digital Evidence. *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1–8. https://doi.org/10.1109/NTMS49979.2021.9432654

Leidig, P., & Salmela, H. (2020). *IS2020 A Competency Model for Undergraduate Programs in Information Systems: The Joint ACM/AIS IS2020 Task Force*. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/is2020.pdf

Luburic, N., Sladic, G., Slivka, J., & Milosavljevic, B. (2019). A Framework for Teaching Security Design Analysis Using Case Studies and the Hybrid Flipped Classroom. *ACM Trans. Comput. Educ.*, *19*(3), 21:1-21:19. https://doi.org/10.1145/3289238

Melnikov, A., & Fette, I. (2011). *The WebSocket Protocol* (Request for Comments RFC 6455). Internet Engineering Task Force. https://doi.org/10.17487/RFC6455

Mirkovic, J., & Peterson, P. (2014). Class Capture-the-Flag Exercises. In Z. N. J. Peterson (Ed.), *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, 3GSE '14, San Diego, CA, USA, August 18, 2014*. USENIX Association. https://www.usenix.org/conference/3gse14/summit-program/presentation/mirkovic

Nieh, J., & Vaill, C. (2005). Experiences teaching operating systems using virtual platforms and linux. In W. P. Dann, T. L. Naps, P. T. Tymann, & D. Baldwin (Eds.), *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2005, St. Louis, Missouri, USA, February 23-27, 2005* (pp. 520–524). ACM. https://doi.org/10.1145/1047344.1047508

Pantic, N., & Husain, M. I. (2015). Covert botnet command and control using twitter. *Proceedings of the 31st Annual Computer Security Applications Conference*, 171–180. https://doi.org/10.1145/2818000.2818047

Petullo, W. M., Moses, K., Klimkowski, B., Hand, R., & Olson, K. (2016). The Use of Cyber-Defense Exercises in Undergraduate Computing Education. In M. A. Gondree & Z. N. J. Peterson (Eds.), *2016 USENIX Workshop on Advances in Security Education (ASE 16), Austin, TX, USA, August 9, 2016*. USENIX Association. https://www.usenix.org/conference/ase16/workshop-program/presentation/petullo

Raglianti, M., Nagy, C., Minelli, R., & Lanza, M. (2022, May 16). Using Discord Conversations as Program Comprehension Aid. *30th International Conference on Program Comprehension (ICPC '22)*. ACM International Conference on Program Comprehension (ICPC), Virtual Event, USA. https://doi.org/10.1145/3524610.3528388

Rapptz, Danny. (2023). *Discord.py* [Python]. https://github.com/Rapptz/discord.py (Original work published 2015)

Sharma, T., Zhou, Z., Huang, Y., & Wang, Y. (2022). *"It's A Blessing and A Curse": Unpacking Creators' Practices with Non-Fungible Tokens (NFTs) and Their Communities* (arXiv:2201.13233). arXiv. https://doi.org/10.48550/arXiv.2201.13233

Sood, A. K., & Enbody, R. J. (2013). Crimeware-as-a-service—A survey of commoditized crimeware in the underground market. *International Journal of Critical Infrastructure Protection*, *6*(1), 28–38. https://doi.org/10.1016/j.ijcip.2013.01.002

Sukhanova, K. (2023, July 23). *The Latest Discord Statistics & Trends for 2023*. https://techreport.com/statistics/discord-statistics/

Thompson, M. F., & Irvine, C. E. (2014). CyberCIEGE Scenario Design and Implementation. In Z. N. J. Peterson (Ed.), *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, 3GSE '14, San Diego, CA, USA, August 18, 2014*. USENIX Association. https://www.usenix.org/conference/3gse14/summit-program/presentation/thompson

TRM Labs. (2022, July 25). *Analysis of Recent NFT Discord Hacks Shows Some Attacks Are Connected | TRM Insights*. https://www.trmlabs.com/post/trms-analysis-of-recent-surge-in-discord-hacks-shows-some-attacks-are-connected

Vergos, D. (2011). *Botnet lab creation with open source tools and usefulness of such a tool for researchers* [Rochester Institute of Technology]. https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1144&context=theses

Vormayr, G., Zseby, T., & Fabini, J. (2017). Botnet communication patterns. *IEEE Communications Surveys & Tutorials*, *19*(4), 2768–2796. https://doi.org/10.1109/COMST.2017.2749442

Wang, M., Carr, S., Mayo, J., Shene, C.-K., & Wang, C. (2014). MLSvisual: A visualization tool for teaching access control using multi-level security. In Å. Cajander, M. Daniels, T. Clear, & A. Pears (Eds.), *Innovation and Technology in Computer Science Education Conference 2014, ITiCSE '14, Uppsala, Sweden, June 23-25, 2014* (pp. 93–98). ACM. https://doi.org/10.1145/2591708.2591730

Wang, M., Mayo, J., Shene, C.-K., Lake, T., Carr, S., & Wang, C. (2015). RBACvisual: A Visualization Tool for Teaching Access Control using Role-based Access Control. In V. Dagiene, C. Schulte, & T. Jevsikova (Eds.), *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCS 2015, Vilnius, Lithuania, July 4-8, 2015* (pp. 141–146). ACM. https://doi.org/10.1145/2729094.2742627

Wiles, A. M., & Simmons, S. L. (2022). Establishment of an Engaged and Active Learning Community in the Biology Classroom and Lab with Discord. *Journal of Microbiology & Biology Education*, *23*(1), e00334-21. https://doi.org/10.1128/jmbe.00334-21

Yang, L. (2009). Teaching database security and auditing. In S. Fitzgerald, M. Guzdial, G. Lewandowski, & S. A. Wolfman (Eds.), *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2009, Chattanooga, TN, USA, March 4-7, 2009* (pp. 241–245). ACM. https://doi.org/10.1145/1508865.1508954