Introduction:

In this semester, we have worked on a project in which we constructed a cybersecurity program and system that identifies and categorizes possible threats, vulnerabilities, and attack courses that could threaten an Organization/Company that would pose a threat to our hypothetical organization. Basically, the purpose of this project has been to challenge us to come up with an organizational system that is designed to combat any kind of data breach or security breach that may come into contact with our organization in the future, as well as all the components associated with it from everything we have learned from this semester, including Threat identification, Risk mitigation strategies, Risk management, incident response and monitoring, Automation, Machine Learning, and Disaster Recovery.

Objectives:

- Apply systems engineering concepts to design a robust cybersecurity system.
- Utilize UML to model the system architecture and processes.
- Implement Python automation scripts for various cybersecurity tasks.
- Integrate data analytics and machine learning for threat detection and response.
- Develop risk management and disaster recovery plans.

Systems engineering (SE) is a structured way to design and manage complex systems. Using SE in cybersecurity helps create strong and flexible security systems to protect against changing threats. The text outlines how SE can improve cybersecurity. Key ideas include defining clear cybersecurity goals and understanding stakeholder needs, designing a comprehensive security structure, and integrating layers of security. Risk management assesses threats and vulnerabilities, while ensuring security solutions fit with existing systems. Regular testing detects weaknesses, and adaptability is crucial for responding to new threats. SE also focuses on incorporating security throughout all development phases and ensuring secure system disposal.

UML (Unified Modeling Language) is a standard way to visually show systems, including their structure and relationships. It is vital in cybersecurity for designing and analyzing security features of systems. UML helps visualize system

architecture to find vulnerabilities by displaying data flow and unauthorized access points. Key diagrams include Component, Deployment, and Class Diagrams. It also models security use cases, illustrating user interactions with the system through Use Case and Activity Diagrams. UML helps identify threats by showing potential attack paths with Sequence and State Diagrams. Moreover, it documents security policies and controls, enhancing communication with developers. UML diagrams effectively convey security designs to non-technical stakeholders and support threat modeling and risk assessment. Cybersecurity experts use UML to analyze security measures and protect sensitive data. Overall, UML is crucial for identifying and addressing vulnerabilities in cybersecurity.

Implementing Python automation scripts for various cybersecurity tasks can significantly enhance the efficiency, accuracy, and effectiveness of security operations. Python, with its extensive libraries and ease of use, is particularly well-suited for automating repetitive, time-consuming tasks, and for implementing complex security processes. Its importance in cybersecurity is from the fallowing reasons such as Increased Efficiency: Automating repetitive and mundane tasks frees up security professionals to focus on more complex issues, reducing human error and operational overhead. Consistency and Repeatability: Automation ensures that security tasks (like vulnerability scanning, log analysis, or testing) are performed the same way each time, improving accuracy and completeness. Faster Detection and Response: With real-time monitoring and automated incident detection, security teams can respond more quickly to potential threats and breaches. Scalability: Automation allows organizations to scale their cybersecurity efforts to handle large networks, systems, and data volumes, which would be difficult or impossible to manage manually. In summary, Python automation is a powerful tool in cybersecurity because it improves efficiency, reduces manual.

Integrating data analytics and machine learning (ML) into cybersecurity systems enhances an organization's ability to identify, respond to, and reduce cyber threats. Traditional threat detection is often not enough as threats change, but data analytics and ML offer advanced methods to analyze large data sets, uncover hidden patterns, and automate responses, leading to faster and more precise threat management. Anomaly detection helps identify unusual activities that may indicate cyberattacks, while behavioral analytics recognizes abnormal behavior by understanding normal patterns for users and devices. ML automates threat classification by learning from previous attacks. Real-time responses can be triggered once a threat is detected, and predictive modeling allows systems to predict potential threats. After cyber incidents, data analytics and ML assist investigations, aiding in fraud detection and improving security strategies. The

significance of using these technologies includes proactive threat detection, automation for efficiency, adaptive defenses, and enriched threat intelligence. Overall, they transform cybersecurity practices, enabling organizations to handle threats more effectively.

Risk Management and Disaster Recovery (DR) are key components of a cybersecurity strategy, helping organizations protect their data and maintain operations against cyber threats and disasters. Risk management focuses on identifying and reducing risks, while disaster recovery restores essential systems post-disruption. Together, they improve an organization's resilience against cyberattacks, natural disasters, and other incidents. A Risk Management Plan includes spotting and dealing with risks to the organization's assets. Steps to develop an Risk Management Plan involve identifying threats, assessing risks, evaluating risks using a risk matrix, and implementing mitigation strategies. Continuous monitoring and regular reviews ensure the plan stays effective. A Disaster Recovery Plan (DRP) outlines steps to restore IT systems after a disaster, focusing on quick recovery and minimizing downtime. Steps include conducting a Business Impact Analysis (BIA), defining recovery strategies, detailing recovery procedures, and regular testing. These plans reduce financial losses, ensure compliance, and build trust with stakeholders, making them essential for organization stability and response to cyber threats.

Methodology:

Let's go through each of the tools and techniques you mentioned and explain their relevance and significance in software development and cybersecurity. UML is a standardized modeling language used to specify, visualize, and document the structure and behavior of a system. It is widely used in object-oriented software engineering. UML diagrams model a system's security architecture, identify components, visualize threats, and help apply mitigations effectively.

Python is a versatile, high-level programming language known for its simplicity and readability. It is widely used for automation, web development, data analysis, and cybersecurity tasks. Python automates repetitive cybersecurity tasks like scanning networks, analyzing logs, and penetration tests. It aids in malware analysis to identify vulnerabilities. Python also analyzes large datasets for threats, using libraries like scapy and requests.

GitHub is a cloud-based platform that hosts Git repositories, enabling collaborative software development through version control and code sharing. GitHub helps teams manage versions of security scripts and tools, maintaining code integrity. It enhances collaboration in security research, allowing multiple contributors. GitHub also supports security audits and checks through integrations, and it hosts many open-source security tools for easy access.

CI/CD is a set of practices that involve automatically testing and deploying code changes. Continuous Integration (CI) involves integrating code into a shared repository frequently, while Continuous Delivery (CD) automates the delivery process to production environments. Automated security testing includes static code analysis, vulnerability scanning, and penetration testing to find security issues early. CI/CD enables quick deployment of security patches. Security configurations can be version-controlled and deployed automatically. Automating deployment and testing reduces human error and security risks.

Docker is a platform that enables developers to package applications and their dependencies into containers, ensuring consistent environments across development, testing, and production. Docker creates isolated environments for applications, reducing vulnerabilities by limiting attack surfaces. Securing container communication is crucial in microservices to prevent attacks. Hardening Docker containers is essential to avoid security risks. Docker images should be updated and scanned regularly for vulnerabilities.

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is widely used for data analysis, machine learning, and scientific computing. Jupyter Notebooks are used for security data analysis, like examining network traffic and log files. They help build machine learning models for detecting anomalies and classifying malicious activity. Jupyter allows team collaboration on security projects, letting members share and modify notebooks. They can create visualizations to understand and communicate security trends, like attack patterns or vulnerabilities.

A Use-Case Diagram is a type of UML diagram that visualizes the interactions between users (or "actors") and a system, highlighting the system's functionality from an end-user perspective. Use-case diagrams help identify security needs by showing interactions that may affect security, like login and data access. They also reveal where attackers could exploit the system and help establish role-based access control (RBAC) by clarifying which users can perform specific actions.

Log analysis, automation and other tasks that are related to log analysis, automation, and other similar activities can be handled in several ways and with many strategies that are commonly employed. Throughout the process, each task was broken down into smaller pieces and we worked through each one of the problems one by one. In the event we encountered any difficulties, we'd take a break, and then come back and resolve them, and during that time, we'd research the syntax of the new liberates in order to continue working on them.

Results:

For The First Script:

There were suspicious logs identified, including multiple authentication failures and unknown users. Then a summary was successfully generated, and it documented the total number of suspicious logs and their details.

For The Second Script:

It was the system performance results. system monitoring, process management, and cross platform compatibility. Another purpose is checking how well applications are running, finding, and fixing performance issues, and keeping track of the system's overall health. From these scripts we got that the systems **CPU Usage is** 4.8%, **Memory Usage is** 58.4%

For The Third Script:

It was nmap which shows the open ports and the ones that are available and active on a system. It also shows details on the system, like weak points or unauthorized services. The results from this are there was a scanned local host (127.0.0.1) which detected open ports.

The Fourth Script:

Packet Monitoring which Tracks and analyzes data moving through a network shows where the data comes from and where it is going. It also shows which devices are sending and receiving data, information about the type of data that is being sent and how the devices are communicating. It captured 10 TCP packets from the code 192.168.1.173 and the destination 13.107.138.10

Challenges and Solutions:

- Generally speaking, there were not a great deal of challenges that had to be overcome. Nevertheless, figuring out how to install the libraries proved to be the biggest challenge.

Recommendations:

- The recommendation that we would make is to beef up the scripts so that they are more secure. Adding more data and information will help us to make our research more effective.

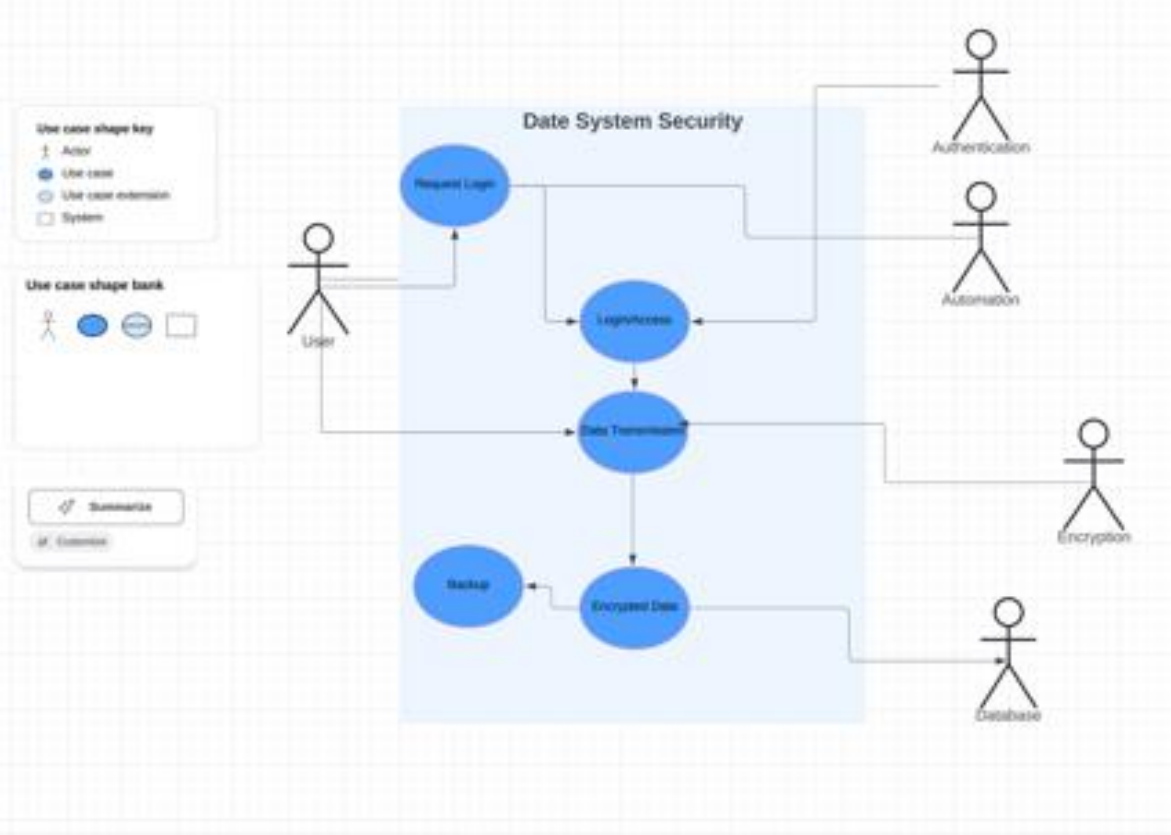
Conclusion:

Throughout this project, we have learned a lot about the systems we have done and a lot about cyber security and all the components that come within it, throughout this project we have accomplished a lot within automation, designing, data analysis, components and parts, UML modeling, Python, and risk management. During the course of our cybersecurity situation, we have learned a lot of information as to how most of the things that can be done to create a system that protects our information and prevents intruders from breaching our system have been done in the past couple of years. Essentially, the project aimed to challenge us to think about the various aspects of cybersecurity, such as system architecture and implementation, as well as monitoring and incident response in order to evaluate their effectiveness. It can be said that the accomplishments we were able to achieve were a result of the fact that we managed to build a system that covered all of the components of the criteria, but not only that, but we were also able to learn from all of the subjects and points we covered throughout the semester as well. This project was also an excellent opportunity for us to learn about working as a team, hard work, communication, and collaboration as a result of our collaboration on this project. We can defiantly say that we learned a lot both from the project and outside the project itself.

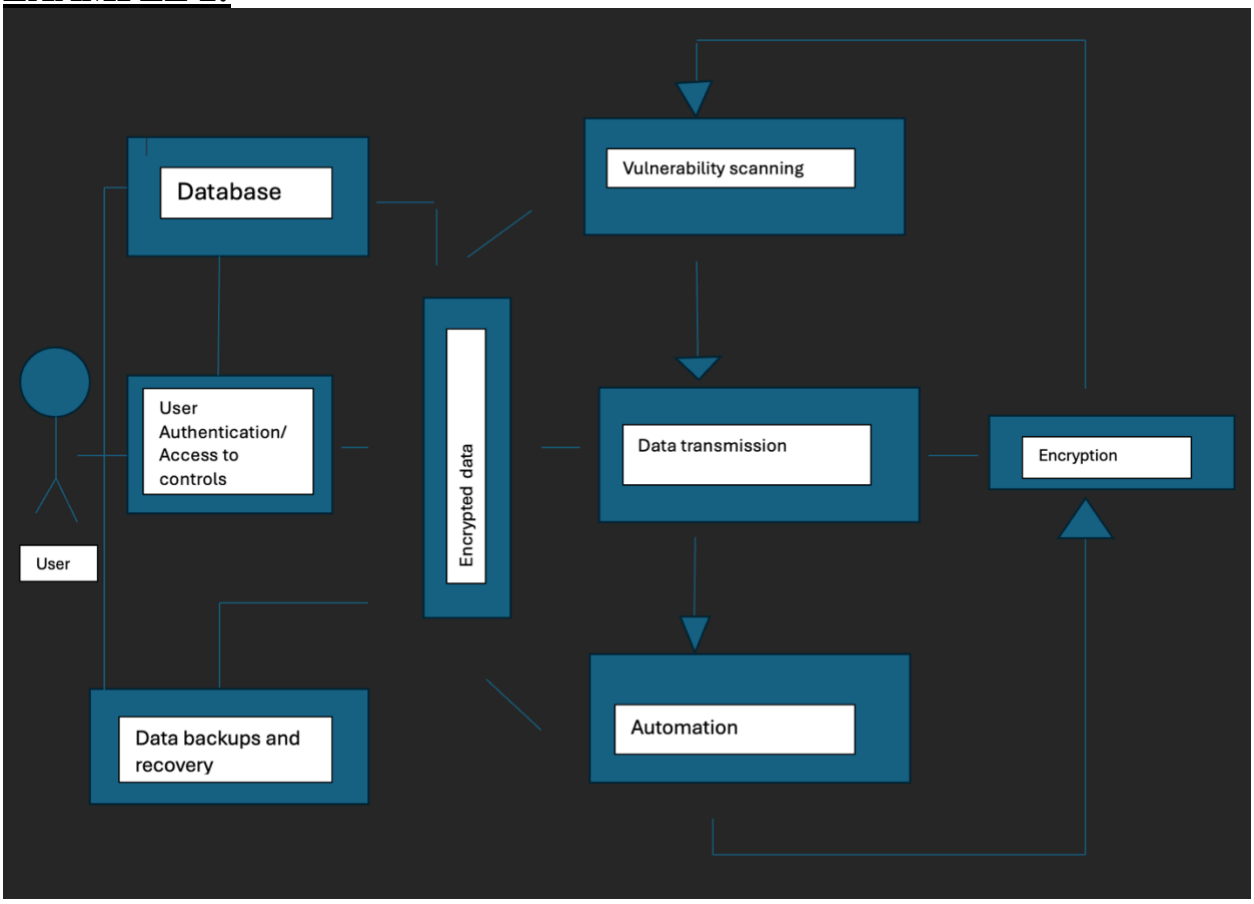
Appendices:

Here is some example sample scripts and UML diagrams from the project:

EXAMPLE 1:


Use case shape key
Actor
Use case
Use case extension
System
Use case shape bank
Summarize
Customize
Date System Security
Request Login
Login/Access
Data Transmission
Backup
Encrypted Data
User
Authentication
Automation
Encryption
Database

EXAMPLE 2:



EXAMPLE 3:

```
import psutil

# Get CPU usage
cpu_usage = psutil.cpu_percent(interval=1)
print(f"CPU Usage: {cpu_usage}%")

# Get Memory usage
memory_info = psutil.virtual_memory()
print(f"Memory Usage: {memory_info.percent}%")

with open('performance_log.txt', 'a') as f:
    f.write(f"CPU: {cpu_usage}%, Memory: {memory_info.percent}%\n")

if cpu_usage > 90:
    print("ALERT: High CPU usage detected!")

import smtplib
from email.message import EmailMessage

subject= "something"
```

```
body = "idk"
def send_alert(subject, body):
    msg = EmailMessage()
    msg.set_content(body)
    msg['Subject'] = subject
    msg['From'] = 'dumby805@gmail.com'
    msg['To'] = 'eshaanvaranasi@gmail.com'

    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
        smtp.login('dumby805@gmail.com', 'ltym djkt xrnc yknw')
        smtp.send_message(msg)
if cpu_usage > .2:
    send_alert('High CPU Usage Alert', f'CPU usage is {cpu_usage}%')
```

EXAMPLE 4:

```
import subprocess

def run_nmap(target):
    result = subprocess.run(['nmap', '-sV', target], capture_output=True,
text=True)
    print(result.stdout)

run_nmap('127.0.0.1')  # Scan localhost
```

The results on running the code:

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-10-28 19:27 Eastern Daylight
Time
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00037s latency).
Not shown: 993 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
445/tcp  open  microsoft-ds?
5357/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9000/tcp open  zmtp          ZeroMQ ZMTP 2.0
9001/tcp open  zmtp          ZeroMQ ZMTP 2.0
9002/tcp open  zmtp          ZeroMQ ZMTP 2.0
9003/tcp open  zmtp          ZeroMQ ZMTP 2.0
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.31 seconds
```