Group Project Final Report

The objectives of our project were to create a system which would apply basic cybersecurity principles for the financial firm, as well as employee training which would familiarize them with security processes and system functions. This system aims to enhance the organization's overall security posture by addressing critical vulnerabilities and equipping employees with the necessary training to recognize and respond to potential threats effectively. This project holds significance for cybersecurity in protecting a financial firm, which are prime targets for cyberattacks due to the sensitive nature of their data.

The financial firm was deficient in several areas in regards to security. To solve this problem we implemented a system which reads and parses through log files and generates a report on suspicious logs. The system monitors metrics such as system usage and login attempts and generates alerts if it detects activities such as multiple failed login attempts or high system usage, ensuring that the integrity of the system is not compromised. Basic encryption practices were also implemented in order to protect the confidentiality of the messages, with employees given basic awareness training to prevent attacks such as phishing.

For this project, we had to look around Google to identify an efficient tool for UML diagrams. After some research, we settled on *LucidChart* where it provides the user with templates and designs to create your own UML diagram and with this tool, it was much easier to approach the creation of this diagram. We used python to display the messages and fulfil the requirements needed for this assignment. The python code we used helped us form a system that we could use to apply basic cybersecurity principles for the financial firm. We used comments like # Collect System Metrics, # Get CPU usage, # Get Memory usage, and # Log the Performance Data to help organize our code and display the code with the correct group of code that we wanted the information to be associated with. We used UML to create a use case diagram. In our use case diagram we described a banking system, our primary actor is the customer and our secondary actors are Systems Administrator/Auditor, Cloud Service Provider, employee, and service authentication. The use cases we used are Log-in, Access control, Cloud Awareness, and Employee Awareness and Training. We also made a Banking System UML Activity Diagram. The positions we used are, Customer, Auditor, Cloud Service Provider, Employee, and Authentication Service. The activity states we used are Log In, Access Control Request, Monitor Access and Incident Detection, Configure Access Permissions, Incident Response, Monitor Cloud Security, Compliance Monitoring, Receive Security Training, Access Cloud Security, Validate Credential, Access Decision, Access Denied, and Access Granted. We used GitHub to store the code so everyone can use it to fulfil different aspects of our project.

In terms of the methodology of our allotted tasks, we tried attacking all components one step at a time. When completing the automation segment of our project, we first laid out a foundation for the code utilizing the sample code that was given. Then, we used the detailed instructions to form comments that would guide us in what to do with the provided code and what tasks the specific code should complete. Next, we relied on our own knowledge and coding experience to manipulate the sample code into a program that would produce logs and outputs in the terminal. Finally, we ensured our code had minimal errors if any. Similar to how we used the sample code for automation, when completing reports we relied on the provided outline to direct our writing. We attempted to complete and fully answer all bullet points to the best of our ability. Lastly, when completing diagrams, we discussed what the possibilities were for the actors, relationships, and much more before putting our discussion findings into diagrams.

Our UML diagrams explain the process of how customers are able to access their bank information. They show the possible results of every action, so essentially they show that the user has a choice. Some actions will lead to the user getting access to their account and some actions will lead to the user getting denied from their account. So the user has to follow a certain system correctly to access their account. Similar to any account you have that has two-factor authentication. Some accounts will make you put in a code that was texted to your phone number, or some accounts will make you answer a security question you told them the answer to, like what is the name of your dog? There are also many other two-factor authentication methods. The purpose of these two-factor authentication steps are to make sure the person trying to access the account is actually the owner or an authorized user of the account. So in our UML diagrams both the case diagram and activity diagram, we showed the process of how the user is able to access their account. In terms of automation, our code produced output files. One of the output files is a summary report text file that tracks suspicious logs. The other file it produces is one containing performance data with CPU and memory percentages. Furthermore, the code sends out alerts in the form of emails.

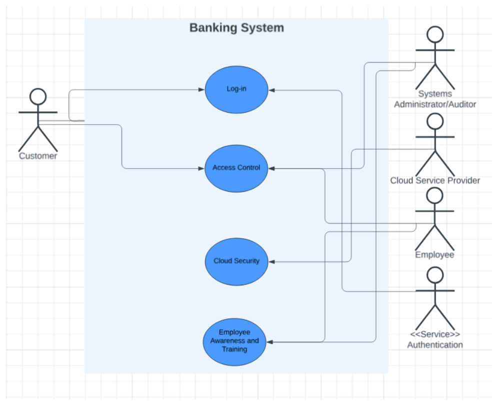
Throughout the process of completing this project, we as a group faced many challenges and obstacles where we had to rely on one another to overcome them. One of the first complications we faced was a lack of coding experience in knowledge for the level of Python required to complete the automation segments of this group assignment. The lack of experience of coding with Python at a higher level than most of us used to forced us to heavily rely on group members who had more experience and were more proficient in Python. Another problem we encountered was with the use of GitHub. A majority of the group had never seen or heard of GitHub thus forcing us again to heavily rely on the group members that had the experience and knowledge required to utilize the collaborative platform. The obstacles we faced with GitHub and Python forced the group to improvise with roles and allocating responsibilities to ensure not one person was pulling too much of the weight and that not one person was shirking their duties forcing others to do more than what was asked. We used the strengths of each of our group members to complete this assignment. If one group member lacked experience in Python but was experienced in GitHub, GitHub is where that group member was able to pull their own weight. Although we relied heavily on the strengths of each member, we found other solutions to our obstacles as well.

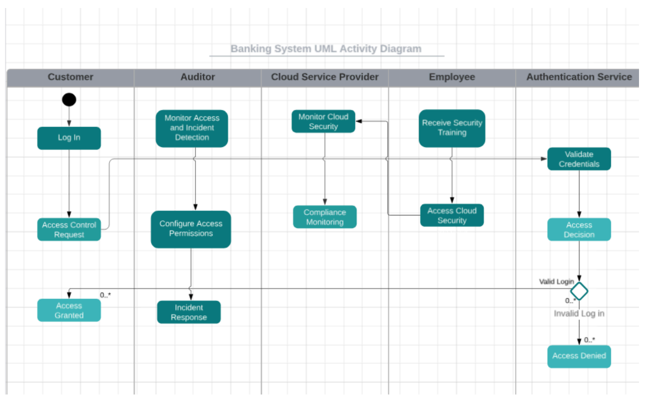
Outside of relying on groupmates that knew more about specific fields than others, our group utilized other solutions to overcome the conflicts of lack of knowledge and experience. One of the solutions to our lack of knowledge and prior experience with creating UML diagrams was simply to look back at the notes we took in class and the slides provided on Blackboard. This helped guide us through determining possible actors, relationships, and relationships of the system of our use-case diagram. We also utilized this solution when designing our activity diagram for determining the flow of activities, swimlanes, input values, output values, and much more. Resorting to given materials in resources, we also utilized sample code provided in detailed instructions in order to assist the automation behind our project. Utilizing the sample code was crucial as it gave us a foundation to then build upon in finishing our code. Although coding and designing diagrams was difficult, through looking at provided examples in class and online, we were able to successfully complete our semester long project.

To enhance the system’s effectiveness and scalability, we propose several improvements and additional features. Implementing multi-factor authentication would boost user access security, reducing the chances of credential theft. A real-time dashboard would allow administrators to effectively track system activity, security metrics, and alerts. We suggest adding an automated incident response feature that sends alerts and performs predefined actions like blocking suspicious IP addresses or isolating compromised accounts. To aid in employee training, incorporating interactive simulations of common cyber attacks, such as phishing and ransomware, would offer practical learning experiences. These improvements would create a stronger, more user-friendly system that can adapt to the constantly changing world of cybersecurity.

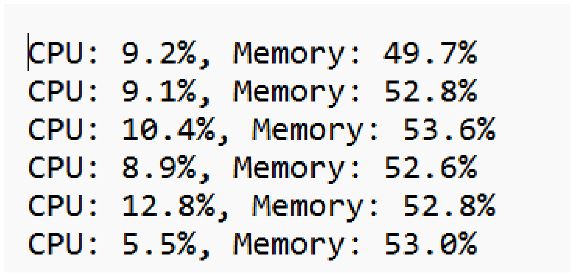
In conclusion, we view our project as a successful learning experience in team collaboration and utilizing outside websites to complete our work. The biggest takeaway was learning how to manage our work equally and effectively as most of the group are freshmen. Group assignments are no easy feat in college, as all participation is necessary for a good outcome of the shared assignment, once all job roles are assigned the team can progress fluidly.

UML Diagrams

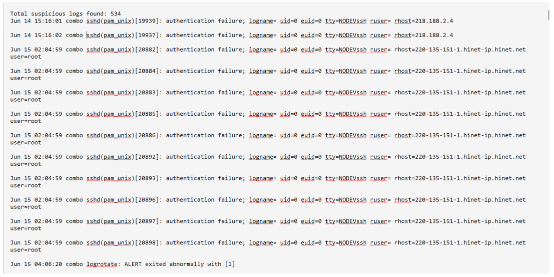

Banking System
Log-in
Access Control
Cloud Security
Employee Awareness and Training
Customer
Systems Administrator/Auditor
Cloud Service Provider
Employee
<<Service>> Authentication



Log File Analysis and System Performance Monitoring

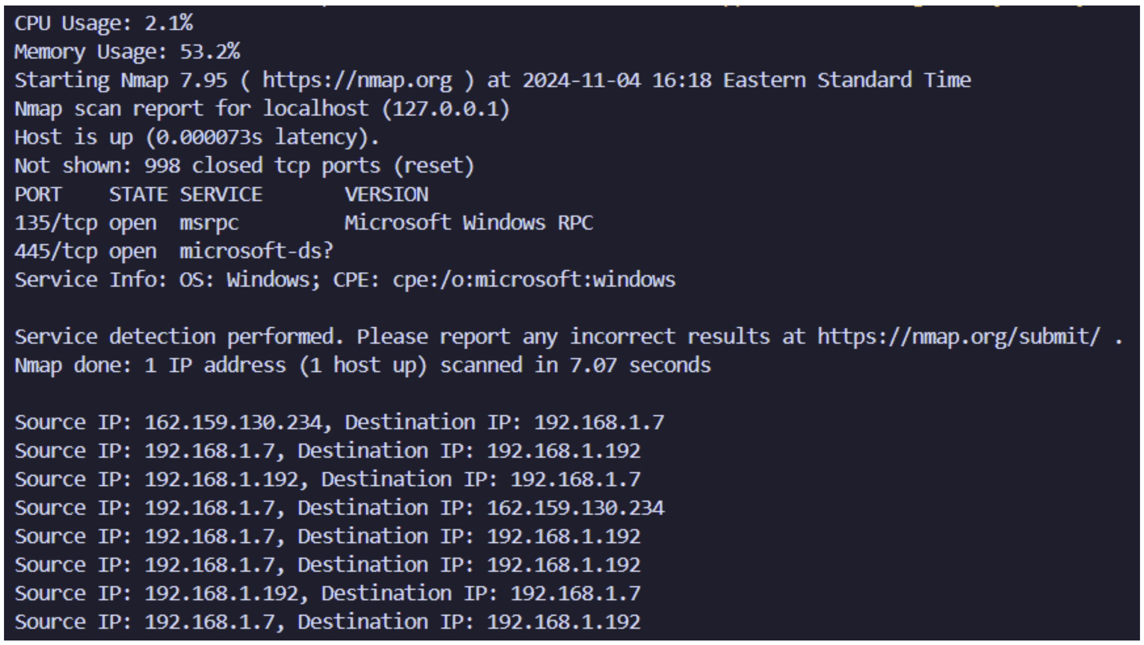


```
CPU: 9.2%, Memory: 49.7%
CPU: 9.1%, Memory: 52.8%
CPU: 10.4%, Memory: 53.6%
CPU: 8.9%, Memory: 52.6%
CPU: 12.8%, Memory: 52.8%
CPU: 5.5%, Memory: 53.0%
```



```
Total suspicious logs found: 534
Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4

Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4

Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20884]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20883]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20885]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20886]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20892]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20893]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20896]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20897]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 02:04:59 combo sshd(pam_unix)[20898]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net
user=root

Jun 15 04:06:20 combo logrotate: ALERT exited abnormally with [1]
```

Terminal Output



```
CPU Usage: 2.1%
Memory Usage: 53.2%
Starting Nmap 7.95 ( https://nmap.org ) at 2024-11-04 16:18 Eastern Standard Time
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000073s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
445/tcp open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.07 seconds

Source IP: 162.159.130.234, Destination IP: 192.168.1.7
Source IP: 192.168.1.7, Destination IP: 192.168.1.192
Source IP: 192.168.1.192, Destination IP: 192.168.1.7
Source IP: 192.168.1.7, Destination IP: 162.159.130.234
Source IP: 192.168.1.7, Destination IP: 192.168.1.192
Source IP: 192.168.1.7, Destination IP: 192.168.1.192
Source IP: 192.168.1.192, Destination IP: 192.168.1.7
Source IP: 192.168.1.7, Destination IP: 192.168.1.192
```