FINAL REPORT

Cybersecurity is a constantly evolving field with new threats appearing every day. This project aimed to develop an automated system designed to enhance the cybersecurity posture of an organization by identifying, analyzing, and mitigating security risks in real time. Specifically, the project focused on creating a toolset that automates data analysis, integrates security insights, and generates actionable reports, all while constantly adapting to various security infrastructures.

The objectives of this project are to address the need for organizations to automate the identification and mitigation of security risks in a rapidly changing digital environment. Organizations are constantly under threat from various cyberattacks, including phishing, malware, and advanced persistent threats (APTs). These threats are often difficult to detect manually due to their complexity and the sheer volume of data that security teams must sift through. More specifically, in this scenario, a Russian spy with a position of power was able to use it and breach private information stored within the targeted company's database. The data that was stolen was bank statements, customer account information. Using privileged access, the spy was able to access the cloud systems and was able to take financial information like bank statements, personal customer account information, as well as account numbers. More importantly, the spy was able to obtain private contracts like national defense and safety.

The tools needed to tackle and solve these problems included UML, Python, Lucid Chart, GitHub, etc. Firstly we created 3 different UML diagrams using the program LucidChart. The three diagrams included Sequence diagram, Activity diagram, and Use-case diagram.

Secondly, For this automation assignment, we developed several Python scripts aimed at improving cybersecurity by automating key tasks such as log file analysis, system performance monitoring, alert generation, and routine security checks. We started by creating a log file analysis script that parses logs to detect suspicious patterns, such as failed login attempts or unauthorized access. Using Python's built-in file handling and regular expressions, the script filters logs based on criteria like date, IP address, or error codes and generates a summary report of critical incidents. Next, the system performance monitoring script was developed using the psutil library to track CPU, memory, and disk usage. It logs performance data at regular intervals and triggers alerts via email when resource usage exceeds defined thresholds (e.g., CPU usage over 90%). The alert generation system monitors both log files and system performance for critical events, such as multiple failed logins or performance spikes. When an event is detected, the script sends notifications to administrators via email or SMS using smtplib and the Twilio API. Finally, we automated routine security checks, including vulnerability scanning with nmap and network traffic monitoring using scapy and pyshark. These scripts were set to run regularly to identify potential vulnerabilities and abnormal traffic patterns. Each script was well-documented, tested with sample data, and integrated to provide an automated solution for real-time security monitoring and alerting.

Lastly, for analysis, we took the “logging” test which was put into a CSV file stored in the Github repository. Then, the analysis script will take the CSV file, read it, and flags if the CPU usage is above a certain threshold. The flags will be brought to a separate column and will be labeled as “warnings”s. The algorithm used will perform “anomaly analysis” which looks for large amounts of high CPU usage, and sends an email to the administrator.

OUTCOMES:

CSV file outcome of logging.

	Timestamp	CPU Usage	Hostname
2	2024-12-06 13:44:46	9.7	F3-C3-Switch
3	2024-12-06 13:44:49	91.3	F3-C3-Switch
4	2024-12-06 13:44:51	48.1	F3-C3-Switch
5	2024-12-06 13:44:53	14.7	F3-C3-Switch
6	2024-12-06 13:44:55	93.8	F3-C3-Switch
7	2024-12-06 13:44:57	28.4	F3-C3-Switch
8	2024-12-06 13:44:59	6.6	F3-C3-Switch
9	2024-12-06 13:45:01	59.9	F3-C3-Switch
10	2024-12-06 13:45:03	60.9	F3-C3-Switch
11	2024-12-06 13:45:05	36.1	F3-C3-Switch
12	2024-12-06 13:45:08	99.7	F3-C3-Switch
13	2024-12-06 13:45:10	6.9	F3-C3-Switch
14	2024-12-06 13:45:12	29.8	F3-C3-Switch
15	2024-12-06 13:45:14	46.2	F3-C3-Switch
16	2024-12-06 13:45:16	81.0	F3-C3-Switch
17	2024-12-06 13:45:18	47.9	F3-C3-Switch
18	2024-12-06 13:45:20	80.9	F3-C3-Switch

ALL UML CHARTS

https://lucid.app/lucidchart/dc948b56-5a1f-41d0-8b6d-8356ed43c2b7/edit?viewport_loc=-1079%2C-56%2C4840%2C2451%2C0_0&invitationId=inv_1b53f5b7-89a3-4694-82c0-67aad7341c6e

Most obstacles encountered throughout this project were solely communicative and time management. Fortunately, the group worked well and there were minimal obstacles encountered throughout this process

Some improvements we could add would be to use an API in the logging system to scale up if new devices were added to the network.