Cybriant Project Plan

CS 4850 - Section 04 - Fall 2024

December 2, 2024

Team Members:

Name	Role	Phone #/Email
Jose	Team Lead	404.740.2039 Jmendo28@students.kennesaw.edu
Daniel	Lead Developer	404.936.1082 Dgutier8@studens.kennesaw.edu
David	Documenter	770.557.6673 nagyenfr@students.kennesaw.edu
Diwakar	Organizer	404.333.5322 Drai6@students.kennesaw.edu
Nicholas	Database Designer	305.790.4294 dlauren1@students.kennesaw.edu

1.0 Project Overview / Abstract (Research)

Companies need to be able to rate liability on a customer based off their security practices and our program is making an attempt at measuring that. We are going to take most of information we can get about them based off what they give us and what we can acquire. With that we plan on creating a rating based off data to determine what risk level insuring this company or how we could improve this company security will be. We are going to have to do this in a way in which we utilize Google Cloud resources. We will take certain data that we can access from companies and insert into a database. Then we have to update this database onto Google Cloud services. The hard part, which is programming a software in which it can read all this data and spit out a numerical rating is the last but most important step.

There are 718 lines of code in our main python file and 23 in our docker file, we also have a README documentation teaching how to use the project as a whole and a requirements documentation.

2.0 Project website

Deliverables - Specific To Your Project

Some of our deliverables are given to us by the project.

- System for gathering and analyzing security data
- API for data ingestion and access
- Documentation and user guide + Technical Detail.

However some more deliverables that we can have are:

- Final Documentation- This Final Report Package
- Final Github code link- Unavailable due to NDAs
- Final Presentation- A video presenting our work to our industry sponsors describing how we came up with the solutions and what we did to tackle their prompts.

Milestone Events (Prototypes, Draft Reports, Code Reviews, etc)

#1 Event Project Plan Development - By ___September 1st_____

#2 Event Data Gathering and processing- By ___October 1st_____

#3 Event Extensibility and Final Testing + Documnetation - By ___November 25th_____

Meeting Schedule Date/Time

Every Monday @ 8.

Collaboration and Communication Plan

While we are all on Teams, we feel it is most appropriate for us to all meet in person and therefore we meet every Monday @ 8 PM.

Project Schedule and Task Planning

Attached to the assignment submissions

Contents

Cybriant Project Plan 1
1.0 Project Overview / Abstract (Research) 2
2.0 Project website 2
Deliverables - Specific To Your Project 2
Milestone Events (Prototypes, Draft Reports, Code Reviews, etc) 2
Meeting Schedule Date/Time 2
Collaboration and Communication Plan 2
Project Schedule and Task Planning 3
Introduction 4
1.0 Introduction 4
1.1 Overview 5
1.2 Project Goals 5
1.3 Definitions and Acronyms 5
1.4 Assumptions 5
2.0 Design Constraints 5
2.1 Environment 6
2.2 User Characteristics 6
3.0 Functional Requirements – EXAMPLE for MOBILE APP 6
4.0 Non-Functional Requirements (use if applicable) 7
4.1 Security 7
4.2 Capacity 7
4.3 Usability 7
4.4 Other 7
Introduction and Overview 10
Design Considerations 11
Assumptions and Dependencies 11
General Constraints 11

Development Methods 12
Architectural Strategies 12
System Architecture 12
Detailed System Design 12
Classification 12
Constraints 13
Resources 13
Interface/Exports 13
Glossary 13

Introduction

In collaboration with Cybriant, a leading Managed Security Service Provider based in Alpharetta, GA, this project aims to develop an advanced Attack Surface Management (ASM) system tailored for mid-market organizations. Cybersecurity has become an essential factor in shaping companies' operational integrity and financial stability, with security ratings such as the BitSight score playing a pivotal role in determining cybersecurity insurance premiums and vendor trust. While existing ASM solutions cater primarily to large enterprises with 2,000+ employees, their high cost and complexity often make them inaccessible to smaller businesses. Cybriant seeks to bridge this gap by creating a cost-effective and scalable solution designed specifically for companies with 500–2,000 employees. Leveraging Google Cloud services, including Google BigQuery and Google Chronicle SIEM, this system will provide organizations with actionable insights to enhance their security posture, improve their BitSight score, and reduce insurance costs. The project also provides Kennesaw State University (KSU) students a unique opportunity to gain practical cybersecurity experience through hands-on development and integration within a real-world infrastructure.

Requirements

1.0 Introduction

The basis of our project is to work almost like a credit reporting agency but in terms of cybersecurity instead of credit worthiness. This means we have to take into account security policies and technologies in place to determine the overall risk and security of a corporation as a whole. Our leaders in this project, which is Cybriant employees, have told us that we are also expected to be connected to some cloud services offered through Google to be able to run some of the data through a database and other programs to create the logic to achieve our goal. What is expected of us is a broad range of skills going from data science and software engineering, to cybersecurity and statistics.

1.1 Overview

We are going to get this done in a matter of steps. The first matter would be collecting data from different sources. For the purpose of this project, Cybriant will be providing us with a sandbox environment to test out our programs as well. We will collect data that is both public and confidential. We have to collect this data and feed it into Google BigQuery which we will have to do by creating a program that can connect to this cloud service

After we have collected the data, we will then need to run it through some of machine learning program which can spit out a predicted score based off data received. We will have to write our own logic code and create our own program for receiving the data and analyzing it so this will be a big challenge on its own as well.

Finally we will have a predicted “cybersecurity” score and we will be able to act upon that. While this program could be used for insurance premiums to cover any type of measurement for liability we could also use this program to increase security for some of these companies. Since Cybriant is also a company that trains cybersecurity teams for companies this can be quite useful for them.

1.2 Project Goals

Our Project goal is mainly to create a software that is able to accurately rate companies as well as point out their weaknesses and strengths. We want to be able to create a BitSight rating so that we could potentially lead more companies to be more secure with their information

1.3 Definitions and Acronyms

Cybriant- The company sponsoring this that is focused on cybersecurity services

BitSight Security rating – Powerful tool used to assess, monitor, prioritize, and commiunicate cyber risk

1.4 Assumptions

Some assumptions I have for this project is that it is going to be used by the company if any team makes a good and effective software. I also believe that this will increase my cybersecurity skills and knowledge. This project will be tougher than some of the other options but I think me and my team can handle it. And we will be using a multitude of software and programming techniques to accomplish this task.

2.0 Design Constraints

Some Design Constraints could be the sheer storage and processing of huge amounts of data. It will take a decent amount of power to use some of this software. As a result of the amount of data it could also effect the data quality and reliability. One of the bigger challenges might be integrating the system with Google Chronicle. The last two constraints might be scalability and the most important is

security and privacy. If the database for this software were to be hacked it could be a very catastrophic situation.

2.1 Environment

The digital environment will be a sandbox free from resources and information that could be put at risk while developing this program. If this program were to be used without being tested it could result in huge security risks and consequences for the Cybriant organization.

2.2 User Characteristics

Most people who used this software are going to be working for a company and this program will not be open source. Therefore, the users for this would most likely be employees with varying levels of technical expertise. They would also be familiar with cybersecurity concepts. The users would also be aimed at towards mid-market to lower size enterprises.

3.0 Functional Requirements – EXAMPLE for MOBILE APP

3.1 Data Collection and Processing

3.1.1 Collect security metrics from various sources, including publicly available information, internal systems, and external threat intelligence.

3.1.2 Process and analyze collected data to identify security vulnerabilities, risks, and trends.

3.1.3 Integrate with external data sources, such as threat intelligence feeds and vulnerability databases.

3.2 Risk Assessment and Reporting

3.2.1 Generate a comprehensive risk profile for organizations, highlighting critical vulnerabilities and potential threats.

3.2.2 Prioritize risks based on their severity and potential impact.

3.2.3 Provide customizable reports that can be shared with stakeholders.

3.3 Visualization and User Interface

3.3.1 Use clear and informative visualizations to present complex data in a digestible format.

3.3.2 Offer a user-friendly interface that is easy to navigate and understand, even for non-technical users.

3.3.3 Enable customization to meet the specific needs and preferences of different users.

3.4 Integration and Extensibility

3.4.1 Seamlessly integrate with Google Chronicle to provide a holistic view of an organization's security posture.

3.4.2 Allow for integration with other security tools and platforms.

3.4.3 Design the system to be easily extensible to accommodate future data sources, analysis features, and integrations.

3.5 Security and Privacy

3.5.1 Comply with relevant security and privacy regulations, such as GDPR and HIPAA.

3.5.2 Implement robust security measures to protect sensitive data.

3.6 Performance and Scalability

3.6.1 Be able to process and analyze large volumes of data efficiently.

3.6.2 Scale to meet the needs of growing organizations and increasing data volumes.

4.0 Non-Functional Requirements (use if applicable)

4.1 Security

The system must implement robust security measures to protect sensitive data, including encryption, access controls, and intrusion detection systems. Regular security audits and vulnerability assessments should

be conducted to identify and address potential security risks. Compliance with relevant security and privacy regulations, such as GDPR and HIPAA, is essential.

4.2 Capacity

The system must be designed to handle large volumes of data and scale to meet the needs of growing organizations. It should be able to perform efficiently under peak loads and maintain high availability.

4.3 Usability

The system should have a user-friendly interface that is easy to navigate and understand, even for non-technical users. Clear and concise documentation and tutorials should be provided to support users. Responsive customer support is essential to address user inquiries and issues promptly.

4.4 Other

The system should be compatible with different operating systems and browsers. It should be optimized for performance and speed. Accessibility requirements should be considered to make the system usable by individuals with disabilities.

5.0 Analysis

1. *5.1 Stakeholder Analysis*

The key stakeholders for this project are Cybriant, the development team (students), and the target customers, which are mid-market companies. Cybriant, as the project sponsor, seeks to create a cost-effective and scalable Attack Surface Management (ASM) system to address the cybersecurity needs of companies with 500–2,000 employees. Their primary objective is to offer a solution that enhances their customers' security posture, integrates seamlessly with Google Chronicle SIEM, and provides actionable insights to improve

BitSight-like security scores. For Cybriant, the success of this project also represents a significant market opportunity to fill a gap in the current landscape of ASM products.

The student development team benefits from this project by gaining real-world experience in cybersecurity, data science, and software engineering. By contributing to a practical project with direct business impact, students not only enhance their technical skills but also gain valuable exposure to advanced tools like Google BigQuery, Python, and Google Cloud infrastructure. This hands-on experience prepares them for future careers in the cybersecurity and technology sectors. The target customers, mid-market companies, require an affordable tool to assess their cybersecurity posture effectively. They aim to identify vulnerabilities, improve their security ratings, and reduce their cybersecurity insurance premiums. By addressing this need, the project delivers significant value to businesses that lack access to expensive enterprise-focused solutions.

Secondary stakeholders include cybersecurity insurance providers, who can use the generated security scores for risk assessments and policy decisions, and security teams at mid-market companies, who will rely on the platform to manage vulnerabilities and implement proactive measures.

5.2 Feasibility Analysis

The project demonstrates strong feasibility in technical, operational, and resource terms. From a technical perspective, the use of established tools like Google BigQuery, Google Cloud Run, and Google Chronicle SIEM provides a robust foundation for handling large-scale data processing and analysis. These tools are well-documented and widely used in the industry, ensuring compatibility with the project's requirements. The chosen technology stack, including Python, Node.js, and Express, is suitable for building the backend and aligns with the team's skill set.

Operationally, Cybriant’s sandbox environment offers a secure and controlled space for development and testing. This reduces the risk of exposing sensitive customer data during the development phase. The project milestones and timeline are well-defined, making the objectives achievable within the given constraints. In terms of resources, Cybriant has provided access to the necessary tools and infrastructure while offering mentorship to the development team. This ensures the project remains on track and benefits from expert guidance, addressing any knowledge gaps that may arise.

5.3 Risk Analysis

The project involves several risks that must be mitigated to ensure success. Data security and privacy are critical concerns, as unauthorized access to sensitive information during development could lead to reputational damage or legal issues. This risk is mitigated by using sandbox environments, encrypting data transmissions, and implementing strict access controls. Another significant risk is the integration of the system with Google Chronicle SIEM and other APIs, which could present technical challenges. This risk can be addressed by leveraging Cybriant engineers' expertise and utilizing comprehensive documentation and support provided by Google Cloud.

Scalability and performance also pose challenges, as the system must handle large datasets efficiently. To address this, the team can leverage BigQuery's scalability features and optimize the backend code for performance. Additionally, the complexity of using advanced tools like machine learning and APIs may highlight gaps in the development team's expertise. These gaps can be mitigated through mentorship from Cybriant and time allocated for research and learning.

5.4 Gap Analysis

The current market for ASM solutions is dominated by expensive, enterprise-focused products such as BitSight and SecurityScorecard. These solutions cater to large organizations with 2,000 or more employees, leaving mid-market companies underserved. Many smaller organizations lack access to affordable, scalable tools to assess and improve their cybersecurity posture effectively. This project addresses this gap by developing a mid-market-focused ASM solution that provides actionable insights and integrates with Google Cloud tools.

The desired state is an affordable, extensible solution tailored specifically for mid-sized companies. By leveraging existing infrastructure and developing a scalable, data-driven system, the project offers a unique value proposition that bridges the gap between high-end ASM products and the needs of smaller organizations. The integration with Google Chronicle further enhances the product's capabilities, making it a robust tool for both security assessment and risk management.

5.5 Functional Requirements Analysis

The project's functional requirements align directly with its objectives and stakeholder needs. The system's data collection and processing capabilities ensure comprehensive security metrics by sourcing information from both public and private datasets. The integration of external data sources, such as threat intelligence feeds, strengthens the accuracy and reliability of the collected data. This robust data collection forms the foundation for risk assessment and reporting, which generate detailed security profiles and prioritize vulnerabilities based on severity and impact. The ability to create customizable reports adds value for stakeholders by supporting clear communication and compliance with regulatory requirements.

The system's user interface and visualization features enhance usability, presenting complex data in a clear and digestible format. The design of the interface prioritizes accessibility, ensuring it is easy to navigate even for non-technical users. Furthermore, the system is designed with extensibility in mind, allowing for seamless integration with Google Chronicle and future enhancements, such as additional data sources or advanced analytics features. This flexibility ensures the solution remains relevant as cybersecurity threats evolve.

Conclusion

The analysis of stakeholders, feasibility, risks, gaps, and functional requirements demonstrates that the project is well-positioned to achieve its goals. By addressing the unique needs of mid-market companies and leveraging advanced tools and technologies, the project fills a critical gap in the ASM market. With strong operational support from Cybriant and a clear focus on scalability and extensibility, the project has the

potential to deliver significant value to all stakeholders while providing the development team with invaluable real-world experience.

ChatGPT can make mistakes. Che

This concludes the requirements documentation for our project.

System Design

Introduction and Overview

The Cybriant Attack Surface Management system aims to provide organizations with a comprehensive assessment of their security posture, identify vulnerabilities, and offer actionable recommendations for improvement. By analyzing various security indicators, the system will help organizations enhance their security hygiene and reduce their cybersecurity insurance premiums.

This can used for both insurance reasons and also to help develop and strengthen organization's security.

The project encompasses the development of a system that can:

1. Collect and analyze security metrics from multiple sources.
2. Assess an organization's security posture and identify vulnerabilities.
3. Generate detailed risk profiles and recommendations for improvement.
4. Integrate with Google Chronicle and other existing security tools.
   Some of the goals we have for the project is:
   1. Improve the security posture of Cybriant's customers.
   2. Reduce cybersecurity insurance premiums for customers.
   3. Provide actionable insights to help organizations address security vulnerabilities.
   4. Enhance Cybriant's reputation as a leading cybersecurity provider.

Design Considerations

Assumptions and Dependencies

- The system will rely on publicly available data and data from Cybriant's existing infrastructure.
- The system will integrate with Google Chronicle, assuming compatibility and API availability.
- Users will have varying levels of technical expertise and familiarity with cybersecurity concepts.
- The system will be used by mid-market to lower enterprise organizations.
- The system will operate within the constraints of Cybriant's existing IT infrastructure.

General Constraints

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

1. **Data Quality:** The quality and reliability of publicly available data may vary.
2. **Scalability:** The system must be scalable to accommodate a growing user base and increasing data volumes.
3. **Performance:** The system must provide timely and accurate results, even with large datasets.
4. **Integration:** The system must integrate seamlessly with Google Chronicle and other existing security tools.
5. **Security:** The system must adhere to strict security standards and protect sensitive data.
6. **Cost:** The development and maintenance of the system must be cost-effective.

Development Methods

Agile Methodology- An agile development approach will be used to ensure flexibility, adaptability, and continuous improvement throughout the project.
Iterative Development- The system will be developed in iterations, with each iteration incorporating feedback and improvements based on testing and evaluation.

Architectural Strategies

The system will employ a centralized data repository to store and manage collected data, along with robust data governance policies to ensure data quality, integrity, and security. Integration with Google Chronicle and other external systems will be facilitated through APIs, and mechanisms will be implemented to synchronize data between the system and external sources. To safeguard sensitive data, role-based access controls will be enforced, and data will be encrypted both at rest and in transit. Regular security audits and vulnerability assessments will be conducted to identify and address potential security risks, ensuring the system's overall security and reliability.

System Architecture

For this semester, the system will consist of five primary components: a data ingestion module, a data processing module,a risk assessment module, a reporting module, and an integration module. The data ingestion module will collect data from various sources, including publicly available information and internal systems. The data processing module will process and analyze collected data to identify security vulnerabilities and risks. The risk assessment module will assess the severity of identified vulnerabilities and generate comprehensive risk profiles. The reporting module will generate reports and visualizations to effectively communicate findings to users. The integration module will facilitate seamless integration with Google Chronicle and other external systems.

Detailed System Design

Classification

1. **Data Sources:** Define the specific data sources that will be used, such as public DNS records, web application vulnerability scanners, and threat intelligence feeds.
2. **Data Extraction:** Develop methods to extract relevant data from these sources.
3. **Data Normalization:** Standardize data formats and ensure consistency across different sources.Definition

The specific purpose and semantic meaning of the component. This may need to refer back to the requirements specification.

Constraints

1. **Data Analysis:** Implement algorithms and techniques to analyze collected data and identify security vulnerabilities.
2. **Scoring Mechanism:** Develop a scoring mechanism to assess the severity of identified vulnerabilities.
3. **Baseline Establishment:** Establish baseline metrics to compare against current security posture.

Resources

1. **Risk Prioritization:** Prioritize identified vulnerabilities based on their severity and potential impact.
2. **Risk Mitigation Strategies:** Provide recommendations for mitigating identified risks

Interface/Exports

1. **Report Generation:** Generate customizable reports that can be shared with stakeholders.
2. **Visualization:** Create visualizations to present findings in a clear and understandable manner.

Glossary

- **Attack Surface:** The portion of a system that is exposed to potential attacks.
- **Security Posture:** The overall state of an organization's security practices and defenses.
- **Vulnerability:** A weakness in a system that could be exploited by an attacker.
- **Risk:** The likelihood and potential impact of a security vulnerability being exploited.

Test Document

1. Testing Strategy

1.1 Overall strategy

● UnitTesting: Testing individual functions like get_spf, get_dkim,

analyze_spf_security, and dns_punycode_analysis to verify that they return

expected outputs for given inputs. The developer team performs this testing to ensure that

the function works correctly.

- Integration Testing: Verifying that modules interact correctly, such as how prepare_security_data handles outputs from the individual security-checking functions.The developer team will work on this to ensure that the modules are working correctly when passing them through each other.
- SystemTesting: Testing the complete workflow from domain data collection to successful insertion into BigQuery to ensure end-to-end functionality. The testing team will review this procedure making sure the system is functioning.
- Regression Testing: Retesting modified code to ensure new changes don’t introduce new bugs or break existing functionality.This testing is done by the testing/QA team who is responsible to check possible ways of crashing the program and preventing it.

1.2 Test Selection

- Black-Box Testing: We used Black-Box Testing for system testing, focusing on expected outputs without knowledge of the code's internal workings.Our testing team performed this testing
- White-Box Testing: Applied for unit tests to ensure individual functions provide correct outputs, testing boundary conditions and handling of invalid inputs.

Techniques:

- Equivalence Partitioning: Grouping inputs for domain analysis and verifying representative cases.
- BoundaryValue Analysis: Testing edge cases, such as domains with/without DNSSEC or invalid SPF rec
- ErrorGuessing: Manually testing likely failure points, like malformed JSON in SSL data.

1.3 Adequacy Criterion

- UnitTesting: Achieve 90% code coverage, ensuring all critical paths and edge cases are tested.
- SystemTesting: Ensure all workflows execute successfully for a variety of real-world domains.
- Integration Testing: Verify data integrity as it flows through the system and interactions among components.

● Regression Testing: Confirm that all previously passing test cases remain valid after code changes.

For the creation of our ASM (Attack Surface Management) application, we've settled on architecture based on one suggested by Cybriant. This architecture involves three major systems: a series of Python Scripts, a Google Cloud Run environment, and a Google BigQuery database/storage system. Between these components there exist a set of docker and requirement files for integrating our software into the cloud environment. As such, we've chosen to break down our testing by isolating the three major systems and their respective components, before handling configuration files for the cloud environment which will further be tested.

The Python Scripts will be compiled in a composite file titled Main.py. This file contains the 10-security metrics we've selected as the basis for our system. Additionally, it contains the necessary connections to implement these files into BigQuery. Each security function will be tested separately to ensure compliance, and the file will be tested in the cloud once the environment is established.

Any testing done on Google Cloud Run will be using the logging feature present within the platform. It breaks down fault points, expected outputs, and other behavior expected within a given job or service. Diwakar Rai will be using these logs to evaluate the overall program via system testing once complete.

BigQuery will comprise much of the integration testing cycle as any obvious faults or changes needed will appear here. Our team will be pushing all versions of our program to the GitHub repository granted to us by Cybriant. Through it, we will conduct regression testing (by Jose and Daniel) to ensure each component works before pushing changes to the Cloud. Additionally, any database changes or connections will be monitored by Nic and David, though we expect container + job set up to be a one-time endeavor.

1.4 Bug Tracking

All bugs will be cataloged and tracked using GitHub Issues within our repository. As each team member is expected to work on their given pair of security metrics (and each metric has specialized information), tickets will largely be handled by the individual that submits them. Additionally, we have bi-weekly meetings with our Cybriant contacts where we bring up any major tickets published to the repository.

Enhancements are discussed during team meetings or via our Teams channel. They are discussed before being pushed and are tracked via branches from the main repository,

1.5 Technology and Tools

- UnitTesting: unittest (Python standard library).
- SystemTesting: Google Cloud SDK for BigQuery interactions.
- NmapTesting: python-nmap.
- DataValidation: bigQuery Data preview.
- Version Control: GitHub
- BugandTickets: GitHub Issues

2.TestCases

Testingdomain:salesforce.comandFacebook.com

Test Case ID	Purpose	Steps	Expected Result	Actual results	Pass/Fail	Additional Information
TC001	Validate SPF record retrieval	Input a domain Call get_spf	Returns the correct SPF record a message indicating no record was found			Ensure domain is reachable.
TC002	Validate database entries in BigQuery	1 Open Google BigQuery under project repository 2 Call table preview to view recent entries	Returns a table containing all function outputs in their expected locations			Ensure outputs match CR logs and fields match their expected values.
TC003	Validate DNSSEC check	1 Input a domain with DNSSEC 2 Call check_dnssec	Returns true			Ensure a site has DNSSEC enabled
TC004	Validate BigQuery insertion	1 Collect valid data 2 Call insert_security_data	Data is successfully inserted into BigQuery			Esnsure proper BigQuery

			without errors			permissions are set up.
TC005	Test active squatting domain retrieval	1 Input a domain 2 Call check_domain_squatting	Returns a list of squatting domains or an empty list			Limit possible domainsto those allowedby TLD protocols.
TC006	Validate SSL data retrieval	1.Input a domain 2 Call fetch_certificate	Returns SSL certificate details			Verify against known valid SSL certificates.
TC007	Validate Nmap SYN Scan	1 Input a valid IP and a given port to scan 2 Call nmap_syn	Returns scan results for specified port via a half-open scan			Ensure Nmap is installed and accessible.
TC008	Validate DMARC record	1.Inputa domain. 2.Call get_dmarc_r ecord(domai n)	Returnsthe DMARC records for the given site			Test sites With known records, many lack the security function of DMARC records.
TC009	Validate Docker file for container creation	1 Pull all dependencies and PAckahaes from Main + Requirements 2 Execute docker run in a local environment	Establishes local image/docker and run executes successful			Ensure all dependencies are present when running the container
TC010	Create a Cloud Run Container	1Call docker push image to store into registry	Container appears in the repository and accepts Docker files			Check the project registry to ensure the container image exists

TC011	Image execution within CR container	1 Run main.py in the container	Executes successfully and shows function outputs in the terminal			Validate expected outputs against CR logs for scripts
TC012	System testing: Script Execution within CR container	1Execute the CR job for a set of parameters 2 Ensure logging in the terminal as the job executes 3 Validate security metrics within BigQuery	Accepts input for each function within Main.py storing data in Google BigQuery Data appears as intended and all connections are valid			Ensure the container can run on schedule without manual input

Version Control

Version control was a critical component of our project to ensure efficient collaboration and maintain the integrity of our codebase. We utilized a private GitHub repository to store, track, and manage our code throughout the development process. This approach allowed our team to work concurrently on different aspects of the project while ensuring that changes were systematically tracked and easily reversible if needed. GitHub's branch and pull request system enabled us to implement a clear workflow for code reviews, ensuring that every update met our quality standards before being merged into the main branch. Additionally, the repository provided a centralized location for storing all project-related files, facilitating seamless integration and access for all team members. By leveraging version control with GitHub, we maintained a robust and organized development process, minimizing conflicts and enhancing collaboration efficiency.

Conclusion

The Cybriant Attack Surface Management project exemplifies a collaborative effort between Kennesaw State University students and Cybriant to address a pressing need in cybersecurity for mid-market companies. This initiative aimed to bridge the gap in affordable, scalable Attack Surface Management (ASM) solutions, enabling

companies to enhance their security posture, improve risk assessments, and reduce cybersecurity insurance premiums. Leveraging advanced tools like Google BigQuery, Google Cloud Run, and Google Chronicle SIEM, the project delivered a comprehensive system that integrates data collection, risk assessment, and actionable reporting.

Throughout the project, the team applied diverse skill sets in data science, software engineering, and cybersecurity, culminating in the development of 718 lines of Python code, a functional Docker environment, and comprehensive documentation. The iterative approach, supported by Agile methodologies, allowed the team to navigate challenges such as scalability, security integration, and processing vast amounts of data. Regular team meetings, mentorship from Cybriant, and the use of version control with a private GitHub repository ensured collaboration efficiency and project success.

This project not only addresses an underserved market but also highlights the capabilities of students in delivering a high-impact solution to real-world problems. The integration of innovative technologies, robust testing, and practical experience positions this ASM system as a significant step forward in improving the cybersecurity landscape for mid-sized enterprises. The project underscores the value of academic-industry partnerships in fostering innovation and equipping future professionals with the skills needed to excel in a rapidly evolving field.