Portant - Protecting Sensitive Data from Multiple Sources

Scoping Document

COMP3850 – Computing Industry Project

Group 37 – Bradley Anderson, Socheat Chhun, Arshita Jaryal, Md Mehedy Hasan, Jarrod Adair, Edward Morris

Contents

1. Table of Revisions

Version	Publish Date	Change Description/Features
1.01 Scoping Document	24/04/2021	Section 2.6 Deployment – monitoring, reporting, review, training/support, security polices). Added training and support section Section 2.4 Attacks and Vulnerability Detection – Added more detail to this section
1.02 Scoping Document	28/04/2021	Final edits and changes made to Section 3.5 to make it more relevant and readable

2. Analysis Overview

2.1. Purpose and scope

The company sponsoring the project, Portant has an application that can request and displaying data in a concise format using Google Docs via the Google API. What is missing and what the aim and scope of the project is to create an application which is capable of what Portant's original product can do with the addition of security measures that allow for the secure and verifiable requesting and delivering of information. Such an application would allow for a more seamless experience of requesting and displaying large quantities of data from groups of people while also ensuring that data remains secure.

Currently without such measures in place an attacker could hijack information as it is being requested or delivered and have access to the information that is being requested or sent. With such info an attacker has access to potentially sensitive data or if they are particularly nefarious can mimic legitimate traffic to feed false information to a document. Addressing the security issues will result in an application that can securely transport mass amounts of data between entities in an organization and display the results neatly in a Google Doc, which makes the application more marketable to security conscious organizations looking for information streamlining software.

The project will involve us undertaking some blue team activities mixed with application development, since the framework needs to be built to allow for requesting/providing of information before encryption and verification is implemented for the requesting, providing and transfer of information. This will require us to evaluate different methods of implementing encryption and identity verification and seeing which ones suit the application that is being developed.

2.2. Attacks/Vulnerability Understanding

In our project we have users, respondents, the application, and the Google API. See Figure 1 below.

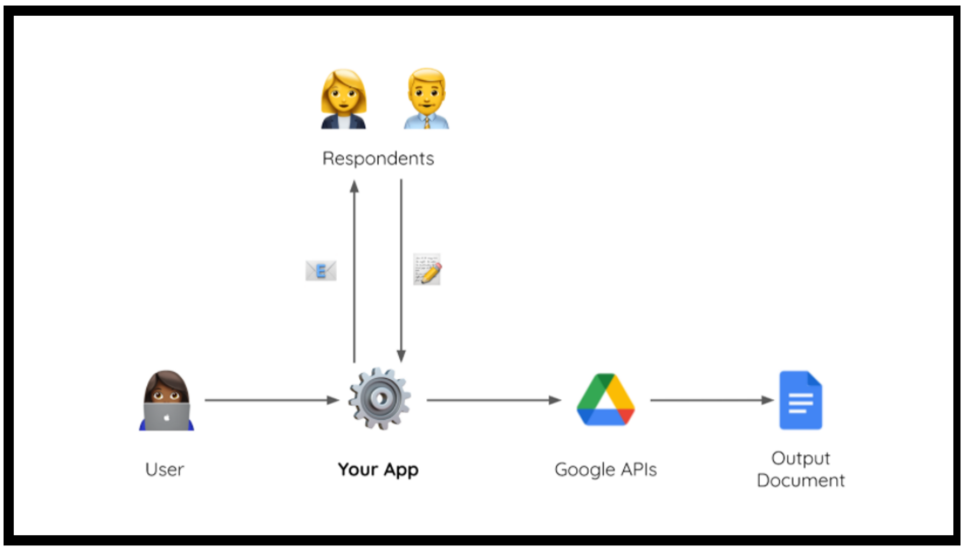

Figure 1: High level project concept

Here we have the user and respondents interacting with the application. We have communication channels going between the user to application, application to respondents and vice versa. Our main security concerns lie in three areas:

Web application security

Cross-site scripting attack, Cross-site request forgery, SQL injection attack, Insecure direct object reference attack etc.

Data security

Security of data both when at rest and in transit between entities.

Attack vectors

Attack vector is a method used by an attacker to gain access to victims' computer or machine and infect the machine with malicious software. It is also a method to steal victim's data. Attack vectors are listed below.

- Eavesdropper
- Man-in-the-middle.
- Malware
- Ransomware
- Compromised credentials
- Poor encryption
- Distributed Denial of Service
- Cross site scripting
- Session Hijacking

2.3. Adversaries

The kinds of adversaries that our project will defend against fall under a few distinct groups, these being:

2.3.1. Rival Companies

A company whose competitor has information moving through our application may be susceptible to a hijacking of information to try and find trade secrets or other sensitive information. A rival company could hire people to attempt to break into the application's servers looking for information pertaining to that specific company or other companies. Depending on the funding available these attackers could have a vast amount of equipment and funding behind them and thus could execute large-scale attacks with sophisticated tools.

2.3.2. Criminal Groups/Cyber Criminals

Criminals that target high value targets for information to sell online will vary in ability and capabilities. A group of cyber criminals can crack a great deal of systems, so provided the perceived value is high enough they might target the application's systems. These could technically fall under the previous category but without the backing of a company. Due to the lack of a backer a criminal would indiscriminately steal data and sell it on a black market to the highest bidder, making them harder to keep a paper trail tied to another company or organization.

2.3.3. Rogue Insider/Ex-employee

If an employee of the company is disgruntled about treatment, pay or anything else work related, they might seek to expose company or client data. They might be motivated by revenge or taking information for sale on the black market. These kinds of adversaries are particularly dangerous since they have insider knowledge of the company and will know the procedures that go on within the programs that the company uses. As such these are dangerous adversaries as their presence can go unnoticed for a longer period since their attacks can come from within the company itself.

2.4. Attacks/Vulnerability Detection

For data communication over the internet/network, SSL/TLS protocol will be used. This protocol has detection mechanisms for data tampering or falsifying.

A web application vulnerability scanner such as BurpSuite will be used to scan the application regularly to find or detect vulnerabilities in the application. The scanner will find holes in application processes and communications that would allow for exploits like XSS, SQL injection and path traversals to be performed on the application. Once uncovered, they can then be addressed by developers and the security team. [1]

For monitoring the web application, external software such as Nagios or SolarWinds can be used. These systems allow for precise monitoring of the system, enabling the team working on security to easily detect if there is a threat. External software is a cost-efficient and secure way of preventing attacks and identifying system vulnerabilities. [2]

A SIEM product will be implemented for the collection and analysis of logs for threat detection and incident response.

2.5. Assumptions

- Services used such as Google API, Amazon AWS, Email and Twilio will be up and running almost all the time.

- The previously mentioned services have equal to or greater security than our application (else the ‘weakest link’ lies in external systems which we have no control over).
- That the team will keep in regular communication with each other and the sponsor to ensure that team members and the sponsor know where the project currently stands.

Our application will be leveraging external services to maintain functionality. For this to be employable, assumptions must be made on the effectiveness and availability of these third-party services. Resources such as AWS Cloud, Google Mail and Twilio, which our application will heavily rely upon, has a high standard of security which will ultimately shape our security posture. A security risk may arise however if these services become faulty or are compromised. Our assumption is this anomaly will not occur such that it will affect the performance of the application.

The budget of the development would also be a considerable concern in regular circumstances. As our expectation is this application will serve as a proof of concept prior to it being considered a real solution intended to be implemented, our assumption is there is no budget for developing and implementing our app.

2.6. Document Conventions and Audience

This document is intended for all stakeholders of the project (team members and project sponsors, as well as unit convenors for the unit COMP3850). These stakeholders will want to be informed about the scope of the project and factors identified that will influence the design and development of the project.

The document will detail things such as the scope and purpose of this document, identified threats and actors in the system, the threat detection methods being considered as well as the assumptions being made when developing the system. These are covered in sections 1.1 – 1.6.

Sections 1.7 and 1.8 will detail data being collected by the application and any preparation data undergoes before any processes the application does.

Section 2.1 details the security features that are being considered at different points in the application.

Sections 2.2 and 2.3 detail vulnerabilities and threats to the application’s security as well as the defensive countermeasures that are available and are being considered to bolster application security.

Section 2.4 will explain the system architecture, what parts of the application interact with each other and the processes that occur between them, as well as data that flows through them.

Sections 2.5 and 2.6 detail the penetration testing methods that will be used and at what stages of development they will take place as well as the application deployment, maintenance, reporting and security policies that will be in place to maintain secure operations.

2.7. Data (packages, logs, …) Collection

The collection of data will entail extracting information from cookies for the purpose of monitoring session keys. This is necessary to prevent malicious activities such as replay and masquerade attacks.

Other methods of data collection will be from user input, where identifiable data such as usernames and passwords will be documented. As the information being requested will potentially be sensitive Portant data, this information will be stored using encryption services, as well as hash digests for the storage of passwords.

Relevant legislation that will be considered is the Australian Privacy Act 1988, which specifies the rights of the individual during the data collection process. The General Data Protection Regulation (GDPR), which details how to store information of users based in Europe, will also be considered for the purpose of scalability.

2.8. Data Preparation

Data captured throughout the application will require processing to store the data correctly. Data profiling is the process of reviewing the collected data and identifying anomalies and inconsistencies. After this, data cleansing ensures correct, complete, and accurate data for storage and analysis purposes. This relates to removing or invalidating faulty or inconsistent data inputs. [3]

Although data preparation is primarily used for analysing data sets such as primary data collection, with further steps requiring the structuring, transformation, and publishing of data, the Portant application will not involve these features. The profiling and cleansing stages will however provide useful processes for ensuring consistency throughout the collected information.

3. Analysis activity details

3.1. Security features extraction

3.1.1. Web Application Security

Since this is a web application, all web application vulnerabilities are threats. The most obvious being XSS (Cross Site Scripting), CSRF, and SQL injection. These are the most common vulnerabilities that our web application is susceptible to. We are considering all these attack possibilities that can be malicious to our application.

Since our application are utilizes Google services, Google provide us with their API Key. One of the vulnerabilities is not storing this API key in a secure place.

3.1.2. Data at Rest

We are including any types of storage of data. It could be files shared in application and information stored in database to any kinds of logs etc.

The security properties are confidentiality, integrity and availability which are also called CIA triad.

Confidentiality – protecting unauthorised access to data and subsequent misuse of it.

Integrity – Data has not been tampered with and is accurate and consistent.

Availability – Data should be available to authorised parties.

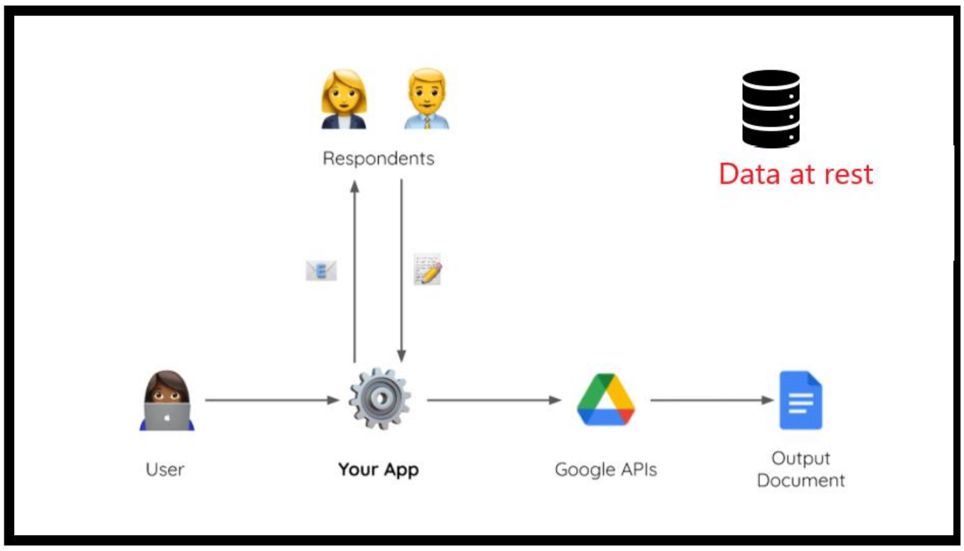

Figure 2: Data at rest

When we have data at rest, we must consider that users are authenticated to access the data. Sensitive data such as password in the database need to be protected with proper security measures, ensuring confidentiality. Data logs and anti-tampering measures will ensure data integrity and proper authorisation protocols as well as web availability safeguards will ensure availability.

3.1.3. Data in Transit

The security concern here is the connection between user to app and respondents to app. This connection will be made via the internet. Any connection via the internet is considered an unsecure channel.

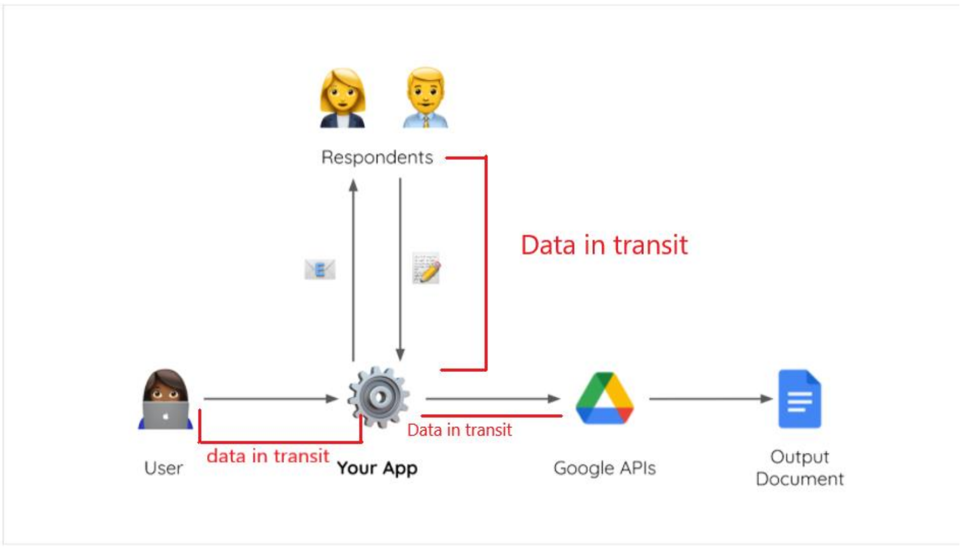

Figure 3: Data in transit

Security weakness to be considered here are the protocols that are used to communicate between user to app. Questions to be asked here: is the data communication between users/respondent to app is encrypted? Is there any protocol being used that can protect against data integrity while data in transit? Does this communication satisfy the CIA triad?

3.2. Attacks/Vulnerability Analysis

3.2.1. Eavesdropper Attack

As discussed in Section 2, communications between user/respondents and the application are always an unsecured channel.

Eavesdropping is an easy attack to perform and can be very passive and undetectable. An attacker can simply sit between the users/respondents and the app and capture or sniff network/internet traffic for later analysis.

An attacker is usually after sensitive information such as user credentials, financial data, business data etc.

As mentioned before, assume that communications are always on an unsecure channel and we will have to assume that an eavesdropper is always sniffing or capturing communications between users, respondents and the application.

3.2.2. Man-in-the-Middle Attack

A Man-in-the-Middle (MIMT) is an extension of an eavesdropping attack. An attacker intercept traffic between two parties and either steals or modifies the traffic before it reaches the receiving party01.

3.2.3. Poor Encryption

Encryption is the biggest part of information security. It is a process that scrambles the plain text to unreadable text or cipher text. Strong encryption helps to ensure confidentiality of the CIA triad. If weak or poor encryption is used for data in transit, then an eavesdropper or man-in-the-middle could easily decrypt that data. That could cause serious issues such as the loss of user credentials or the loss of personal/business data to the malicious party.

If there is no encryption used on user credentials in the database, if attackers breach the database, they can make off with un-encrypted credentials with no further hurdles to get over to gain access.

The sponsor informed us that SSL/TSL version 1.2 is being used for transport layer security in the existing system.

This version has several security issues and is vulnerable to attacks like POODLE, BEAST, CRIME, BREACH, HEARTBLEED, 3SHAKE, and Raccoon.

In SSL/TLS, there is key management involved. When this protocol is being used a certificate and key is provided. If an attacker can get access to this key, then they will be able to decrypt the encrypted data flowing through the application and it’s communications channels. [4]

3.2.4. Malware

Malware is an all-encompassing term that describes any software specifically designed for malicious purposes. Malware can be used on computers, servers, or an entire network; it can be used for the purpose of distributing spyware, installing rootkits on the host machine, and implementing ransomware products, among others.

Malware is typically distributed in three common methods:

- A worm, which is a self-reproducing program which can easily spread across a network.
- A virus, a hidden code inside of another ‘legitimate’ reproducing program, which is typically activated upon user interaction.
- A trojan is like a virus, in that it is a hidden program inside of another legitimate appearing program. A trojan however, is unable to self-reproduce.

3.2.5. Ransomware

Ransomware is a widely used malicious software that encrypts a users’ files, denying access to them until a fee is paid, or the files are deleted. Typically, these fees are paid by cryptocurrency, and are therefore exceedingly difficult to trace. This makes ransomware a low-risk, high-reward attack as they can be widely distributed through phishing emails and other low-cost transmission methods. The most feasible method of prevention is ensuring up-to-date operating systems, as regular patches are required to remove known vulnerabilities.

3.2.6. Compromised/Weak credentials.

Compromised or weak credentials are the most common type of attack vectors and If credentials are lost or stolen then users could lose data through breaches or leaks. A weak password is also vulnerable to dictionary or brute force attacks.

3.2.7. Distributed Denial of Service

A Distributed Denial of Service is an attack that floods a service with too many requests for it to handle, slowing the service down for legitimate users. Such an attack could take the application down for several hours as it tries to filter through the junk requests being sent to it.

3.2.8. SQL Injection

SQL injection is another common web application security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. Despite being an attack that is rather old in its origin, it is still prevalent due to the lack of proper precautionary measures against such attacks. SQL injection can be harmful because if your site is vulnerable to this kind of attack, the attacker can easily have read, write and viewing access to your databases. Moreover, with a tool like SQLmap, it makes it even easier for attacker to quickly intrude your database.

3.2.9. Cross Site Scripting

Compared to Cross-site request forgery, cross-site scripting can be more versatile because attackers can write any code in JavaScript to change the normal behaviour of a web application. In a **cross-site scripting** attack, the attacker makes you involuntarily execute client-side code, most likely JavaScript. A typical XSS attack would be trying to inject a script into input fields, URL. For example, www.mq.edu.au/?search<script>alert(“Vulnerable”);<scirpt/>.

3.2.10. Cross-Site Request Forgery

CSRF is a type of malicious exploit on a website which allows unauthorised commands to be submitted using the trust that a website has with a user's browser using specifically crafted tags, forms, and JavaScript requests.

Our application will need to protect against CSRF attacks as they could be used to possibly gain unauthorised internal access to data transportation sources and use the gathered data and information in a negative manner.

3.2.11. Stolen Google API Key

Since Google services are being utilized, it is required to use the provided API key so that the application can authenticate to Google that the application is legitimate. The application needs this API key to work. However, this API key cannot be stored in the actual code itself.

It must be stored in an environmental variable. If it's stored in the code, when pushing commits to version control services, we are vulnerable to attacker taking the API key and using it for their own purposes.

Furthermore, if they have access to our API Key, they can use these keys to build web applications that are malicious. For example, stealing user data while pretending to be our web application. This kind of vulnerability is serious, and the API key need to be stored securely. For Defence Countermeasures for this vulnerability, please go to section 2.3.

3.2.12. Session Hijacking

Session Hijacking is another common web application vulnerability. This occurs when an attacker obtains the sessionID from the client and sends that sessonID to the server to get the session token for that user. The attacker can steal your sessionID in various ways including via phishing emails, XSS, brute forcing and if your website is not TLS/SSL encrypted, they can sniff your connection using a packet sniffing/capture tool.

3.3. Defence countermeasures

3.3.1. Keeping the API Key Secure

The best practice to store your API key is in your environmental variables in your cloud service when you host them. This will ensure that no one has access to these keys except for developers and trusted administrators.

3.3.2. Guarding Against CSRF

SameSite cookie attribute

A "SameSite" attribute can be included in the server cookies to indicate to the browser that it should ignore cookies being accessed that are not related to the application's website.

Cookie-to-header token

The Cookie-to-header countermeasure is a JavaScript solution to CSRF which creates a random, unique but unpredictable token cookie which is copied into a custom HTTP header. This cookie only lasts as long as the web session persists. This technique relies on the assumption that only client-side JavaScript that set the initial cookie will be able to read its actual value, even if malicious JavaScript were to try and read/copy the cookie.

Double Submit Cookie

Like the Cookie-to-header countermeasure; Double Submit Cookie works similarly but without the use of JavaScript. When a user authenticates on a site, a CSRF token is created which requires a hidden form value in every request. [5]

Client-side safeguards

There exist browser extensions that can help mitigate and prevent CSRF attacks by blocking cross-site requests, distinguishing between trusted and untrusted sites, or removing payloads from requests altogether, to list a few methods. This type of countermeasure against CSRFs can significantly interfere with normal operations of websites. [6]

3.3.3. Guarding Against XSS

Encoding Data on Output

Encode output data for special characters that can be used as part of XSS. For example, For HTML: < converts to: &lt; and For JavaScript: < converts to: \u003c;

Validate Input Upon Arrival

After encoding, we would also validate input at the point when we receive input from the user. For example, when we expect a field to be a number from a user, we can validate that input to make sure it is an integer. Also, when we know a type of input that we want our user to fill in, we can also validate that input to be that specific type.

Prevention using JavaScript

We can filter through user input with the help of JavaScript. We can write our own JavaScript function to filter through input from the user. For example, for HTML forms, we can have a htmlEncoder function that takes every character from an input field that we think is important and encode that to store in our database. We can do the same for JavaScript.

3.3.4. Guarding Against SQL Injection

Parameterized Queries

The most important step we can take to prevent our site from attack like SQL injections is to prepare SQL statements known as Parameterized Queries. Instead of concatenating user input at the end of queries.

We can use a feature called PreparedStatement. Using PreparedStatement will ensure that every user input will not be directly concatenated at the end of our statement instead it will be replaced by “?”.

3.3.5. Guarding Against Session Hijacking

First and most important point is to use TLS/SSL encryption on the application website. The server should only accept cookie from sites that are HTTPS secured. We can also use long and randomly generated sessionID. By doing this we can make sure that the attacker cannot just brute force to find the user sessionID. Finally, the server should only generate the sessionID after the user has logged in. This prevents session fixation because the session ID will be changed after the user logs in.

3.3.6. Defences measures take for DDoS Attacks

Black-holing and Sink-holing

We can use this approach to block all traffic and divert it to a black hole (null route), where it will be discarded. This will make sure that any large-scale requests that make it to the server in a very short period will be discarded.

Routers and firewalls

We can configure our router to stop any simple ping attacks by filter out the non-essential protocols. By doing this we can also stop any invalid IP addresses. Firewalls can also be used to shut down network traffic flow that is associated with an attack.

Intrusion-detection systems

We can use IDS system to detect any strange network activities. The system will then send any alert when it finds any anomaly activities in our network.

DDoS mitigation appliances

We can also use third-party appliances that are built to stop DDoS attack.

3.3.7. Guarding Data at Rest

Apply authentication that manages access control to the application. If possible then use multi factor authentication.

Store credentials in a secure database.

Use strong encryption such as advanced encryption standard (AES) on all data to ensure confidentiality in the CIA triad.

Keep redundancy backups to prevent data loss in the event of ransomware attacks.

Use strong hashing algorithms such as SHA-512, RIPEDMD-320, Whirlpool to encrypt the user password in the database. This will ensure the integrity in CIA triad.

Apply proper permission/authorisation to each file, folder, database, or any data source. This will ensure availability in CIA triad.

3.3.8. Defensive measures for data in transit

As discussed in 2.2, encryption plays a big role for data in transit. When two parties are communicating with each other a strong cryptographic encryption protocol needs to be used to encrypt that data so that attackers cannot decrypt the cipher text.

As mentioned in 2.1 and 2.2, we should consider data communication channels as always being unsecure over the internet and that an eavesdropper is always intercepting or listening to web traffic. If the encryption is strong then an eavesdropper or man in the middle will not be able decrypt and make use of that data.

To ensure confidentiality and integrity of our data communication, the new version of TLS 1.3 will be implemented.

SSL/TLS Version 1.3 also protects against all known vulnerabilities from SSL/TLS Version 1.2 such as ROBOT, POODLE, BEAST, CRIME, BREACH, HEARTBLEED, 3SHAKE, and Raccoon. [7]

3.4. Architecture, Algorithms and Models

3.4.1. Application architecture

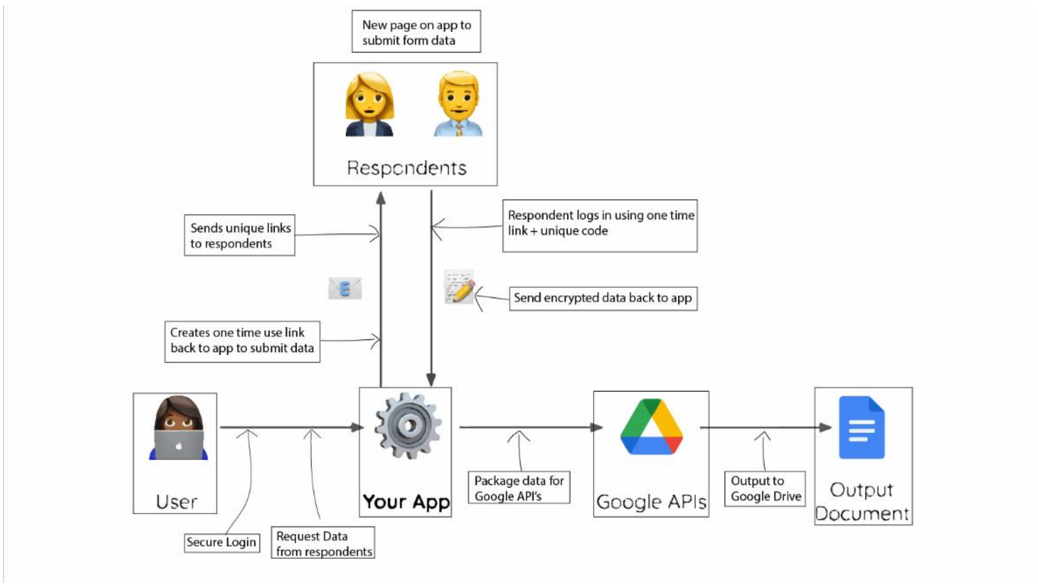
New page on app to submit form data
Respondents
Sends unique links to respondents
Respondent logs in using one time link + unique code
Send encrypted data back to app
Creates one time use link back to app to submit data
User
Your App
Google APIs
Output Document
Package data for Google API's
Output to Google Drive
Secure Login
Request Data from respondents

Figure 4: Application architecture

The architecture of the application begins when the user securely logs in using Google authentication, this consequently allows access to requesting data from the respondents triggering a request for a one time use link.

This link system is implemented to redirect each respondent to a page within the app that contains a form to input data.

Links are vital to the security algorithm as they are attached with a unique code and sent to another medium allowing for the model to prevent unauthorised log-in.

Once completed the form data is sent securely back to the application where each respondents data is packaged up and forwarded to the Google Drive and subsequent google API's. The data is then output as a single google document owned by the initial user of the application.

3.4.2. SSL/TLS detection method

In TLS protocol, data is firstly client and server authenticate each other followed by an exchange of cipher suite key. Once that is done then the data gets encrypted.

Before sending the data, the sender signs the data with a message authentication key (MAC) and hashes that message with a hashing algorithm. The receiver then decrypts the message with the sender's private key and the receiver compares with previous value.

If the two values match, then the receiver takes that as data as legitimate. If they do not match, then it’s evidence of the data received having been tampered with.

3.4.3. Web application scanner

A web application vulnerability scanner is an automated tool that scan the application against database of known vulnerabilities and misconfigurations. The scanner can look for common flaws in our application such as SQL injection, cross site scripting, remote code execution, path traversal, and insecure direct object reference.

As per recommendations of centre for information security, a weekly scan will be conducted on the web application to identify vulnerability.

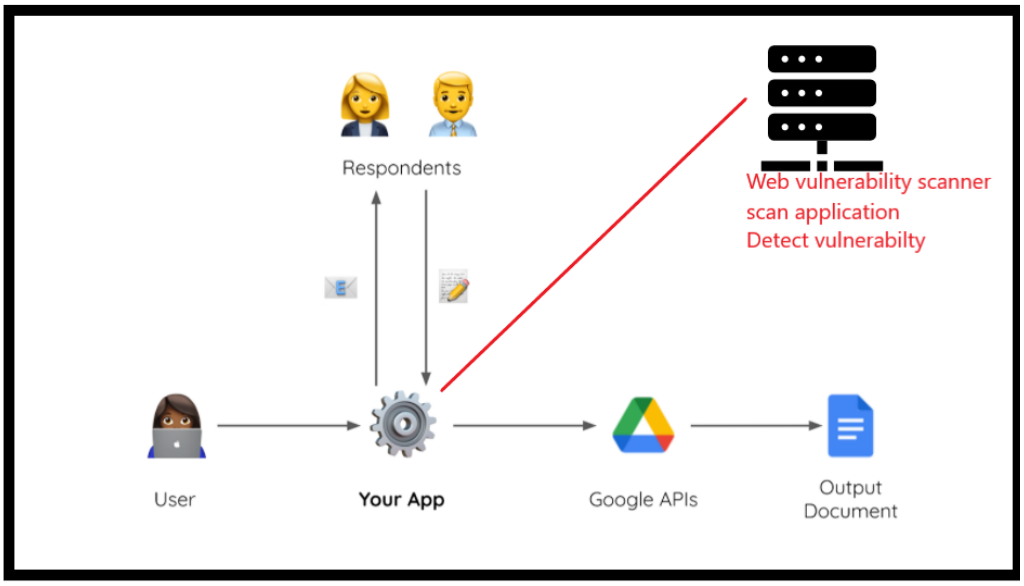

Figure 5: Web application scanner

3.4.4. Website monitoring software

As mentioned in section 1.4, an external software product will be used for web application monitoring. This software can monitor websites, web applications, application performance, identify critical problems, recommend proper resource utilisation as well as application availability, URL monitoring, content monitoring, HTTP status, session hijacking detection, website hijacking detection, SSL certificate monitoring and much more.

3.4.5. Security information event management (SIEM)

A SIEM solution such as SPLUNK could be implemented to collects logs for real time and continuous monitoring and analyse these logs for threat detection purposes. SIEM product can collect any types of data and allows to set an alert based on user's preference.

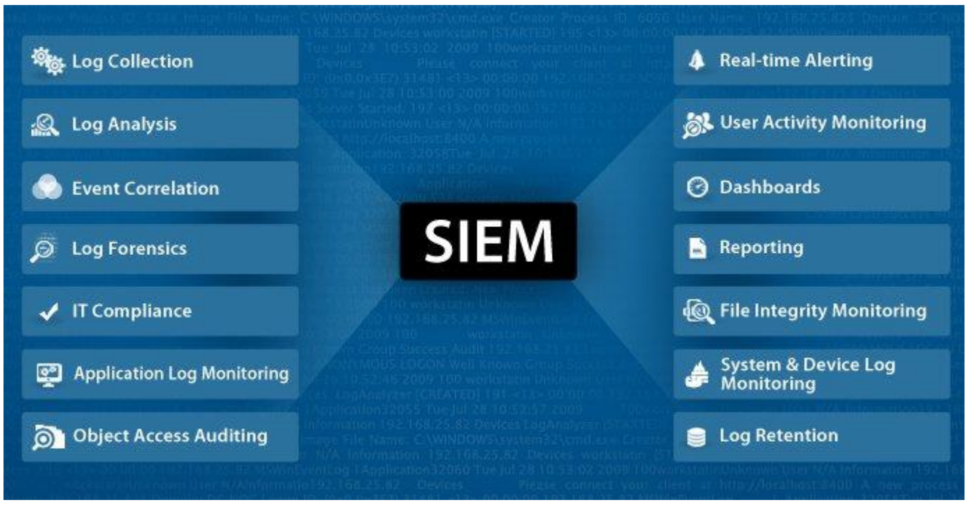

Figure 6: SIEM capabilities, source: ManageEngine

For example, we could set an alert for SQL DELETE statements on the database, if an attacker tries to delete any tables or columns in our database, then it will create an alert for us and based on that alerts we could prevent any specific incidents.

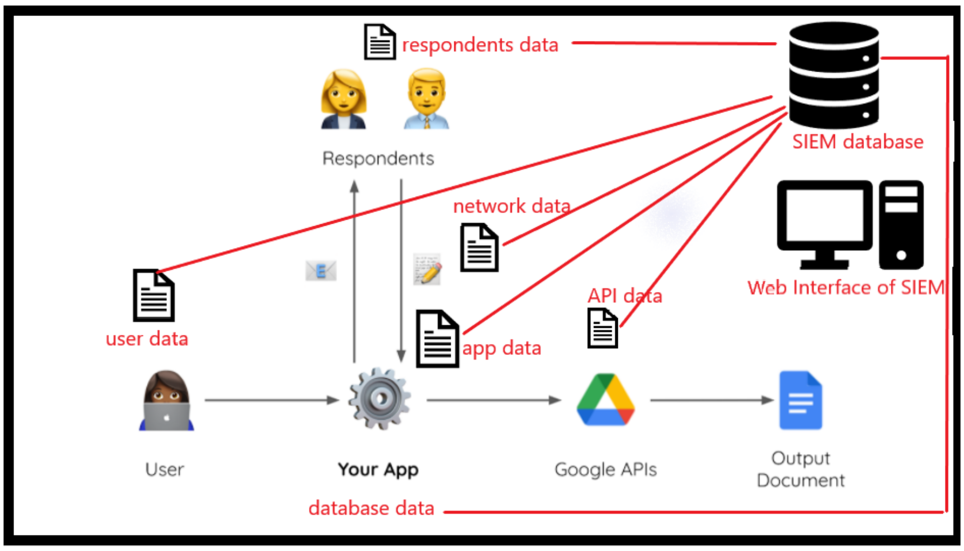

Figure 7: SIEM implementation and collection of logs

3.5. Security Testing (e.g., penetration testing)

For security testing, a generic software development life cycle (SDLC) model will be used. This generic model has five phases such as before development begins, during definition and design, during development, during deployment, and during maintenance and operations. [8]

3.5.1. Before development begins

In this phase an acceptable software development life cycle needs to be defined. All related policies need to be reviewed and any development and security standards documentation needs to be in place. A measurement and metrics criteria to ensure traceability need to be defined and created.

3.5.2. During definition and design

In this phase security requirements need to be defined and reviewed based on how the application works. Other phases may include design and architecture reviews, the creation and review of UML models, and the creation and review of threat models.

3.5.3. During development

Once development reaches its latter stages with enough functionality, the security teams can begin preliminary penetration and security testing to identify any flaws in development. This information can then be used to fix vulnerabilities and guide future security development as developers will have more of an idea of how to develop security-conscious functionality for the application.

3.5.4. During deployment

Once the application has been developed, final penetration tests need to be performed to ensure that the deployed application does not have any zero-day exploits. The security team should examine the infrastructure being used to deploy the application as well as its configuration management to ensure no weaknesses exist in those areas.

3.5.5. During maintenance and operation

In this phase a process needs to be created outlining how the operational side of application and its infrastructure will be managed. Periodic heath checks will be performed to ensure there are no new security risks. A change verification process will be in place to ensure any changes made to the application does not affect security.

3.6. Deployment – monitoring, reporting, review, training/support, security polices)

3.6.1. Deployment

As mentioned in section 1.5, we assumed that our web application will be deployed to AWS service called AWS lambda. This is a serverless service that is provided by AWS, so all we must care about is the application code that we provide them.

3.6.2. Monitoring

We have mentioned in section 1.4, that monitoring the web application can be done using a third-party application. The monitoring software that we are planning to use is called SPLUNK. It will collect any logs from the server in real time and continuous monitoring and continuous analyses. If we want to detect any anomaly activities in our server, we can also write our own filter, to filter out any suspicious activity and SPLUNK will alert us when it detects such activity.

We will also be monitoring the web traffic volume and page load times. By doing this we can dynamically scale our web server to handle as many traffic as we can.

3.6.3. Reporting

We are reporting all the processes on how we are deploying the web application and how we are planning to implement the feature and concept that our team and client have discussed about.

The review from the client to us is done weekly through our weekly meeting to see if we are on track of what the client wants from us.

3.6.4. Training/Support

Details and instructions on the workings of the system and how to set up the environment it runs in will be written up as development takes place. Internal code comments and a readme file will enable future developers to understand how the code works to be able to use and build off it in future.

Furthermore, the sponsor will be demonstrated the functionality and workings of the project. These demonstrations will be done for prototypes as well as the final product that is being delivered to the sponsor. This will enable the sponsor to understand the workings of the application, both in it is use and internal workings. From there, the sponsor will then be capable of providing instructional material or training any new developers or users in the use and/or workings of the application.

3.6.5. Security Policies

In reference to section 2.4 and 2.5, our security protocols and testing is thoroughly explained. There are security policies which we implement to strengthen these algorithms and models. Policies utilised are as follows:

Access Control Policy (ACP)

- ACP is the consideration of the access allocated to each employee and user; it is essential to secure a software from all users. Some data and information which is unrequired by an employee or security information which is best to be unauthorised from employees is a result of this Policy. It introduces access control standards and user access standards.

Acceptable Use Policy (AUP)

- AUP enforces constraint and privacy within the system and those who use it. Users of the system, for example new employees must agree to the IT assets and their protection code of conduct before they are given access to the network.

4. Customer Feedback

Date of Review by Sponsor: 30th of March 2021

Feedback Received:

- Grammatical and structural corrections (capitalizing entities such as ‘Google’ and re-wording to not imply direct collaboration with companies in the project) to make the document look more cohesive and professional.
- Rewrite of 1.1. Scope and Purpose section of the document as it did not differentiate between the ‘why’ the project was being done and the ‘how’ it was going to be accomplished.
- Addition of some figures to clarify points being made.

- More information in section 2.5 Security Testing and cutting back a small amount on section 2.3 Defence Countermeasures.

Actions Taken: All members of the group will look at the feedback and update their sections to include the feedback given. The document manager will do final editing and formatting to ensure the document is of a consistent and professional looking format.

5. References

[1] L.Constantin, "What are vulnerability scanners and how do they work?", *CSO Online*, 2021. [Online]. Available: https://www.csoonline.com/article/3537230/what-are-vulnerability-scanners-and-how-do-they-work.html#:~:text=Vulnerability%20scanner%20definition,could%20expose%20them%20to%20attacks. [Accessed: 31- Mar- 2021].

[2] "Web Application Monitoring Software - Nagios", *Nagios*, 2021. [Online]. Available: https://www.nagios.com/solutions/web-application-monitoring/. [Accessed: 31- Mar- 2021].

[3] Burns, E. and Pratt, M., 2020. *What is Data Preparation?*. [online] SearchBusinessAnalytics. Available at: <https://searchbusinessanalytics.techtarget.com/definition/data-preparation> [Accessed 1 April 2021].

[4] 2021. [Online]. Available: https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/. [Accessed: 31- Mar- 2021].

[5] "Double Submit Cookie Pattern", *Medium*, 2021. [Online]. Available: https://medium.com/cross-site-request-forgery-csrf/double-submit-cookie-pattern-65bb71d80d9f. [Accessed: 31- Mar- 2021].

[6] "Cross-site request forgery - Wikipedia", *En.wikipedia.org*, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Cross-site_request_forgery#Synchronizer_token_pattern. [Accessed: 31- Mar- 2021].

[7] 2021. [Online]. Available: https://www.cloudinsidr.com/content/known-attack-vectors-against-tls-implementation-vulnerabilities/. [Accessed: 31- Mar- 2021].

[8] "WSTG - v4.2 | OWASP", *Owasp.org*, 2021. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/v42/3-The_OWASP_Testing_Framework/0-The_Web_Security_Testing_Framework.html#A-Typical-SDLC-Testing-Workflow. [Accessed: 31- Mar- 2021].