Portant - Protecting Sensitive Data from Multiple Sources

Design and Testing Document

COMP3850 – Computing Industry Project

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

Contents

1. Table of Revisions

Version	Publish Date	Change Description/Features
1.01	18/05/2021	Updated the Prototype section to reflect the changes to the prototype since the last deliverable.
1.02	18/05/2021	Updated Security/Development Assumptions section and Added SLA requirements.
1.03	19/05/2021	Deleted black box testing section in Security Test Plan.
1.04	19/05/2021	Updated data preparation to include user data and volume and pattern data for maintenance
1.05	19/05/2021	User errors, user impersonation and client side security protocol, added to Detection of Intrusions/ Vulnerabilities and Defending Solutions Architecture sections.
1.06	20/05/2021	Final edits of document before submission

2. User Stories for Security/Privacy Attacks

There are a few user stories that detail the interactions that users will have with the systems and the security measures that users will be expecting while using the system. As a note, when referring to a 'user' and 'respondent' they are one in the same since both can log in to the application and request/provide information. A user in this case refers to someone requesting information and a respondent is someone providing information in response to a user's request.

1. As a user, I want to log in to the system to authenticate my identity for use in creating and viewing forms.
    a. This covers the general use of the application. The authentication system will allow for users to be verified when they first log on to the application. Any forms answered or created by this point can be verified as having come from the authenticated user. This ensures that information being sent around can be traced back to a verifiable source.
    b. Considerations for security include replay attacks and masquerade attacks, whereby an attacker can utilise a user's credentials to pose as that user and perform tasks on their behalf.
2. As a user, I want to create and send a secure form that can only be seen by specified respondents for the collection of information.
    a. This covers the creation of forms for information collection. The intent is that users will be able to create forms that can then be guaranteed to be secure and accessible only by the people they specify.
    b. Encryption of the form being sent to the respondent aims to remove the ability for eavesdroppers to access and and/or modify data in a Man-in-the-Middle (MitM) attack.
3. As a respondent, I want to authenticate myself before accessing a form to ensure that the information I provide can be attributed to me and not an unknown 3rd party.
    a. When a respondent logs into the application, they will already have gone through the authentication process to verify that they are who they say they are. However, for each individual form that a respondent accesses, there is an added layer of authentication to ensure that the respondent in question accessed and responded to that form.
    b. Sensitive information must be restricted to the intended recipient only. Verification of the respondent is necessary to ensure this standard is upheld.
4. As a respondent I want to securely provide responses to back to the sender.
    a. When a respondent answers a form, the information provided will need to be encrypted to ensure that if the information is accessed while in storage or intercepted in transit it cannot be read by anyone other than the sender and intended recipient.
    b. This is inherently in the interest of security, to prevent data theft and MitM attacks. For the server, data sanitisation can also be employed to prevent cross-site scripting attacks, whereby malicious code can be inserted into text input areas.
5. As a user, I want to view collected information from a form in a presentable format in Google Docs.

a. Much like when information is first provided, when information moves from the application into a Google document, that information will be encrypted in transit to ensure the resulting Google Doc cannot be accessed by unauthorised people.

3. Security/Development Assumptions ///SLA///

3.1. Security

It is assumed that Google’s G-Suite (Google Docs, Google Authentication, GMail and Google’s API) are equally or more secure than our developed application. We use these systems as a part of the application, and we have no control over how their security is implemented.

It is also assumed that any cloud services that we may use are as secure or more so than the developed application.

We also assume that users authenticating with Google Auth can be responsible for ensuring they keep their normal Google account credentials secure so they cannot be stolen and used to impersonate the user.

Since we cannot control the security of 3rd party services and the user’s ability to keep their own credentials to other services secure these assumptions need to be made for development.

It is assumed that the systems use and follow these guidelines which are provided to Google’s G-Suite.

For Security it is essential to take Service Level Agreements (SLA) into account. SLAs are key for vendor contact; it establishes the level of security or service which is required or expected by the user. It is key that we use these as a guideline in security to make sure the guidelines provided are met. Knowing the data that is being stored and its use will enable us to make informed decisions about measures that we need to take in order to keep our application and the data that moves through it secure.

3.2. Development

Since we are using the Google's G-Suite for our user login, we assume that the single sign-on system from Google works as it is expected and can provide the information that is required such as a user’s email.

As discussed in section 3.1 SLAs are key when outsourcing, these requirements also apply to the development of the system as it is required that all metrics and expectations from third-party vendors are known. Ensuring that we know what level of service and coverage to expect from third-party services and if there is any information that needs to be included in our own SLA to users of our application.

4. Intrusion and Vulnerability Detection

4.1. Intrusion Detection Methods

There are two types for intrusion detection:

- Static
- Dynamic

Within these opposing types, the two different methods of intrusion detection involve:

- Network intrusion detection system
- Host-based intrusion detection system.

4.1.1. Static methods

Static intrusion detection generally occurs after the incidents. This is done by parsing the log file using different tools or software. As mentioned in our scoping document, a SIEM solution software will be collecting all the logs from our application and web server. Using security event information management (SIEM) we could analyse the logs for any intrusion.

If we suspect any intrusion in our application or any anomaly in our applications behaviour or network which happened already then, we will look at data source like

- Web servers log
- Applications server log
- Operating systems log
- Any custom audit trails in our application.

Looking at these logs would help us identify if any attack happened such as SQL injection, cross site scripting, directory traversal, remote code execution, website scanning, brute force attack etc.

4.1.2. Dynamic methods

On the other hand, dynamic methods are used for during intrusions of the system. It triggers the alarm based on our policy or rules when the attack happens. Dynamic intrusion detection method is used to detect real time attack or anomaly and that could be used to prevent the incident.

Techniques used for dynamic detection methods are signature based and anomaly based. These two techniques have its advantage and disadvantages, so it is recommended to use both techniques. Application firewall, in-line application intrusion detection system, network intrusion detection system used for dynamic method. SIEM is also used for dynamic method as we collect real time logs, and we can set alarms and if any intrusion happened that trigger the alarms.

Most modern or next generation firewall has signature for detecting web application attacks in real time and we can always write our own signature or rules based on our criteria.

4.1.3. Web Application Firewall (WAF)

A web application firewall is used for protecting an application by monitoring and filtering the traffic from internal and external network. It is generally used to protect the web application for attacks such as cross site scripting, cross site request forgery, remote and local file inclusion, remote code execution, SQL injection etc. It does work in application layer of OSI model, and it is designed to detect, monitor, and defend against any attacks in the application layers.

A policy or rules can be set in the WAF to filter and monitor web application traffic. It is one of the best ways to defend against distributed denial of service as it can work as a reverse proxy.

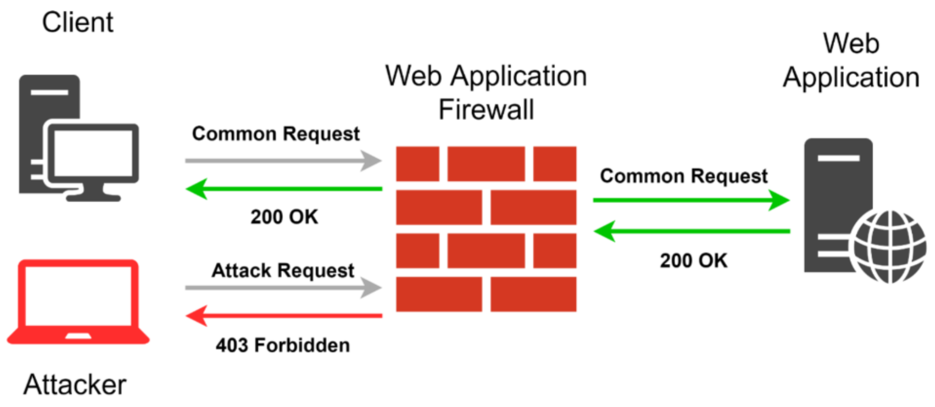


Figure 1: WAF visual from I. Bulatov, D. Rybin, and A. Romanov, habr. Digital Security, 2019

4.1.4. Network-based Intrusion detection system (NIDS)

A network-based intrusion detection system can monitor and detect malicious traffic on a network. The detection method can be set to both signature-based and anomaly based. It can detect port scanning, brute force, malicious traffic towards web application, distributed denial of service and other different attacks.

4.1.5. User impersonation

If a user's credentials are stolen, an attacker can impersonate them and access the application using their credentials. The only thing that can then set the legitimate user apart from an imposter at that point would be the device of access (differences in MAC and IP addresses used to access the application). Much like what many companies will do with account access, if a foreign IP address is detected trying to access a user's account, it will prevent access and require the user to authenticate the access via email or SMS. A similar system can be put in place to detect intrusion via impersonation.

4.2. Vulnerability Detection Methods

4.2.1. Web application vulnerability 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.

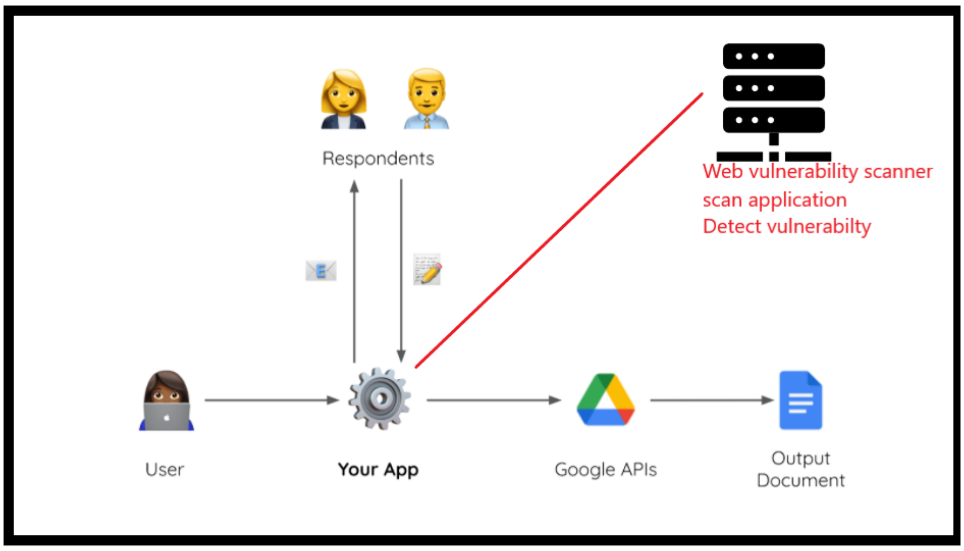


Figure 2: Web application vulnerability scanner

A generic vulnerability scanner which can scan web application, network, system and can find weakness that could expose them to attacks. A combination of both will be good option to detect vulnerabilities.

4.2.2. **Penetration testing**

A penetration testing is a simulated attack against a network, system, operating system, physical security, web application etc to test if they are vulnerable to any attacks or exploits. This is also done to test if the web application firewall or network-based intrusion detection system functioning as expected.

These tests can be done internally and externally. A penetration test and vulnerability scanning complement each other to ensure that all exploits are found and can be reported.

Penetration testing has different stages, and it will be discussed in detail in Section 6.

4.2.3. **User errors**

Over the shoulder surfing and public network usage can lead to vulnerabilities in circumventing authentication. Defending such vulnerabilities can be managed through IP packet tracing, however this requires a database on usage volumes. User training would be required for managing workplace practices such as policy on leaving devices unattended and maintaining a clear desk policy. Workplace training on phishing detection will be necessary to limit malicious software affecting the web application.

5. Defending Solutions Architecture

5.1. Architecture of the System

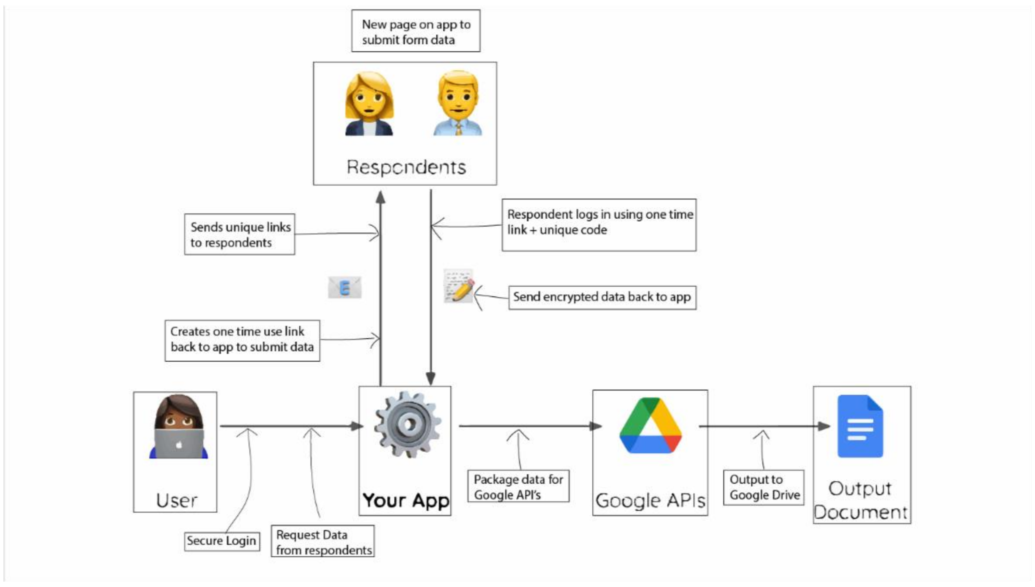

New page on app to submit form data
Respondents
Sends unique links to respondents
Respondent logs in using one time link + unique code
Send encrypted data back to app
Creates one time use link back to app to submit data
User
Your App
Google APIs
Output Document
Package data for Google API's
Output to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

Figure 3: Architecture of the System

The system consists of a front-facing web application with a server backend that links to a database for the storage of user and form information. The application has links to a Google API that enables the application to interact with Google's services such as Drive and Gmail.

The front-end web application utilizes JavaScript with a Python backend utilizing the MongoDB interface for internal database creation and storage. The application itself will be run through a web browser that can either be hosted locally on a machine utilizing Apache HTTP or on a remote hosting service such as Microsoft Azure or Amazon's AWS.

The first interaction that a user will have with the application is to authenticate themselves using their Google account which essentially acts as a login for a user. This system is used for both existing users to access their profile and new users for which their profile will be created which will allow them to use the application.

A user's 'profile' consists of the forms that they have created, forms that they have been given access to and their email, as provided through Google Authentication.

A user can create a form using the form builder provided by the application. This will create a form on the app and send it to the database for storage. The link to the form will be provided to the user creating the form to send to people they wish to answer the form.

A user can also request that information given via responses to forms be formatted and written to a Google Drive document for easier viewing. This is handled externally using the Google API.

When a person receives a link to a form, they will be first directed to enter a one-time use code that will give them access to the form, this code is sent via email to people that a user designates as being a respondent to a form. After entering the code, the respondent gains access to the form. After filling in the required information, the form will be sent back to the sever for the info to be stored waiting for the requesting user to access it.

5.2. Mechanisms / Process flow / Defences

5.2.1. Mechanisms

There are several security mechanisms in place to ensure that users that use the application and the information that moves throughout the application are secure. These are as follows:

- Cryptographic schemes – Things such as RSA-1024.
- Message digests – Hashes of information like passwords that should only be known by a user and nobody else.
- Digital certificates – Certifies that the application is secure and can transmit encrypted information as well as monitor the flow of encrypted transmissions.
- Digital signatures – Verifies forms being sent to ensure that they have been sent from the expected sender.
- Public Key Infrastructure (PKI) – Public-Private key pairs allow for asynchronous encryption of information using schemes such as RSA. The generation of such keys allows for the encryption keys to constantly be changed, adding entropy and variance to cryptographic schemes.
- Authentication Exchange - Via the use of Google Auth to verify yourself as a user before entering the application as well as verification codes to forms to only allow for authorized people to view forms.
- Transport Layer Security – Enables communications between the application and clients to be secure from interception and tampering.
- Client-side Security Protocols – Things such as auto log-out after a certain period of inactivity and a wrong password lockout if too many incorrect password attempts are logged.

These security measures will ensure that the application fulfils the AAA triad: Authorization, Authentication and Accountability. All three of these things ensure that the application maintains its security and integrity for its users and their information.

5.2.2. Process Flow

As mentioned in our scoping document, a generic software development lifecycle will be used for security testing. The process flow of this testing method includes requirements, design, coding and unit testing, integration testing, system testing, implementation, and support.

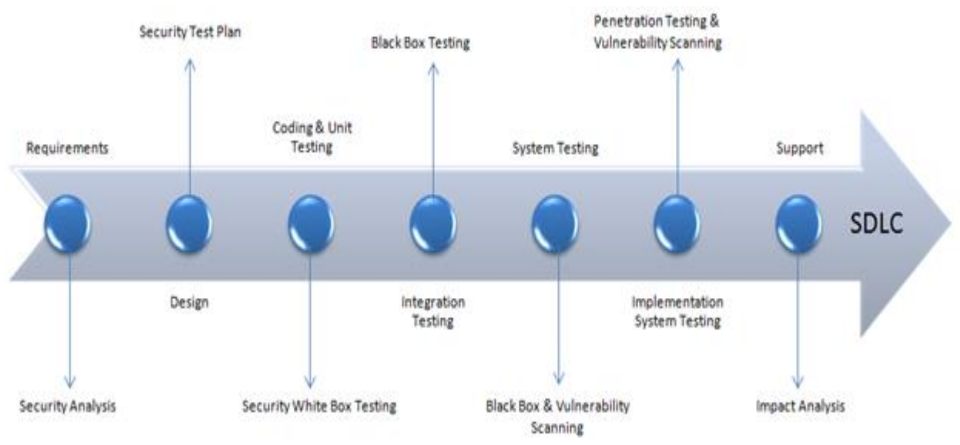


Figure 4: Process flow of security testing, Image source: guru99

Security analyses require in the beginning of development. It is also involve checking any misuse or abuse the application. Then for the design, a risk analysis needs to perform. For coding, unit testing and integrating testing, white box and black box testing need to be performed. Once that done, then test the system and web application with web application vulnerability scanner. As for implementation and support, identify any vulnerability and conduct impact of analysis of patches.

5.2.3. **Defences**

Securing the Application

Always better to secure the application development such as input validation, sanitization, anti XSS HTML filtering.

As mentioned in our deliverable 2, we will be using a lot of technique to prevent malicious users form tempering with our web application.

Since we are using our users' Gmail account, we will be getting a Google API Token, to access our user's Google services. These API token should be secure to prevent attackers from tempering with the users' information. In development environment, we will be storing this in a .env file for easy access to development. However, when the web application is in production, we will be storing these API Token in a secure environmental variable in the web server.

We will also protect our web application against other type of attackers such as CSRF, XSS, SQL injection, Cookies Hijacking and many more. The details of how we are implementing this has been listed on our deliverable 2. However, to summarise on how we implement this would be that we are sanitising all user input from the form that the respondent sends back to our web server.

Apache Module Mod_security

Since we will be hosting our web application on Apache Server, we can use a module called mod_security to protect our web applications. It works with Apache and can scan in-depth and fine gained checks. It can scan cookies and session id too. It can also protect a web application from various attacks and can block commonly known exploits.

Intrusion Prevention System

An Intrusion prevention system (IPS) is a network security system which can detect threat and protect the network and different services from that threat. It can examine network traffic and can block any malicious activity or threat that attempt to harm any application or services running inside the network. Generally, an IPS has the capability of detection and prevention. It can detect and prevent from any know vulnerabilities and from behaviour-based threat. These are called signature-based detection and anomaly-based detection. It can allow or deny traffic based on policy, source IP, destination IP, protocol, port number etc.

Web application firewall (WAF)

A web application firewall is used for protecting an application by monitoring and filtering the traffic from internal and external network. It is generally used to protect the web application for attacks such as cross site scripting, cross site request forgery, remote and local file inclusion, remote code execution, SQL injection etc.

It is also called an application firewall. It can analyse, monitor, detect and protect malicious HTTP traffic. It can also protect from distributed denial of service. It works as shield between application and internet. It works based on a blocklists too which protects against known attacks. A network or cloud based WAF could be implemented to protect our web application.

Security Event Information Management (SIEM)

A SIEM is a software solution which can monitor our web application in real time. It can be used to collect real time log and user could set an alarm for any abnormal or malicious behaviour. It can provide a holistic view of what's happening in our web application and using that information we could protect or prevent that security threats.

Client-side Security Protocols

While there are several systems that can be put in place to defend from external attacks, the most easily exploitable parts of a system are its users. Client-side protocols such as an auto logout after 2 minutes of inactivity can help in ensuring that users who leave their devices unattended while logged in have a smaller window to be exploited for impersonation purposes.

6. Pen Testing Approaches

The approach taken for penetration testing follows the Cyber Kill Chain by Lockheed Martin. The stages of the chain as described by Lockheed Martin will be used as the methodology for penetration testing of the application. There are 7 stages of the chain that will be run through, these are as follows:

1. **Reconnaissance**
    a. In this stage the system will be combed over to gather information about the application, systems, and its users. The information gathering will involve active and passive reconnaissance of the application’s form creator, form response and Google API interaction as well as how users interact with these systems.
2. **Weaponization**
    a. This stage will involve exploiting found vulnerabilities to create a backdoor into the system, where which malicious payloads can be delivered to the system. Depending on where exploits are found in the system, different kinds of attacks can be employed to gain a backdoor into the system.

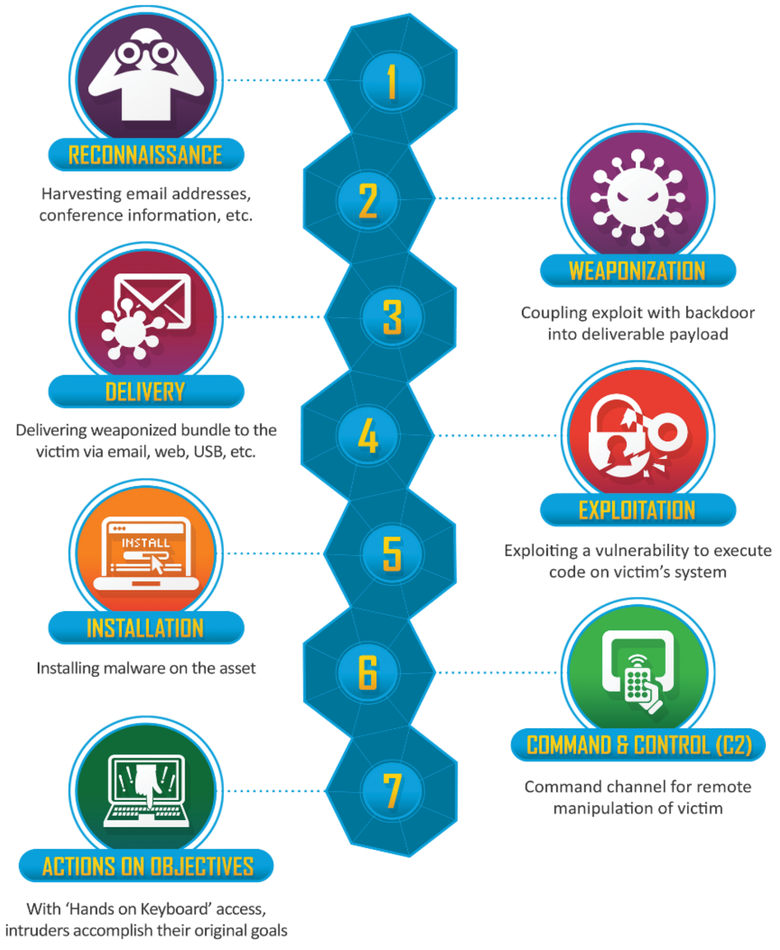

1
RECONNAISSANCE
Harvesting email addresses, conference information, etc.
2
WEAPONIZATION
Coupling exploit with backdoor into deliverable payload
3
DELIVERY
Delivering weaponized bundle to the victim via email, web, USB, etc.
4
EXPLOITATION
Exploiting a vulnerability to execute code on victim's system
5
INSTALLATION
Installing malware on the asset
6
COMMAND & CONTROL (C2)
Command channel for remote manipulation of victim
7
ACTIONS ON OBJECTIVES
With 'Hands on Keyboard' access, intruders accomplish their original goals

Figure 5: Cyber kill chain, Image source: Lockheed Martin

3. **Delivery**
    a. In this stage, we will transmit the exploit or weapon to the target via the created backdoor. This can be done using kali Linux, email attachments, stored scripts on the website or USB injection.
4. **Exploitation**
    a. In this phase, an exploit will be used to execute a weapon that has been injected or placed into the system, this weapon can then be used to install malware that can do things such as capture transmissions and submit forms while imitating legitimate communications.
5. **Installation**
    a. In this phase any malware placed on the system can start to be installed and prepared to take control over the application and its functions.
6. **Command & Control**
    a. In this stage our exploit will gives us access to the system from a remote point of access, at this point it is no longer needed to use backdoors to manipulate the application.
7. **Action on Objectives**

a. With persistent access to the system it’s then possible to carry out operations such as data access, data manipulation, data destruction and installing of further malware to accomplish the goal of obtaining as much as possible.

7. Data Description

7.1. Data Collection

Data being collected, generated, and stored:

- Forms that have been created with an associated creator and respondents.
- User details (email) that are collected for respondent.
- User’s names for linking to responded forms.
- Forms that have been replied to (this will be replaced by sending info to Google Docs via API)
- Private keys to associated public keys and forms used for encryption.

Data that is being collected and stored will be included in a SLA and EULA that is given to users when they first sign up to the service. In these documents, the data that is being collected and stored will be detailed as well as how it is used and how long it is being kept for. These documents are standard in informing users of how the information they are providing is going to be used and will enable them to give informed consent to having their data being used in the ways required by the application.

While not a part of the application at this stage, data on usage and volume of users for analytics as well as IP addresses for security purposes may be collected in future versions of the application. If this data is collected and stored, current users will need to be notified on the updates that are being made to documents such as the SLA and EULA to reflect the newly collected information.

7.2. Data Preparation

Data that is being stored statically within the system will be encrypted. This consists of user and respondent emails and their authentication tokens.

For data moving throughout the system, subsequent mechanisms are also in place to ensure security standards are being met. The raw form created by the user will be encrypted, and to ensure further verification of the respondent, access will only be granted to the form after multi-factor authentication is passed. This is decided upon due to the sensitive nature of the information being sent. After this process, the form (with data added by the respondent) will be encrypted using RSA-512. Once the server decrypts this data using its private key, the form will be transformed into a Google Docs which will be emailed back to the user. Security for these steps will rely upon the security mechanisms offered by Google as our application uses the tech company’s resources and architecture for the handling of this data.

User identifiable data such as emails and names are being stored within the system for authentication and mailing purposes. This data is not being encrypted as it would intervene with system operations and is not sensitive in nature. All information pertaining to a user is frequently purged from the database after a set period of non-usage (such as when a user unregisters from the service).

8. Prototype

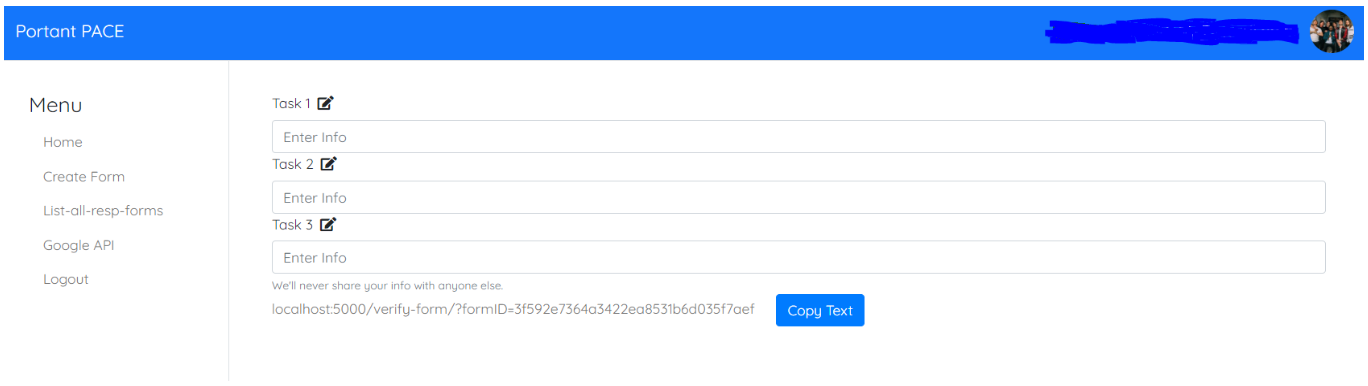
8.1. Second Iteration of Prototype

This section of the document has been updated from when it was first presented in Deliverable 3. For the sake of brevity, the details of the 1st prototype aren't being included, the details of which can be seen in the Deliverable 3 version of this document.

The second iteration of the prototype currently has the following functionality:

- Forms can be created with the option to rename fields and add new ones. When the form is completed, the user can generate a link which can be sent to the desired respondents.
- Forms can be accessed using the generated link, upon attempting to access a form, a verification code is required.
- Email addresses can be specified for forms and verification codes to be sent to upon form creation.
- A respondent can fill in the form to send back to the user who requested the information, this form will be displayed on the user's application when they log in, with field names and responses listed underneath them.
- The public and private keys for encryption can be generated, stored, and sent where they need to go for encryption.
- Information that is stored on the sever as well as being provided in form responses can be encrypted and decrypted for viewing when needed.

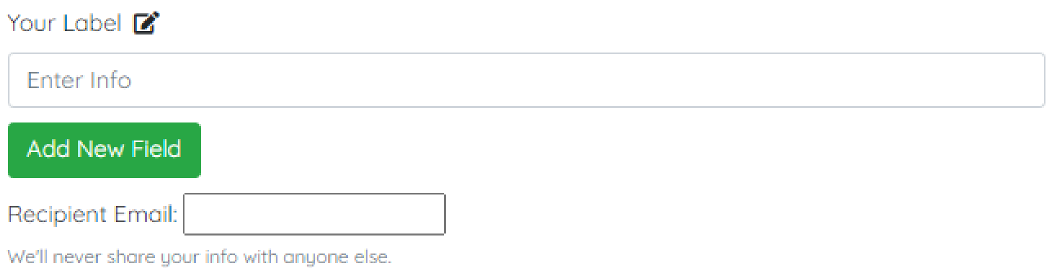
The look and feel of the application can be seen in the images below:



We'll never share your info with anyone else.

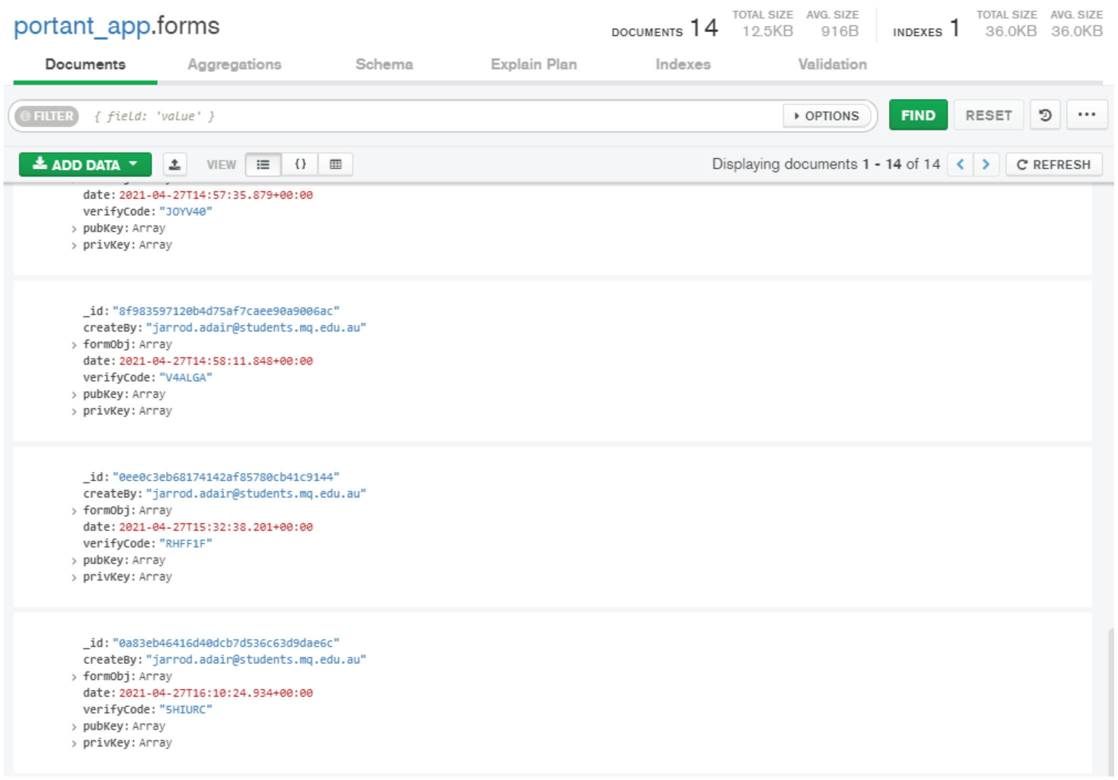
localhost:5000/verify-form/?formID=3f592e7364a3422ea8531b6d035f7aef

Figure 6: The form creation screen, users can rename fields, add new ones and generate the link to the form on the same page.



We'll never share your info with anyone else.

Figure 7: Recipient Email Specification, the link and code to the form will be sent to the emails specified in this field.



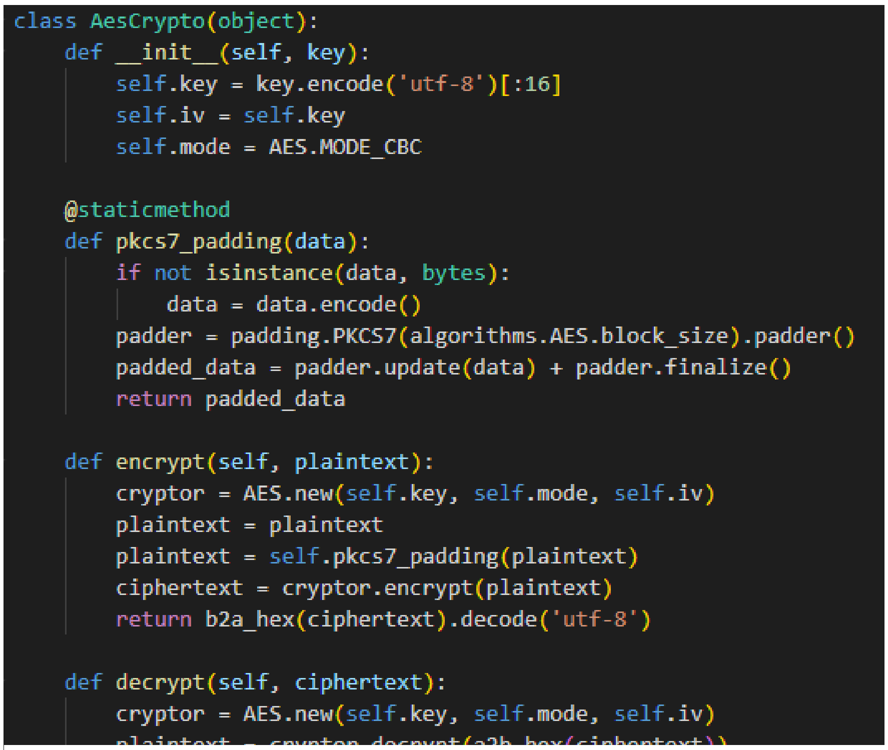
```
    date: 2021-04-27T14:57:35.879+00:00
    verifyCode: "JOYV40"
  > pubKey: Array
  > privKey: Array

    _id: "8f983597120b4d75af7caee90a9006ac"
    createBy: "jarrod.adair@students.mq.edu.au"
  > formObj: Array
    date: 2021-04-27T14:58:11.848+00:00
    verifyCode: "V4ALGA"
  > pubKey: Array
  > privKey: Array

    _id: "0ee0c3eb68174142af85780cb41c9144"
    createBy: "jarrod.adair@students.mq.edu.au"
  > formObj: Array
    date: 2021-04-27T15:32:38.201+00:00
    verifyCode: "RHFF1F"
  > pubKey: Array
  > privKey: Array

    _id: "0a83eb46416d40dcb7d536c63d9dae6c"
    createBy: "jarrod.adair@students.mq.edu.au"
  > formObj: Array
    date: 2021-04-27T16:10:24.934+00:00
    verifyCode: "5HIURC"
  > pubKey: Array
  > privKey: Array
```

Figure 8: The database entries containing forms that have been responded to. The requesting user, form fields and data, verification code and public and private keys are stored in the database when a respondent submits a form back to the application.



```python
class AesCrypto(object):
    def __init__(self, key):
        self.key = key.encode('utf-8')[:16]
        self.iv = self.key
        self.mode = AES.MODE_CBC

    @staticmethod
    def pkcs7_padding(data):
        if not isinstance(data, bytes):
            data = data.encode()
        padder = padding.PKCS7(algorithms.AES.block_size).padder()
        padded_data = padder.update(data) + padder.finalize()
        return padded_data

    def encrypt(self, plaintext):
        cryptor = AES.new(self.key, self.mode, self.iv)
        plaintext = plaintext
        plaintext = self.pkcs7_padding(plaintext)
        ciphertext = cryptor.encrypt(plaintext)
        return b2a_hex(ciphertext).decode('utf-8')

    def decrypt(self, ciphertext):
        cryptor = AES.new(self.key, self.mode, self.iv)
        plaintext = cryptor.decrypt(a2b_hex(ciphertext))
```

Figure 9: Part of the AES encryption class that forms a part of the encryption present in the application.

The features that have yet to be implemented:

- Hosting of application on a web server for remote access.
- Writing form information to Google Docs when requested by a user.
- Input sanitisation in forms for the prevention of SQL Injection and XSS attacks.
- Extensions to the form creation (specifications of data type for inputs and file/image upload for responses) with relevant security functionality to ensure no new exploits are introduced.

8.2. Demonstration of Prototype to Sponsor and Feedback

Date of Demonstration: 18th of April 2021

Systems Demonstrated:

- Form creation
- Form response
- verification systems (Google Auth and form access verification code)
- Sending of emails with form links.
- E2E encryption of form data
- Ideas about templating documents for Google API.

Feedback from Sponsor: The form creation and building were well received, the sponsor has said that the next step in improving the functionality of the form creator is to be able to specify input types (text boxes, numbers, date, etc.) as well as eventually being able to upload files for responses.

The sponsor is happy with the current system of generating verification codes on a per form basis for identity verification. They were also pleased with the ability for form links as well as verification codes to be emailed to specified email addresses.

The sponsor was also happy with the E2E encryption measures currently implemented to protect data both in storage and when its being sent to the application.

The main idea for the Google Docs functionality is to have selectable response forms that will then be used to make a combined document containing headings corresponding to the form field names. For example, if 3 different people respond to the same form, the forms will come back as 3 different forms that can be selected. Since the 3 people responded to the same form, selecting all 3 forms will result in the information in each form field all appearing under headings that correspond to the field headings in a single Google Doc.

Actions Taken: With the sponsor's approval of the product with its current functionality, the few remaining pieces of functionality such as the Google Docs functionality will need to be implemented. Once that is finished, extra functionality can be added onto the application should time permit it. The product will be demonstrated in Week 13, roughly 2 weeks after this demonstration of the prototype occurred.

9. Security test plan

9.1. Timeline

Projected timeline

Start time: 9 AM

End time: 5 PM

Day	Date	Task
Day 1	10 May 2021	Define testing objectives and scope, drafting test plan.
Day 2	11 May 2021	Review the web application, identify the vulnerabilities.
Day 3	12 May 2021	Setting up the testing environment, perform the test.
Day 4	13 May 2021	Continue testing.
Day 5	14 May 2021	Write penetration testing report.

9.2. Approaches

A combination of unit testing and white box testing will be done on the application. The approaches taken with white-box testing will model the Lockheed Martin's Cyber Kill Chain to ascertain as many vulnerabilities and exploits as possible.

White box penetration testing

White Box Testing is a software testing methodology or technique in which the software's internal configuration, architecture, and coding are examined to validate input-output flow and improve design, usability, and security.

This is also known as clear box testing because the code is visible to testers.

Unit testing

Since development of the application is a core part of the project, unit testing on the software components will need to be conducted as well. A standard unit testing approach will be taken with each component of the application being tested in isolation as well as their interaction with other sub-systems (where applicable). In addition to this, regression testing of existing components will be done when new additions are made to ensure the whole application still functions as intended even when updates and additions are being made.

Black box testing

A future testing approach that can be considered will be that of black-box testing. This is where testing of the application is done without any knowledge of internal code structure, implementation details and internal paths. This would require a third-party to be given the all-clear to find and exploit any vulnerabilities or oversights in the application.

At this current stage of prototyping and development we're conducting testing on our own and haven't gotten to the stage where the application is at the scale that requires black-box testing to be conducted on it, thus why black-box testing is being considered as a future option.

9.3. Resources Required

Resource Name	Used for
Web server	For hosting the application.
Kali Linux	For conducting security and pen testing on the application.
Burp Suite	Spidering, active scanning, finding vulnerabilities and testing the system.
Zap Proxy	Spidering, active scanning, finding vulnerabilities and testing the system.
Nikto	Finding application vulnerabilities.
GoBuster, Dirb	Finding hidden files and directories.
SQLmap	For SQL injection.
SSLScan	Finding weak cryptographic protocols for SSL/TLS.
Wget and Curl	For downloading webpage
Wireshark and TCPdump	For packet capturing and analysing

9.4. People Involved

Phase	People Involved
Define scope and test plan	Md Mehedy Hasan
Pre-test setting	Md Mehedy Hasan, Socheat Chunn
White box Testing	Md Mehedy Hasan, Socheat Chunn, Eddie Morris, Jarrod Adair
Black Box testing	Md Mehedy Hasan
Repairs and Maintenance	Md Mehedy Hasan, Socheat Chunn, Eddie Morris, Jarrod Adair
Writing test reports	Bradley Anderson, Md Mehedy Hasan, Socheat Chunn, Eddie Morris, Jarrod Adair

9.5. Approvals and Reporting Needs

Since tests are being conducted on an application created from a template developed by the client, approval from the client will be needed to conduct pen testing on an application deriving from their creation.

When tests are being conducted, the overview, process and results will be documented, the reporting will include the following:

- The test code of the relevant test.
- The date/s and time/s that the test/s were conducted.

- The results of the test (Make mention of if the test procedure could not be followed as detailed for the relevant test case).
- Any actionable items after the test (things to be fixed or improved)

9.6. List of Test Cases

Test Name	Testing form building and creation.
Test Code	DEV01
Aim of Test	To test that it is possible to add fields to a form, change the field names and generate a link.
Process	1. Navigate to the 'Create Form' page. 2. Click the 'Add New Field' button. 3. Enter names and descriptions for each added field. 4. Click the 'Generate Link' button. 5. Navigate to the link and open the form, all of the fields that were made with the given names should be present and have text boxes to enter information into.

Test Name	Testing form verification procedure.
Test Code	DEV02
Aim of Test	To test that when a respondent opens a form, they are required to input the correct verification code to access a form.
Process	1. Navigate to the form's link, there should be an input saying, 'Enter Verification Code'. 2. Enter the wrong verification code, there should be an error saying, 'Incorrect Verification Code'. 3. Enter the correct verification code, you then should be redirected to the form.

Test Name	Testing form response and storage.
Test Code	DEV03
Aim of Test	To test that when a respondent submits a response, the data comes back to the database properly
Process	1. Navigate to the form's link, enter information into all the fields. 2. As a check, try clicking 'Submit Form' with blank fields, there should be an error saying that a form has been left blank. 3. Click the 'Submit Form' button once all the fields have been filled in with information. 4. Once submitted, check the main application page to see the submitted form with the relevant fields filled in. Alternatively, check the database for the form ID and see if the fields in the form object are filled in.

Test Name	Testing email functionality.
Test Code	DEV04
Aim of Test	To test that all emails specified to receive a form receive both the link and verification code in their email.
Process	1. On the form creation screen, create a form as normal. 2. In the 'Recipient Email' field, enter the emails that you want to send the form to, separated by commas. 3. Click the 'Generate Form' button. 4. Check the emails put into the field, there should be an email with the form link and verification code.

Test Name	SQL injection
Test Code	SEC-01
Aim of Test	To test if the web application vulnerable to SQL injection
Process	1.Find any links that connected to database. 5. 2.Test for both blind and error-based SQL injection including union select.

Test script	SELECT * FROM Users WHERE Username='1' OR '1' = '1' AND
	Password='1' OR '1' = '1'

Test Name	Stored and Reflected Cross-site scripting
Test Code	SEC-02
Aim of Test	To test if our application vulnerable to cross-site scripting attack
Process	1. Find any page with form. 2. Execute JavaScript such as alert, iframe to observe its action.
Test script	https://portant.co"><script>alert(document.cookie)</script>

Test Name	Directory traversal
Test Code	SEC-03
Aim of Test	To test the application vulnerable to directory traversal attack
Process	Find URL and try the test script
Test script	http://portant.co/getUserProfile.jsp?item=../../../../etc/passwd

Test Name	Test for supported HTTP Method
Test Code	SEC-04
Aim of Test	To test HTTP method implemented correctly
Process	1. Open a terminal in kali 2. Use nikto to find HTTP method vulnerabilities.
Test script	nikto –h http://portant.co

Test Name	Distributed Denial of Service
Test Code	SEC-05
Aim of Test	To test if the WAF can protect the application ddos attack
Process	1. Open a terminal in kali 3. Run the python script and specify the protocol, packet size, count per minute and application url or ip and attack
Test scipt	ddos.py

Test Name	Weak Cryptography
Test Code	SEC-06
Aim of Test	Find any weak cryptographic protocol used in our application and network
Process	1. Perform some activity in our site and capture the packet Wireshark from that activity and examine the packet and cryptographic protocol. 2. Use network mapper (nmap) script to enumerate certificate information. 3. Use curl to Test HTTP Strict Transport Security
Test script	nmap --script ssl-cert,ssl-enum-ciphers -p 443,465,993,995 <target> curl -s -D- https://target.com/ | grep Strict

10. Testing Reports

The reports of testing that have been conducted as listed in Section 9. Test cases are listed corresponding to their test code.

10.1. Test Results

The results from conducted tests. They are listed in order of when they were conducted.

Test ID	Date	Outcome	Actions Taken
DEV01	26/04/2021	Test was successful as expected	None required.
DEV02	26/04/2021	Test was successful as expected	None required.
DEV03	26/04/2021	Test was successful as expected	Will need to be re-tested once additional data and information is being stored.
DEV04	17/05/2021	Test was successful as expected	None required.

10.2. Pending Tests

We will conduct our web application security test against OWASP top 10 Security risks. While

Test ID	Vulnerability
001	SQL injection
002	Broken Authentication
003	Sensitive Data Exposure
004	Broken Access control
005	Security Misconfiguration
006	Cross site scripting
007	Insecure Deserialization
008	Insufficient Logging and Monitoring
009	Using components with known vulnerability
010	Distributed denial service

10.3. New Tests / Approaches

Once development reaches later stages with the added form creation functionality and Google Drive integration, tests on both the function and security of both features will need to be done. These tests will follow the same approaches as with current tests for application functionality and security. The areas that would be covered are:

- Google Docs API functionality and security regarding interactions with the API
- Enhanced form creation functionality with data types, files, and images
- Appropriate security tests that ensure file and image inputs cannot cause weaknesses in the system.

11. References

What is a Web Application Firewall (WAF)? cloudflare.com. (2021). Retrieved 28 April 2021, from https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/.

What is Security Testing? Types with Example. Guru99.com. (2021). Retrieved 28 April 2021, from https://www.guru99.com/what-is-security-testing.html#5.

Cyber Kill Chain®. Lockheed Martin. (2021). Retrieved 28 April 2021, from https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html.

OWASP Top Ten Web Application Security Risks | OWASP. Owasp.org. (2021). Retrieved 28 April 2021, from https://owasp.org/www-project-top-ten/.

What is an Intrusion Prevention System? Palo Alto Networks. (2021). Retrieved 28 April 2021, from https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips.