

Report

# Secure Development Life Cycle

---

**3 Musketeers Tech**

Michael Arcangel, Caleb Cheshire, Christian Cheshire

ICS 427, Summer 2020



## Introduction

[Team Name](#)

[Team Members](#)

[Application Name and Description](#)

[Functional Requirements](#)

[Program Type](#)

[Development Tools](#)

## Requirements

[Security and Privacy](#)

[Quality Gates](#)

[Risk Assessment Plan for Security and Privacy](#)

## Design

[Design Requirements](#)

[Attack Surface Analysis and Reduction](#)

[Threat Modeling](#)

[Identifying Threats](#)

[Categorizing Threats - STRIDE Model](#)

## Implementation

[Approved Tools](#)

[Deprecated/Unsafe Functions](#)

[Eclipse](#)

[Static Analysis](#)

## Verification

[Dynamic Analysis](#)

[Attack Surface Review](#)

## Verification

[Fuzz Testing](#)

[Fuzz Testing - Attempt 1](#)

[Fuzz Testing - Attempt 2](#)

[Fuzz Testing - Attempt 3](#)

[Static Analysis Review](#)

---

[Dynamic Review](#)

[Release](#)

[Incident Response Plan](#)

[Privacy Escalation Team](#)

[Contact Information](#)

[Incident Handling Procedures](#)

[Final Security Review](#)

[Threat Modeling](#)

[Static and Dynamic Analysis](#)

[Quality Gates](#)

[FSR Grade Explanation](#)

[Certified Release & Archive Report](#)

[Features](#)

[Version](#)

[Future Development](#)

## Introduction

### Team Name


3 Musketeers Tech

### Team Members

- Michael Arcangel
- Caleb Cheshire
- Christian Cheshire

### Application Name and Description

The application being developed is called Ace PassWord (AcePW). AcePW is a lightweight password manager application designed for users to gain quick access to



their passwords for everyday use on a personal device. AcePW does not rely on cloud based data storage and therefore stores encrypted passwords locally on the user's device for the fastest possible access while mitigating the risk of server crashes or data breaches. It does not provide any connection to web browsers or other network-related applications (e.g. autofill functions). This minimizes the risk of user login credentials being compromised and provides better security for the user.

## Functional Requirements

AcePW stores user's encrypted password data and enables them to access it via simple commands. The application must be simple and intuitive for users to use but robust enough to ensure privacy and security of user information. This will require instructions provided to the user regarding the various capabilities of the different commands as well as code which meets the security requirements. Security, accessibility, and integrity will be the fundamental focus areas for program functionality.

## Program Type

Device Application - Recommended for use only on a user's personal device. Passwords are stored locally to ensure no parties other than the user are able to access their secured passwords. No personally identifiable information will be stored using AcePW.

## Development Tools

- Java
- IntelliJ IDEA
- EclipseIDE
- Command Line

## Requirements

### Security and Privacy

As a password manager, AcePW must implement and ensure the following security and privacy requirements:

- User authentication information (e.g. master password) must be kept secure. This is the primary focus of the system, as user authentication enables access to the various other passwords which the user stores using AcePW.
- The user's passwords and information regarding the sites or applications which they are to be used for must be kept secure. This means the system must be resistant to hacking and data breaches.
- The system must provide basic encryption of user data and authentication information to ensure that it is not vulnerable to hacking.

Developers must follow established best coding practices and specific team practices to ensure that code is clearly written and understandable. This will minimize risk of errors and potential security risks.

Additionally, developers will review and check the security of the system throughout the development of this application. During the development process, security flaws or vulnerabilities will be tracked via Google Docs and GitHub Projects.

### Quality Gates

As a local system application, AcePW will not require networking or transfer of data to other users or systems. Thus, the typical threats associated with a web application or plug-in will not apply. However, there are still threats of local attacks or attackers accessing the system remotely to steal user data or damage the program/system.

- Critical
  - System worms or unavoidable use scenarios (without warnings to user)
  - Elevation of Privilege (remote)
    - If the system is hacked remotely, arbitrary code could be run to create higher-level privileges for the attacker to gain an advantage
- Important
  - Elevation of Privilege (local)
    - If the attacker can gain administrator access to the system, or access the system through another user, the program could be compromised
  - Tampering
    - Modification to the system that allows the attacker to change program settings or user data
  - Security Features
    - Using weak encryption or allowing the system to be bypassed (BitLocker or Syskey)
- Moderate
  - Information disclosure
    - Unauthorized file system access could be a threat; disclosure of Personally Identifiable Information (PII) should not be a concern since the program does not require such data to be collected or stored
- Low
  - Tampering
    - Temporary modification of data
    - Untargeted information disclosure

## Risk Assessment Plan for Security and Privacy

- Risk Assessment Plan
  - Risks will be assessed throughout the design and development process via team risk analysis sessions in which team members will review and analyze possible risks and develop methods for eliminating those concerns or reducing the risk to an acceptable level.
  - Acceptable risk means that the potential threat will have no impact on the user or interfere with the security, accessibility, and integrity of the application.
  - Threats involved are mainly focused around users interacting with the device and unauthorized parties that may interact with the device.
  - Prepare for viruses or other malicious software into the users system (writing secure code).
- Initial Assessment
  - AcePW represents a low privacy risk (P3).
  - There will be no behaviors within the application that affect privacy. No anonymous or personal data will be transferred. No personally identifiable information (PII) will be stored on the user's machine. No settings will be changed on the user's behalf, and no software will be installed by the application.
- Additional Details
  - Users will be provided with a privacy notice which explains that AcePW does not collect or compromise any PII. This privacy notice will also remind users to avoid incorporating any information that could potentially identify them into their passwords or user credentials. This will help ensure the user's privacy when using their passwords to login to other applications or systems (not related to AcePW).

- The privacy notice will be provided in the form of a document and also displayed in the application.

## Design

### Design Requirements

- Authentication System
  - User master password
  - Basic password requirements (e.g. length, special characters, etc.)
- Password Encryption Method
  - Ensure passwords are secure even if the database is accessed by malicious parties
- Password Access/Removal/Modification Capability
  - Reduce risk of accidental deletion
  - Simple, intuitive methods for user to add/modify/delete passwords

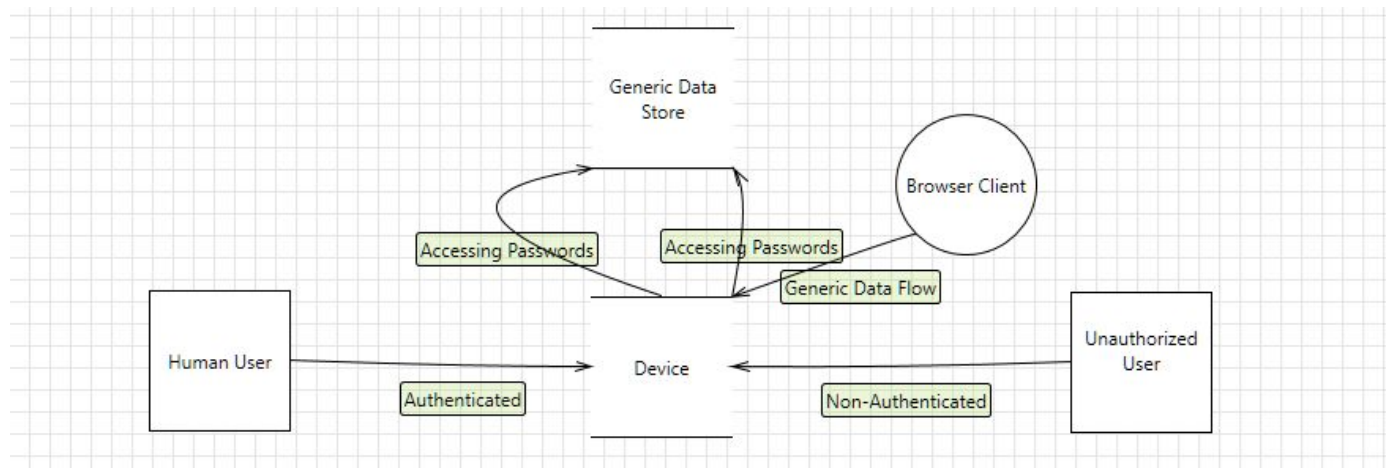
### Attack Surface Analysis and Reduction

As determined by Microsoft in their priorities for calculating a relative attack surface, our main concern for this application will be the enabled accounts with access to this application and its contents. Should an unauthorized user or anyone other than the intended user gain access to the application and its contents, the application will have failed in its task to safeguard the user's passwords. Regardless of whether or not the authentication is bypassed, individuals not intended to access the system may be capable of locating the directory in which the passwords are locally stored with the only step then being to decrypt each password. For similar password manager software, the attack surface is relatively the same. Namely, after searching for more info regarding similar softwares it can be found that the major problem for password managers as



opposed to online password managers is neglecting to remove data thoroughly from storage.

## Threat Modeling



## Identifying Threats

- Human User
  - Failing to properly secure personal device
  - Failing to wipe and remove all password information from data storage
- Unauthorized User
  - Gain access to user's system, bypass or brute force authentication
  - Stems from user being unable to secure their own device
- Browser Client
  - Interaction between browser client/users may introduce new threats to the system
  - May potentially obtain password information depending on the threat

## Categorizing Threats - STRIDE Model

- Human User - Information Disclosure
- Unauthorized User - Spoofing/Tampering
- Browser Client - Tampering

## Implementation


### Approved Tools

This list of approved tools assumes that (with the exception of the jasypt Java library for encryption) only the use of the provided libraries within the listed Eclipse IDE and only the provided features of Git/Github Desktop to prevent complications with cross-platform compatibility.

- Eclipse IDE ver. 2020-03
  - Will serve as the main tool for coding and implementing all desired functions of AcePW
- Git ver. 2.26.2
  - Platform for commit and pull requests to and from the GitHub Repository. Mainly used for doing so within the Eclipse IDE.
- Windows 10 64-Bit
  - Standardized OS to run Java SE 14 with the associated JDK
- GitHub Desktop ver. 2.5.2
  - Same purpose as Git, dedicated platform as opposed to integrated commit or pull functions within the IDE. Simple UI allows for branch management and merges
- Jasypt ver. 1.9.2
  - Java encryption library for simplified encryption of the project; integrated with existing libraries in the Eclipse IDE.

### Deprecated/Unsafe Functions

For this list of deprecated or unsafe functions, we will assume that most if not all of the listed functions may or may not be used during the development process. Special attention is given to the Immutable/Mutable Object since such objects cannot/can be



modified after creation. This feature should prove useful to proper password storage and encryption to avoid data corruption or loss.

## Eclipse

- `org.eclipse.collections.impl.tuple.AbstractImmutableEntry.getKeyFunction()`
  - `Use Functions.getKeyFunction()`

Potential purpose: Use `getKeyFunction` in order to retrieve a particular access key for a password's location and prepare to read and pass the value of that location to the user.

- `org.eclipse.collections.impl.tuple.AbstractImmutableEntry.getValueFunction()`
  - `Use Functions.getValueFunction()`


Potential purpose: Use `getValueFunction` in conjunction with `getKeyFunction()` in order to read the value out to the user after the requested key is used to obtain the desired password.

- `org.eclipse.collections.impl.list.mutable.ListAdapter.newEmpty()`
  - `Use Fastlist.newlist()` *can be used inline*
- `org.eclipse.collections.impl.test.Verify.assertNotEquals(String, String)`
  - `Use Assert.assertNotEquals(Object, Object)`

Potential purpose: For user authentication prior to being able to access their stored passwords. Will check their entered password against the initially created and stored password for AcePW.

## Static Analysis

For this project, Checkstyle will be used as the static analysis tool. This ensures that all developers on the project adhere to the same set of coding standards which will reduce the risk of incorrect code and potential errors. Checkstyle inspects the Java source code and highlights items which do not meet the coding standards so that they can be fixed by the developers. However, its checks are primarily limited to the



presentation and formatting of the code and do not confirm the correctness of the code or its effectiveness.

Checkstyle is an extremely useful tool and has enabled the development team to work more efficiently and effectively. When mistakes are made, they are quickly noticed and highlighted by Checkstyle and the developer can then immediately fix the error and ensure that all of the code is correct and meets the standards expected for this application.


Specifically, Checkstyle helps significantly with spelling errors (which occur often when developers are typing quickly). The only negative aspect of its use is that the tool sometimes flags spelling errors for words that are not in its dictionary. However, this is a minor annoyance and not a major problem. Developers can set Checkstyle to ignore the spelling errors when necessary.

Checkstyle also provides flexibility and the ability to configure the set of rules for the specific project being developed. For this project, Checkstyle is integrated directly into the Eclipse (and IntelliJ) Integrated Development Environment (IDE) which enables the development team to work seamlessly from different platforms and minimizes the risk of confusion or errors due to different coding styles. It is also an excellent tool for alerting the developer to unused functions or instances which are unnecessary and can be removed from the code. This helps to ensure clean, efficient code which will be much easier to deal with when refining the application and preparing it for release.

## Verification

### Dynamic Analysis

For the development of this application, Diver (Dynamic Interactive Views for Reverse Engineering) will be used as the dynamic analysis tool. Diver functions within the



Eclipse IDE and provides an interactive trace-focused interface. It also provides support for understanding and exploring items of interest during the debugging process.


Diver provides these capabilities via an Eclipse plug-in, which enables recording traces of a running program and the ability to navigate through trace-focused source code views. It also provides easy to configure filters that can be applied before, during or after trace capture and visualization. All of these abilities make Diver an incredibly useful tool for development of our software.

In practice, Diver has proved to be very powerful and capable for the analysis of the source code being developed and tested for this program. It has enabled the development team to carefully “dive” into a trace and gain valuable information about sections of the code which need to be reviewed for possible errors or behavior that does not meet the requirements of the final product.

Some frustrations encountered involved the vast amount of information presented, which sometimes makes it difficult to comprehend quickly. Additionally, it enables a better general understanding of the software. Overall, this tool has been quite useful in multiple instances for helping to ensure a better application.

## Attack Surface Review

The attack surface has not significantly changed since the initial analysis. The same concerns and threats mentioned in the initial analysis of the attack surface still apply for the most part, and due to the limited exposure and scope of the product, an environment with minimal risk is still maintained in the program. Another aspect of attack surface that has been evaluated in this review is the adequacy of the developer’s tools that are being utilized to create the program. It is important for our team to ensure that these tools are up to date and have key patches and fixes installed. This tightens our scope of security, and further limits the attack surface.



The tools that have been approved for use of the program (see “Implementation - Approved Tools”, page 7) have not seen any updates or patches from the developers as of 6/13/20, with the exception of Git. Git received some minor patches and fixes in the latest update (v2.27) released 6/10/20. These fixes included additional warning messages to users, added configuration options for a Secure Socket Layer (SSL), eliminating functions which could be misused, etc.

As of now, using the current version of Git does not present a substantial developmental or security risk to the AcePW product, but an update to the most current version could be completed prior to product release. All of the other tools are up to date and have not presented any issues during the development process.

The addition of jasypt for encryption capabilities potentially expands the attack surface of the project, but is a necessary aspect of quality assurance and security on the part of our development team. The current version being utilized is up-to-date with no known vulnerabilities.

The project team will continue to monitor for bugs within the existing tools and will update as needed to preserve project continuity and security. Weekly reviews of the software for patches/fixes should be adequate for this, as well as monitoring the tools for bugs while utilizing them in the development process.

## Verification

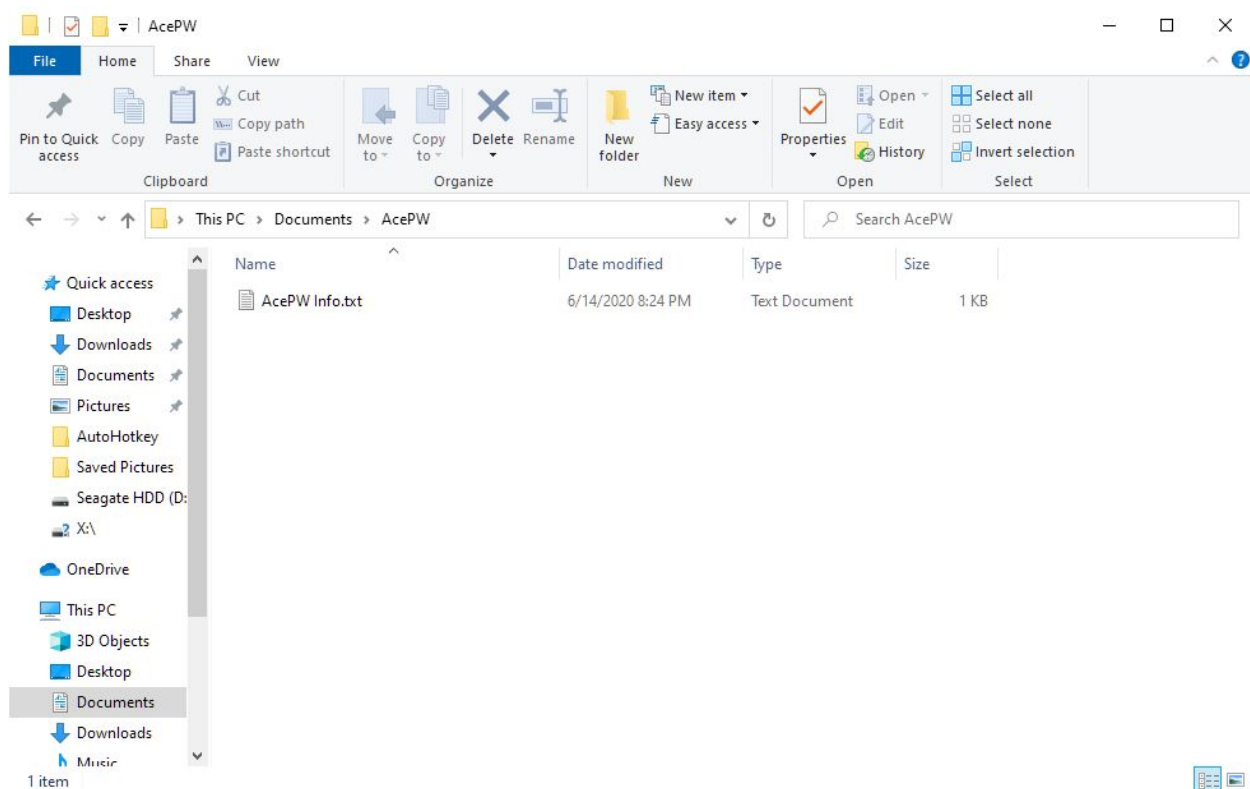
### Fuzz Testing

Prior to beginning any kind of fuzz testing, some of the rougher patches in the code had to be straightened out. Upon each test and launching the main method for the password manager, we would encounter a “File not Found” error even though when checking the directory where the file was supposed to be stored, the file existed. After fixing some issues with the path for the file, we were able to begin fuzz testing. For all

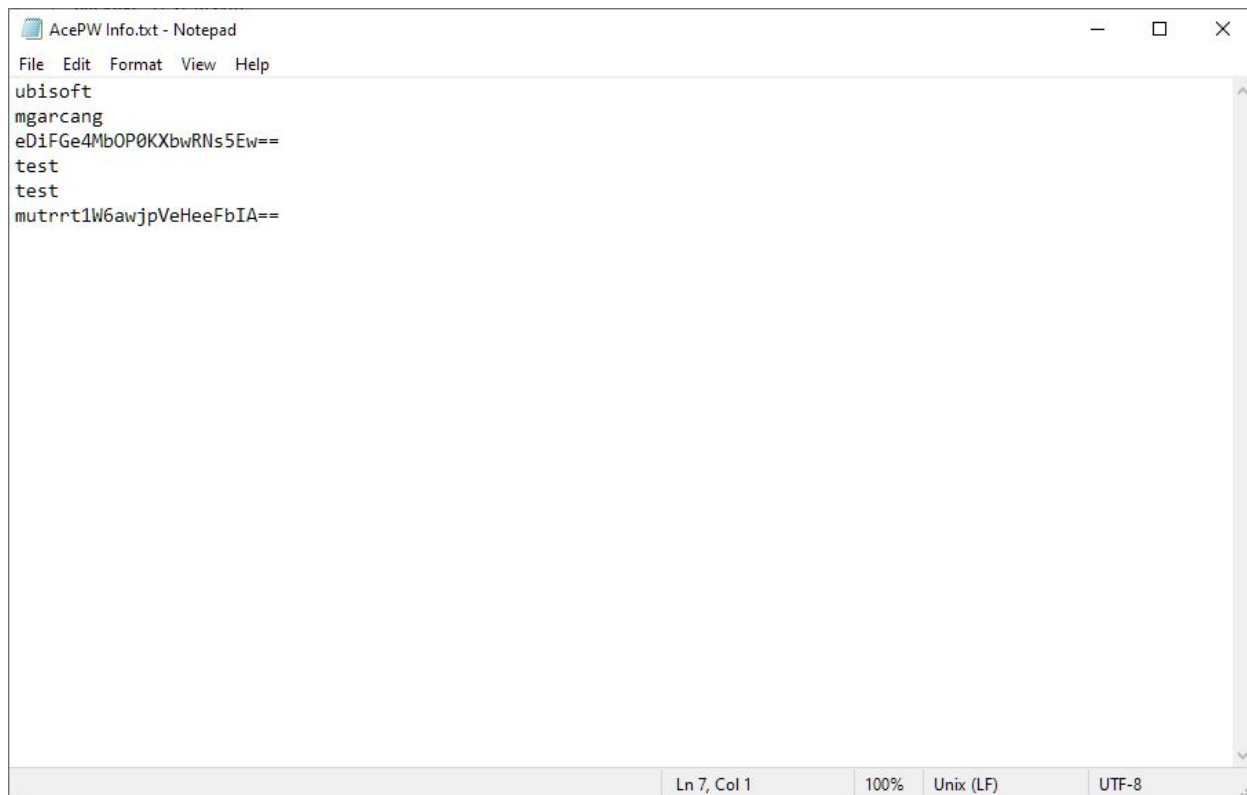
methods below, the password that was encrypted originally is “dog”. The key used for encryption will be “hans”.

## Fuzz Testing - Attempt 1

For the first method of testing our password manager, we kept it simple and wanted to test the strength of the encryption without using jasypt to decode the password string. The first step was only to access the .txt file located in the documents section of the user’s Windows 10 device.



After accessing the file:



```
AcePW Info.txt - Notepad
File Edit Format View Help
ubisoft
mgarcang
eDiFGe4Mb0P0KXbwRNs5Ew==
test
test
mutrrt1W6awjpVeHeeFbIA==
Ln 7, Col 1 100% Unix (LF) UTF-8
```

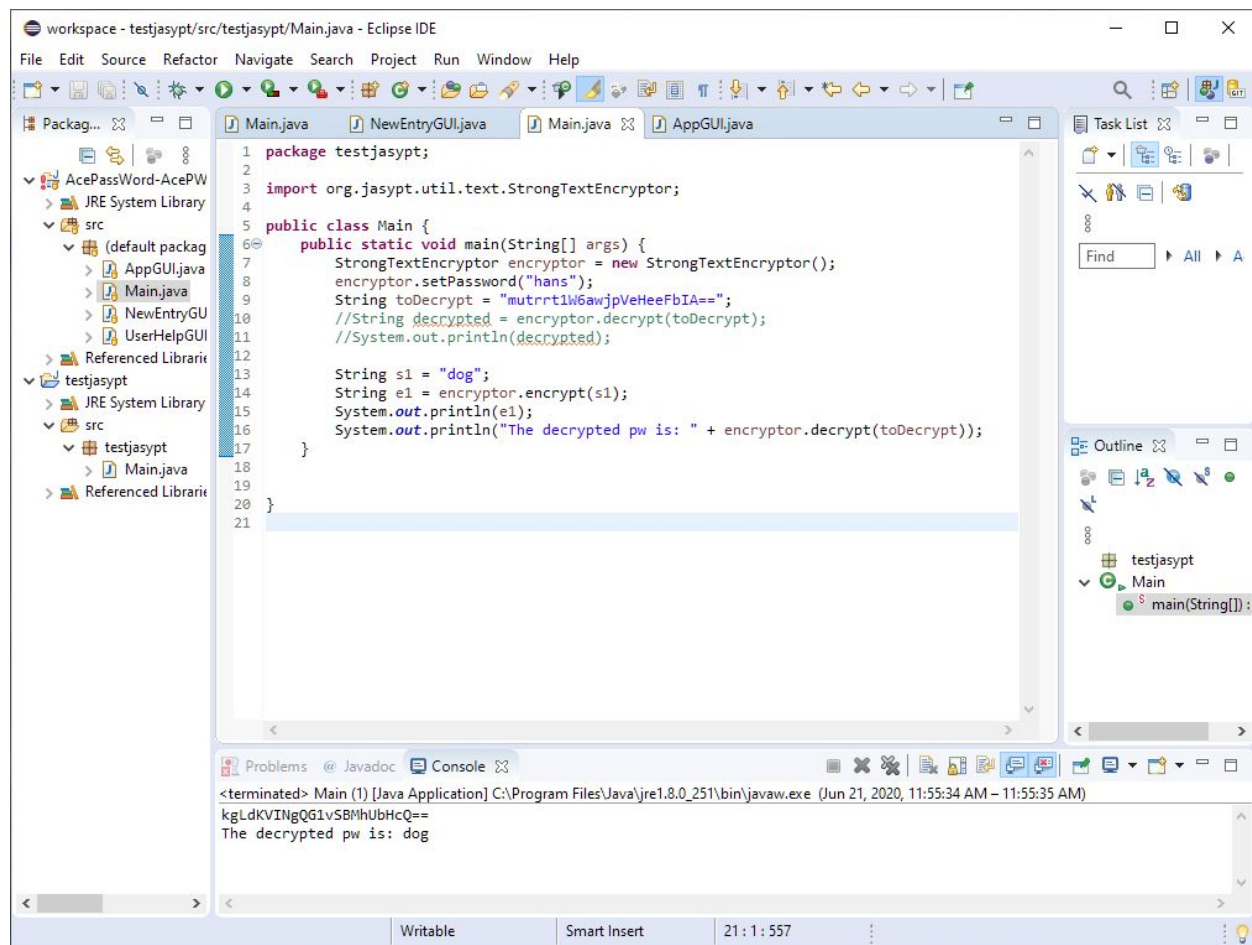
The first two lines prior to the encrypted string are listed for the website and username that was entered prior to the password. We were unable to decrypt the string ourselves through the use of online decryption tools. The site used was <https://codebeautify.org/encrypt-decrypt>, which provided different encryption algorithms and modes in order to decode the string. Using each algorithm to try and decrypt the user's password did not return a correct result.

### Fuzz Testing - Attempt 2

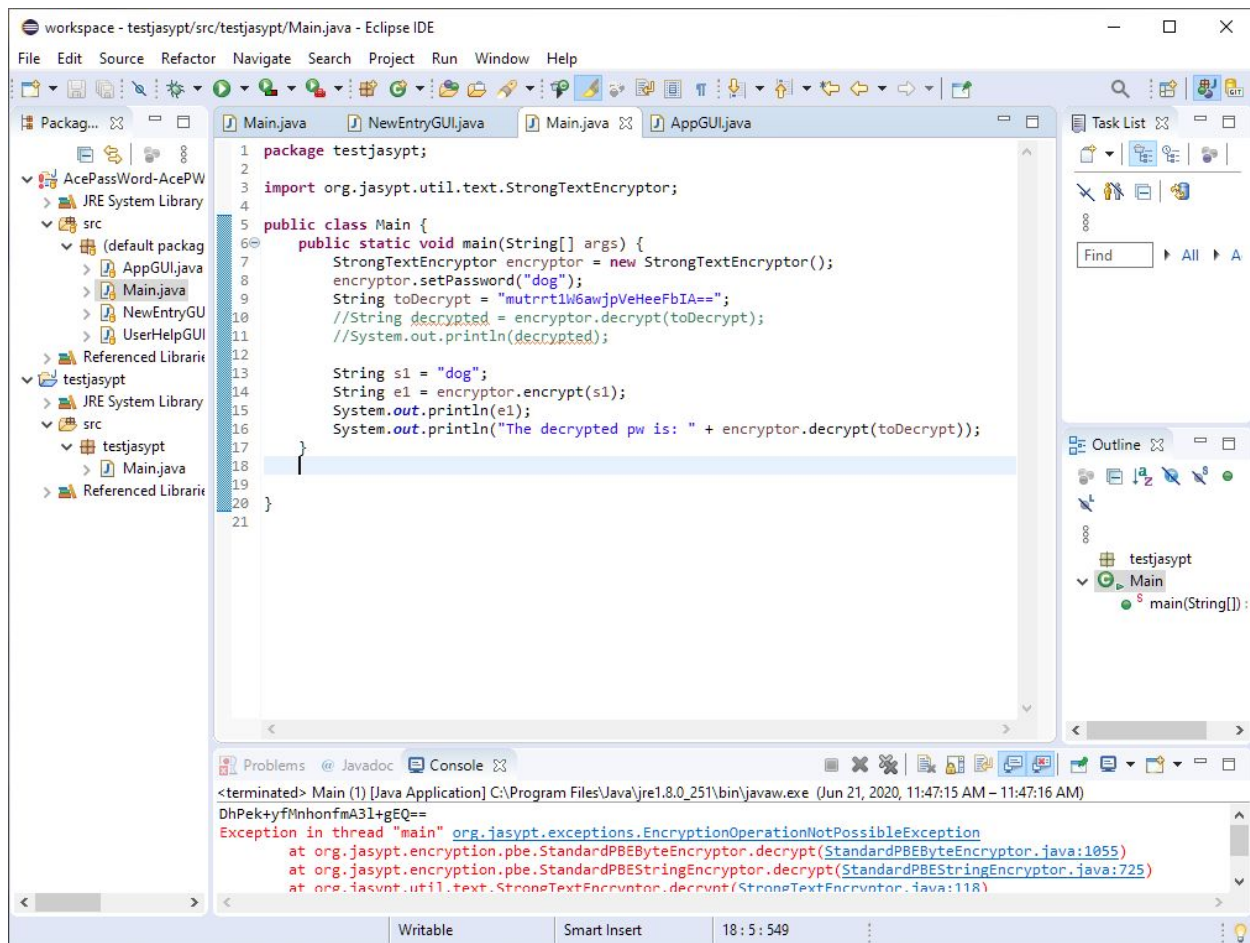
Our second attempt involved creating a new Java project unassociated with the current password manager project and using jasypt to decode the encrypted string. The purpose behind this is to test the reliability and security of the jasypt encryption library. If Jasypt has a secure enough procedure for encryption, another instance of the encryptor object should not be able to decrypt a string that has been encrypted by a separate



instance of the object. To make this as simple as possible in order to isolate the main focus of the testing towards the jasypt library, I only created an instance of the encryptor and tried to decrypt the string that I passed within the Eclipse IDE. As shown below:




In this attempt, I copied the encrypted string from the password manager's .txt file and used the same key that was used to encrypt the password originally within the AcePW project. As seen in the dialog box at the bottom, the correct password was decrypted from the encrypted string that was passed through.



However, when the password used for the encryptor is changed to something other than the original password, an error is returned stating that the decryption is not possible. From this we can determine that the only way to decrypt the user's password string would be to use the original "password" or key that was set for the encryptor.

### Fuzz Testing - Attempt 3

For the third method, I sent the .txt file to another person and only gave them the context that the encrypted strings are a list of passwords. The purpose behind this attempt was to simulate the event of a hacker breaking into the system and being able to find and obtain the password text file. Furthermore, this method would also show whether or not someone on the outside of the development team would be able to decrypt the strings without knowing the type of encryption or the method used to



encrypt the strings. Again, this returned the same results as the first attempt: the user from the outside was unable to decrypt the password string sent into the password manager, even after obtaining the .txt file.

## Static Analysis Review

Based on the latest static analysis review using Checkstyle, there are no significant vulnerabilities. While this means that the likelihood of any security problems with the code itself are low, there is still the potential for problems at runtime. There have been some errors caught which primarily were just incorrect spelling of certain comments and variables.

Most of the issues which came up have all been fixed by either fixing the spelling or (in the case of wanting to use that particular spelling) using @suppress or other methods to rectify the problems and ensure that all errors identified by the static analysis have been addressed. Since Checkstyle immediately notifies the development team of an error, almost all of the problems were caught right away and fixed without any issues. Additionally, the tool warned of unused imports and variables which enabled the developers to remove unnecessary code and ensure that everything is streamlined and efficient.

Overall, the development team has ensured that all source code meets the coding standards established at the beginning of the course and (at least from the point of view of static analysis) will not be vulnerable to the anticipated or unanticipated issues that could occur. While static analysis cannot check everything which might possibly be an issue with the application, it has ensured that when a problem is found it can be more easily identified and rectified due to the clarity and organization of the source code.



## Dynamic Review

A review of our dynamic analysis using DIVER (Dynamic Interactive Views for Reverse Engineering) gave us further insight into the performance of our program and enabled us to resolve some minor issues with the engineering of the code.

With the capability that DIVER provides, our development team was able to “dive” into several traces and identify inconsistencies and errors in multiple sections of code which would not allow our product to run as efficiently as possible once released to the consumer. While these issues were not immediately apparent, using DIVER as a dynamic analysis tool allowed us to review how well our program was engineered, and how effectively it would operate in the customer’s environment.

Overall, the dynamic review of our code was fairly easy, since the static analysis and our approach to engineering the program ensured that our structure started off as efficient and straightforward as possible. However, it did provide that extra layer of testing and analysis which enabled us to work out some of the smaller bugs that we would not have been able to catch otherwise, helping us to optimize the efficiency of our software and our development process.

Conducting dynamic analysis of our code has proven to be an excellent method for catching small inconsistencies, while we deal with most of the larger ones using static analyses, CheckStyle, etc. Using DIVER as a tool has freed up our development team from the requirement of carefully sifting through the program to find vulnerabilities and inconsistencies which could pose a problem to further product development.

Essentially, a review of our dynamic analysis confirms that it has been effective and has helped our team to design a better, safer, and more streamlined product. The effects of dynamic analysis go beyond simply checking the code for efficiency and safety, as it frees up our team to focus energy on further improving other aspects of the product.



## Release

### Incident Response Plan

#### Privacy Escalation Team

In the event of security and/or privacy issues with AcePW, the response will be handled by the Privacy Escalation Team.


- Escalation Manager: Caleb Cheshire
  - Responsible for leading the team, coordinating with outside resources, and ensuring the process reaches completion.
- Legal Representative: Christian Cheshire
  - Responsible for ensuring legal representation for the team and product.
- Public Relations Representative: Christian Cheshire
  - Responsible for representing the team and product to the public, addressing public concerns, and interacting with appropriate members of the government and general public.
- Security Engineer: Michael Arcangel
  - Responsible for handling the technical aspects of the incident and resolution process.

#### Contact Information

In the event of an emergency, the Incident Response Team can be reached by contacting Michael Arcangel at [mgarcang@hawaii.edu](mailto:mgarcang@hawaii.edu).

#### Incident Handling Procedures

The escalation process begins when an email notification of the issue is received. The Escalation Manager will then evaluate the content of the escalation to ensure that all



necessary information has been received and gather more information if required. He is then responsible for working with the reporting party and other contacts to determine:

- The source of the escalation
- The impact and breadth of the escalation
- The validity of the incident or situation
- A summary of the known facts
- Timeline expectations
- Employees who know about the situation, product, or service

The escalation manager must then disseminate this information to the appropriate personnel and lead the resolution process. He should ensure that all aspects of the escalation are resolved, but may assign portions of the workload to other people if needed. Appropriate resolutions should be decided upon by a privacy escalation core team working in cooperation with the person(s) who reported the issue and other applicable personnel.

Resolutions could include (as outlined in the Microsoft SDL Privacy Escalation Response Framework):

- Internal incident management
- Communications and training
- Human resources actions, in the case of a deliberate misuse of data
- External communications, such as:
  - Online Help articles
  - Public relations outreach
  - Breach notification
  - Documentation updates
  - Short-term and/or long-term product or service changes



## Final Security Review

Upon final review of the overall security for the software, AcePW has: **Passed FSR**. See the details for the review process below.


### Threat Modeling

The threats identified here have been either successfully neutralized and accounted for or mitigated within an acceptable level of security.

- Human User
  - All data passed through to the storage text file is now encrypted, the human user will not be able to accidentally disclose the stored text
  - Users are still encouraged to follow strong password practices
- Unauthorized User/Browser Client
  - All attempts to fuzz test the password manager software were unsuccessful with the exception of directly using Jasypt to decrypt the string
  - Note: This required obtaining the original encryption key initially used to encrypt the string

### Static and Dynamic Analysis

Through continued use of the Checkstyle static analysis tool we were able to identify key mistakes and correct them immediately prior to beginning another portion of the project. These Checkstyle features enforce coding standards which in turn made developing the remainder of the project much easier since debugging and actually analyzing the code during the developmental stages became a much more simple task. Through our dynamic analysis it was noted and shortly implemented thereafter that the entire string of data being passed through to the text file should be encrypted, not just the password strings. This proved to be useful as we added a previously unidentified vulnerability to our list which was corrected through our dynamic analysis of the code and checking for



the expected results upon storing data to the text file and retrieving it to be displayed on the table.

## Quality Gates

### Elevation of Privilege

- Assuming remote or local access to the system, the program will only become compromised in the event that the hacker is able to obtain the encryption key in order to decipher the strings used in the password text file
- Should the hacker attempt to decipher the text file through other means, this should not be possible without first passing the strings through jasypt for decryption


### Information Disclosure and Tampering

- Data tampering will only further obscure the main objective of any attack on the system: to decipher and obtain password information from the target on various websites and platforms
- Users will be encouraged to maintain safe password practices for maximum security and deterrence

## FSR Grade Explanation

The grade we have decided on for **Passed FSR** revolves mainly around the concept that the encrypted strings absolutely require Jasypt as a medium for decryption. Any other methods utilized during fuzz testing show that the passwords are unable to be deciphered without the use of the Jasypt library. In the case of the one successful fuzz testing attempt to decrypt the password string, the only method found to be effective is to use the preset encryption key in order to obtain the original string from the encrypted string via the use of Jasypt. As such, in the event that an unauthorized user obtains the password text file with the user's information they will be unable to obtain any useful





information from it unless they have knowledge of the encryption method used in development.

## Certified Release & Archive Report

The release version of Ace Password Manager can be found here:

<https://github.com/christiancheshire/AcePassWord-AcePW/releases/tag/v1.0>

## Features

The features of AcePW include:

- Simple, user friendly interface
- Ability to store unlimited passwords
- Encryption to protect passwords
- Minimal storage space required
- Secure storage location for user information

## Version

The current version of Ace Password Manager is **AcePW 1.0**. This will be the release version for customers, and new versions will be released as bugs are found in the program (either by our development team or the public), and patches are required.

## Future Development

Future Development plans include:

- Improving the GUI to provide a more appealing interface
- Adding the ability for users to link the program with their web browser
- Adding the ability and permissions for the software to auto-fill passwords into the user's websites
- Adding the ability to delete passwords through the UI