University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2022

Arvest Cyber Security Tabletop Web Application

Ashwin Prakash, Jacob Smith, Matthew Mclendon, Rudy Ramirez, Zachary Chapman

Abstract

For this project, Arvest wants a web application that will use a database to generate cyber exercises. Cyber exercises are a great way to keep cyber defense teams up to date with current attacks while also practicing against known attacks. A cyber exercise is broken down into five parts that are the threat/incident, the target/assets, the vulnerabilities, the impacts, and the injections/persistence. All are crucial to the exercise and to make sure their team knows how to handle difficult situations when they appear. Skills are something that you must practice continuously to truly master and that includes defensive skills. Running different exercises with different scenarios that have you utilize the same skill set is a great way to master these skills. Our goal with this project is to create a website that will generate different scenarios for Arvest to use so they can perfect their cyber defense skills, so they will be prepared to face any threat they might come across.

1.0 Problem

Today, Arvest Bank is faced with the potential of cyber-attacks that would affect day-to-day operations for their business. Since they are a bank, they have sensitive information and personal information about their users that they need to protect. Banks are very susceptible to cyber-attacks because the information they hold can be used by malicious users for monetary gain. To better prepare themselves, they can practice cyber exercises routinely so they can understand how to respond in the best way possible when these issues and attacks occur. When the company knows what to look for and how to respond they can fix a problem internally before their consumers even know about it.

Ensuring that Arvest has experience with solving problems related to cyber-attacks that they might encounter is the best way to prepare them for true cyber-attacks from malicious users. This will give them more confidence to deal with problems when they arise and improve their ability to deal with them. It is important for Arvest to be knowledgeable in how to counter these attacks to keep their customer's information safe and about how to delay and defend against these cyber-attacks to make sure day-to-day operations do not get affected and users do not begin to worry about the integrity of Arvest.

2.0 Objective

The objective of this project is to create a web application that can be used by Arvest bank to generate cyber exercises for their team to work on and complete. This will be an application that Arvest can use to generate a cyber exercise for their cyber response team to work on to better develop their response skills to a cyber-attack. This application will be created using the knowledge that Arvest Bank has provided and will focus on the following categories: threat/incidents, targets/assets, vulnerabilities, impacts, and injects/persistence. The end goal is for the application to automatically generate these exercises by pulling one from each category and combining them into a scenario for the cyber response team to react to so they will be better prepared for when these attacks occur in real time.

3.0 Background

3.1 Key Concepts

Companies usually have teams of people that are responsible for fixing the problems that occur with their website, database, or applications. These people are not always working on a problem, so they need exercises to do when the workflow is slow. These exercises are a fantastic way to keep their skills sharp as they flex their brain over topics that they have not used in a while, or with which they do not have much experience with. This makes sure that they are prepared for all issues that arise from the website, database, or application.

A threat/incident is a set of circumstances that has the potential to cause loss or harm to a system. This is where cyberattacks start. The threat or incident is used to exploit a vulnerability in the system. Some examples would be not checking user input to secure the database from SQL injection attacks or storing passwords within a risky format/environment. Threats are the baseline of a cyberattack and where they start.

The target/assets are what we are trying to defend from malicious users. Usually, this is important or sensitive information such as classified documents or personal information. Personal information specifically is important due to it being one of the most common targets of a cyber-attack. Personal information could include a personnel's address, password, or even their social security number; all things a malicious user could use against you.

Vulnerability is a weakness in the system that can be exploited. This is how malicious users gain access to a system. This is one of the more difficult parts of a system to design because it is hard to recognize problems in a system that you design. Problems usually occur from a user doing something with the application that is not its intended purpose. That is why it is always best to have someone test your applications because they might try to perform actions in the wrong order or might try to do something that you as a developer did not intend a user to do. The end goal is to design a system with no vulnerabilities, but this is only something we can strive to achieve because it is impossible to make a perfect system.

Impacts are the day-to-day issues that the attack affects. In our case, we will have five impacts with those being legal/compliance, reputational, financial, health and safety, and operational. These are all things that could be hurt for the duration of the attack or something that could outlive the length of the attack. A damaged reputation is something that takes months if not years to fix.

If a user cannot trust you with their classified information, then they will not use your company. Consumers are less likely to go back to a company after a data breach as they feel that their information is not being protected. Losing more than half of your customers is going to severely hurt your company so its reputation has a significant impact.

Injects/persistence are real-world events that could affect how a cyber-attack is played out. These events are things that may or may not directly affect the application but will affect how the exercise is handled. For example, if a worm gets in the system and is not causing damage to the system yet, then the worm still needs to be removed because it could cause damage to the system or harm the company's image in the public eye.

3.2 Related Work

Prior to our involvement, members of Arvest’s security team would develop and present individual exercises tailored to specific scenarios to better their security and inform other employees about how to handle certain security risks. One such exercise is known as the Garmin exercise. In 2020, the smartwatch company Garmin experienced a ransomware attack for a period of five days. Arvest’s exercise for this attack included identifying the ransomware used, how the hackers specifically were able to attack them, what Arvest policies were in place at the time to prevent a similar attack against Arvest and identifying what actions they could take to improve the security of their company.

Another related work to note is the tabletop exercise packages created by the Cybersecurity and Infrastructure Security Agency. These packages are resources that help stakeholders or other faculty in the company conduct their own security exercises. This would allow them to create discussion amongst others within the organization about various cyber-attack scenarios and how to implement a real solution into their company. The important thing about these packages is that they are customizable. This implies that there exists over one-hundred tabletop exercises to choose from that will allow other faculty to find a package that most closely matches their needs. These packages include a variety of interesting features ranging from multiple scenarios to discussion questions. These scenarios include multiple physical and cybersecurity topics, such as civil disturbances, pandemics, industrial control systems, election security, ransomware, and vehicle ramming. A final thing to note about these packages is that they have questions that will be used to discuss pre-incident info, intelligence sharing, incident response, and post-incident recovery.

The goal of our involvement with Arvest is to develop a generator tool to create such exercises. This will improve the efficiency of creating such exercises by allowing Arvest to use previously created software to develop the exercises rather than constructing them by hand from scratch.

4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

The web application is composed of three pages: a login page, a database modification page, and a scenario generator page.

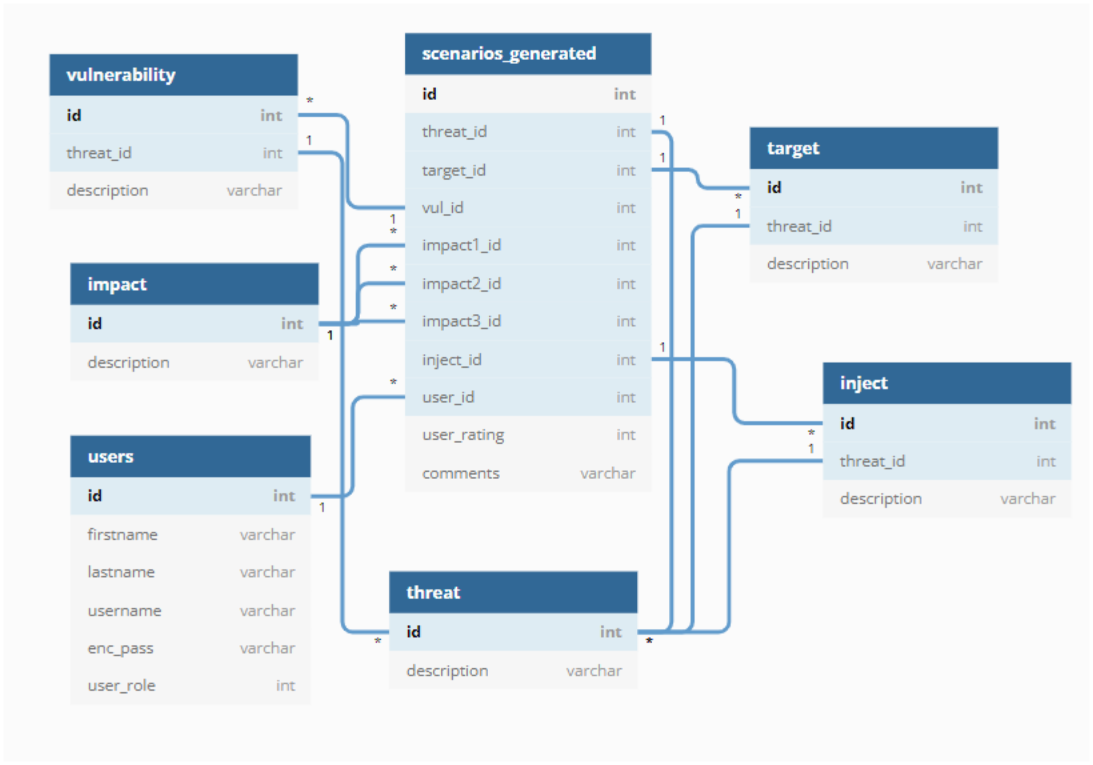
The login page is the startup page of the application and is designed to allow authorized personnel access to the tool. This page would utilize role-based user authentication system to allow different roles access to specific functions of the tool. The administrator (admin) role would have the highest security clearance and could create/remove users. Each user would be created using the following information: username, email, role, and password. The manager role would have access to both the Database Modification page and the Scenario Generator while the employee role would only have access to the Scenario Generator. This hierarchy security structure was requested by Arvest bank since they intend to utilize this as a teaching/testing tool.

The database modification page is the front-end portal that allows admins and managers access to the application's MySQL database. This database would contain the users' login information as well as the components used to create cyber security scenarios: threat, target, vulnerability, impact, and inject. Through this page, managers would be able to view, add, and delete any scenario component without writing SQL commands.

The scenario generator is the main tool of the application and can be accessed by all valid users. The tool would display a drop-down menu containing all the threats present within the database; it would then generate a random scenario using the selected threat. The tool should be able to generate a scenario from either a specific threat or a random threat (based on the user's choice). To make the randomized scenarios make sense upon creation, each target, vulnerability, impact, and inject component should be linked to specific threats. This would prevent unrealistic scenarios from being generated by only pulling information associated with the selected threat.

4.2 Detailed Architecture

- Database Schema



For the database schema, we wanted to create a table for each key element in the scenarios. We also wanted to create a table of the scenarios generated with a tie into the users to help prevent scenarios from being duplicated. So, as you can see above for threat, we have a table with just its ID and description. Because most scenarios will need to be based on a threat, we did not want to clutter up what a threat is so by keeping it simple we can then use it as a foreign key for other tables in the database. So, vulnerability, inject, and the target are all based on the threat, so these tables have their own ID, the threat ID, and a description. The last component of a scenario is the impact. The impact is another simple table as there are only 5 impacts so we did not feel that they should be related to the threat, and a scenario could have up to 3 different impacts. The 5 possible impacts are health and safety, operational, regulatory, financial, and reputational. The next table is the scenarios generated table this will hold a random scenario that is generated via a Python script, the main purpose of this table as stated before is to make sure no scenarios get repeated, we also included a user rating and comments section so that if this scenario has been played out the user could see how well it went and determine if they should do it again. Lastly is the user table, this is just a table that holds user information that will be used for security purposes like their username and password to log into the tool, as well as roles, so they can get access to the parts of the tool that they need and not the whole tool to protect the integrity of the data in the database.

- Login Page/Sign-up page

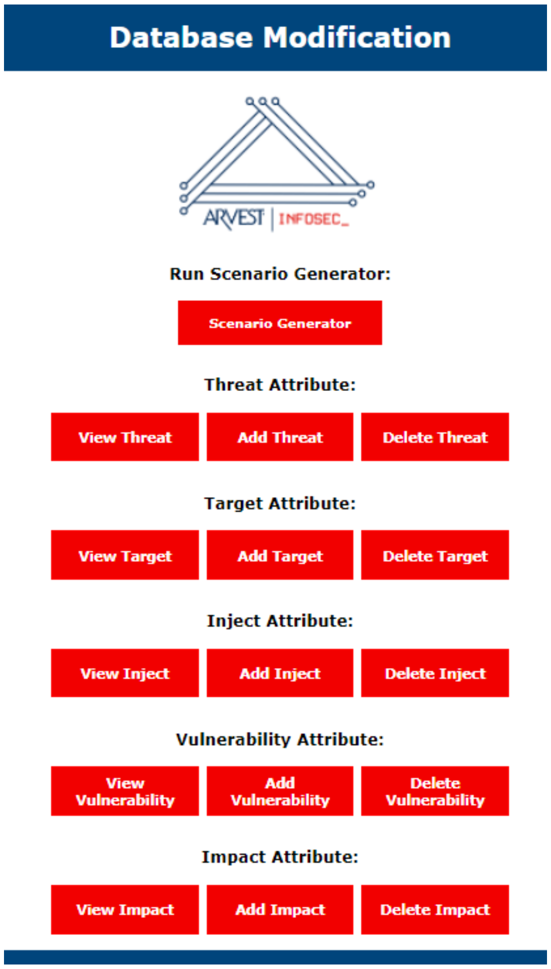
Our login page we kept straight forward and simple so that we can spend most of the time on the main chunk of the project which is the random scenario generator. However, the login page does a few things, such as verifying that the user is in the database, the user has the correct password attached to the username submitted, and that the role that they have is based on their username, so that the system knows what page to route them to, based on the role that they serve, which can be an employee or elevated user. Something to note about the login page is that it takes into consideration either the username or email that is attached to their account. Besides the login page we also have a sign-up page that per Arvest request is only allowed to be done by the administrator so that for the time being they don't get a flood of random sign ups that cannot be monitored. The sign-up page checks for empty inputs, strings of characters that do not form or have characters of a usual email address, and passwords that don't match from both inputs.

- Update user profile

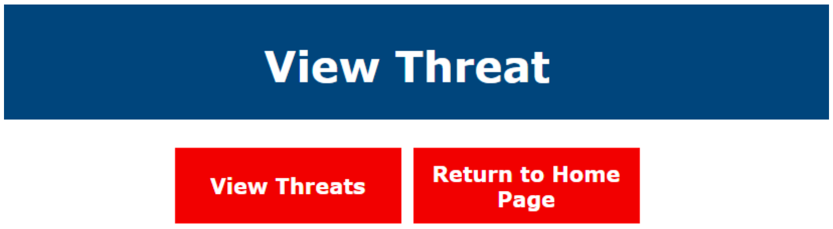
  We also implemented the ability to change and update a person's username and role on the elevated user profile. That allows them to change user's status as need be and utilize that mechanic to help with the flow of the application as it gets going and is used throughout Arvest's different sectors of the company.

- Database Modification Page

  The database modification page is a major point in the tool. As this tool is designed to be used by a variety of users with different levels of technical skills, we wanted to make sure anyone could easily edit the database. MySQL is something that as a CSCE student might not seem hard or difficult to use, but we didn't want to assume that everyone understood how to use it to manage a database, so we created a simple user interface that would gather information from the user and then add that information to the database. There are a variety of web pages that fall under the database modification page from the home page and the three subpages: view, add, delete for each of the five aspects of a scenario. The home page can be seen below.



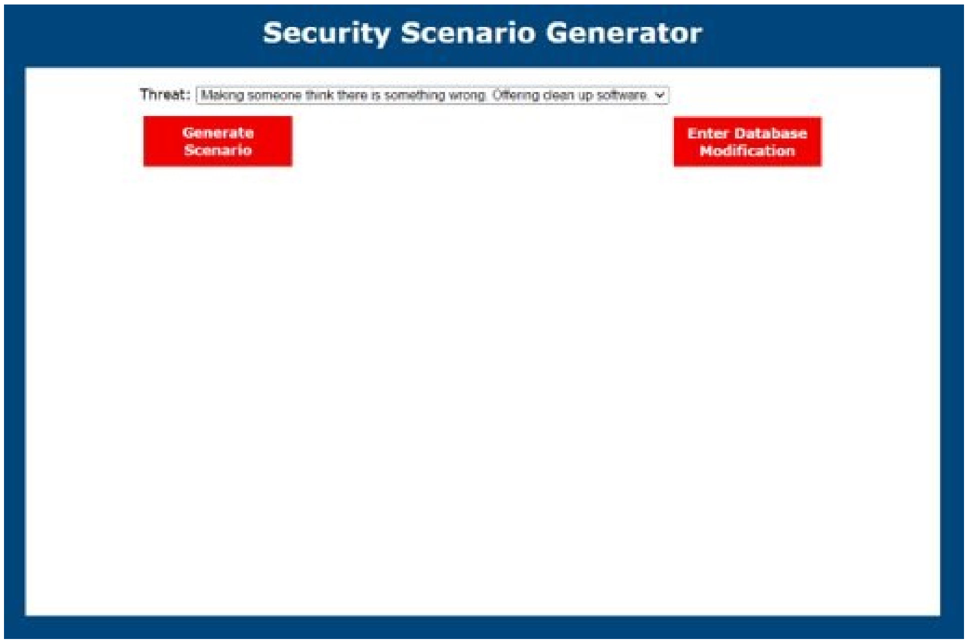
Here a user can pick what attributes they wish to edit or view and perform that action by going to a new page designed for that specific action. This page is here to show all possible actions that an elevated user could perform that would affect the database. We wanted to make sure the user could see everything that they could do and make it easy to access the subpages to ensure that any user could use our tool effectively. With this homepage, the user has three actions that they can perform on five different attributes. Each action takes the user to a subpage like this.



In this example of a subpage, the user could click the button to view all the threat attributes in the database currently and they would be displayed in the white area under the two buttons. The other subpages are very similar to this with some variety like drop-down menus for the delete pages and some of the add pages and some textboxes to get user input to add attributes to the database. All follow the simple design and blue and red color scheme. This was done so that the pages were simple and easy to interact with and the colors are those of Arvest. With the Arvest colors and information security logo, the user is reminded that this is an Arvest tool and that was something our partners at Arvest wanted us to implement.

- Scenario Generator Page

The scenario generator page allows us to generate scenarios, which will be used by Arvest to train employees on how to deal with various cyber exercises. First, the user will have to select a threat from the drop-down menu. After selecting a threat and clicking the generate scenario button, a random target, inject, vulnerability, and impact are selected based on the selected threat from the drop-down menu. All these attributes will represent a possible scenario that Arvest can face. The scenario generator page can be seen in the picture below.


Security Scenario Generator
Threat: Making someone think there is something wrong. Offering clean up software.
Generate Scenario
Enter Database Modification

4.3 Risks

Risk	Risk Reduction
User Access Control	Implement an access control system to protect the integrity and confidentiality of the system.
SQL Injection Attacks	Filter user input to reduce the ability to perform SQL injection attacks on the database.
Account Security	Make sure all passwords are encrypted and stored in the database
Confidentiality	Require a valid user log-in to access the site to keep records from Arvest Bank safe.

4.4 Tasks

1) Create Capstone Group Website
2) Create Capstone Individual Websites
3) Create group GitHub repository
4) Discuss Project Design
    - 4.1) Discuss Backend
        - 4.1.a) Discuss Database Schema/Structure
        - 4.1.b) Generate a list of cyber security scenarios
    - 4.2) Discuss Front End
        - 4.2.a) Talk to Arvest team to provide Arvest Colors & Logos
        - 4.2.b) Create product map for application (paths for site)
        - 4.2.c) Design web application layout
5) Create/Implement Project Aspects (Back End)
    - 5.1) Create a Database based on the database schema on Turing.
    - 5.2) Populate the database using information from the Cyber Security scenario list
    - 5.3) Identify the new host for Database.
    - 5.4) Change the host of the Database from Turing to an alternative host.
    - 5.5) Establish Connection Database and The Web Application
        - 5.5.a) Create Database API
        - 5.5.b) Create variables to hold information from each card type table.
        - 5.5.c) Establish connection between Database and DB API.
        - 5.5.d) Establish connection between DB API and Web Application

- 5.6) Database Features
  - 5.6.a) Create Tables to hold each card type.
  - 5.6.b) Create a Table that contains all scenario information.
  - 5.6.c) Create a Table to contain usernames and passwords.
  - 5.6.d) Encrypt & Decrypt Password information.

6) Create/Implement Project Aspects (Front End)
- 6.1) Create a basic HTML file with PHP extension to prompt the user to press a button.
- 6.2) Create a CSS file to format the html file based on the web application layout.
- 6.3) Create script files (JavaScript as an example) or import packages to support desired features the Web Application should perform.
- 6.4) Web Application Features
  - 6.4.a) Create navigational tabs to navigate the site between pages.
  - 6.4.b) Implement user authentication within the Login Page.
  - 6.4.c) Create Buttons to generate a scenario.
  - 6.4.d) Create a function that displays random attribute information from a database.
  - 6.4.e) Once Scenario is generated, prompt the user to either run a report on generated scenarios or to generate another scenario.
  - 6.4.f) Improve app readability on all Database Modification related pages
  - 6.4.g) Create Difficulty Levels when Generating Scenarios
- 6.5) Establish Connection to Database
  - 6.5.a) Display Table Information through the Web Application
  - 6.5.b) Display random attribute(s) from Database.
  - 6.5.c) To get user information and pass it to the DB API
  - 6.5.d) Ensure User Authentication Code utilizes User Table.

7) Train the Arvest Team to utilize the Web Application

8) Test the final product for any bugs

9) Preliminary Report & Presentation Slides

10) Final Report & Presentation Slides

4.5 Schedule

Task Number	Start Date	End Date
Task 1	Jan. 18th	Jan. 19th
Task 2	Jan. 20th	Jan. 22nd
Task 3	Jan. 20th	Jan. 24th
Task 4	Jan. 20th	Jan. 24th
Task 5	Jan. 20th	Feb. 25th
Task 6	Jan. 20th	Feb. 25th
Task 4.1.a	Jan. 20th	Jan. 24th
Task 5.1	Jan. 24th	Jan. 28th
Task 5.5.a	Jan. 24th	Jan. 30th
Task 5.5.c	Jan. 24th	Jan. 30th
Task 5.5.d	Jan. 24th	Jan. 30th
Task 5.6.a	Jan. 24th	Jan. 30th
Task 5.6.b	Jan. 24th	Jan. 30th
Task 5.6.c	Jan. 24th	Jan. 30th
Task 6.4.a	Jan. 24th	Jan. 30th
Task 6.5.c	Jan. 24th	Jan. 30th
Task 4.2.a	Jan. 28th	Feb. 4th
Task 4.2.b	Jan. 28th	Feb. 4th
Task 4.2.c	Jan. 28th	Feb. 4th
Task 4.1.b	Jan. 28th	Feb. 4th
Task 6.5.a	Feb. 1st	Feb. 4th
Task 5.2	Feb. 7th	Feb. 14th
Task 5.5.b	Feb. 7th	Feb. 14th
Task 6.1	Feb. 7th	Feb. 14th
Task 6.2	Feb. 7th	Feb. 14th
Task 6.3	Feb. 7th	Feb. 14th
Task 6.4.b	Feb. 14th	Apr. 4th
Task 6.4.c	Feb. 14th	Apr. 4th
Task 5.6.d	Feb. 14th	Apr. 4th

Task 6.4.d	Feb. 14th	Apr. 22nd
Task 9	Mar. 3rd	Mar. 10th
Task 6.5.b	Apr. 4th	Apr. 11th
Task 6.5.d	Apr. 4th	Apr. 11th
Task 6.4.e	Apr. 4th	Apr. 22nd
Task 10	Apr. 8th	Apr. 26th
Task 6.4.f	Apr. 11th	Apr. 18th
Task 8	Apr. 18th	Apr. 22nd
Task 7	Apr. 21st	Apr. 21st

4.6 Deliverables

- Database scheme: The DB schema contains the data for the five categories: Actor, Vulnerability, Target, Consequences, and Impact
- Web Application: The web application is for generating a scenario based on the options for the five categories in the database.
- Web site code: The HTML, PHP, and Python code that represents the inner workings of the application and its features.
- Design Document: The design document will provide an overview of the organization, project objective, potential impact of the project, topic difficulty, related work, approach, tasks, schedule, high-level design, and how well we understood the material.
- Final Report: The final report will give details on the completed design. This will include a lot of details, which will be descriptions of the high-level architecture, the interface, the implementation of the software, which includes the database, API, and web application, the lessons learned, and future impacts. This final report will also include documentation on how to use the tool. We want to include some tutorials for inexperienced users to understand how the tool works.

5.0 Key Personnel

Ashwin Prakash - Prakash is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms, Software Engineering, Database Management Systems, and Computer Networks. He is responsible for completing the Garmin exercise observation report and the final proposal deliverable. Other tasks will be delegated by December 2021.

Jacob Smith - Smith is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms, Database Management Systems, Computer Networks, and Software Engineering. He is responsible for completing the Garmin exercise observation report and the final proposal deliverable. Other tasks will be delegated by December 2021.

Mathew Mclendon – Mclendon is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms and Software Engineering. He is responsible for completing the Garmin exercise observation report and the final proposal deliverable. Other tasks will be delegated by December 2021.

Rudy Ramirez - Ramirez is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms and Software Engineering. He is responsible for completing the Garmin exercise observation report and the final proposal deliverable. Other tasks will be delegated by December 2021.

Zachary Chapman - Chapman is a senior Computer Science and Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Database Management Systems, Programming Paradigms, and Software Arvest Cyber Exercise Preliminary Proposal Engineering. He is responsible for completing the Garmin exercise observation report and the final proposal deliverable. Other tasks will be delegated by December 2021.

Beth Rye is the Security Awareness and Education Coordinator at Arvest Bank. She has been working with Arvest Bank for a little over 15 years. She is the team lead on the Arvest side and leads all meetings.

Hayley Clark is a Business Analyst at Arvest Bank. She has been working with Arvest Bank for 2 years now. She will be our main point of contact for this project.

Juan Osorio Alonso is the former IT Manager at Arvest Bank. He will be answering most of our technical questions related to the project. He has been working for Arvest for almost 3 years and worked at Walmart prior.

Kourtney Connel is the IT Director at Arvest Bank. She will be helping on the technical side of this project when she can. She has been working with Arvest for the last 2 years and worked at Walmart prior.

Tina Owens is a Business Continuity Analyst at Arvest Bank. She will be sitting in on the meetings and answering any questions she can. She has been with Arvest for a little over 5 years.

6.0 Facilities and Equipment

For this project, we will require access to a server like Turing for testing purposes and keeping the web application running. The students will only need computers and the ability to access Turing. No other facilities or equipment were necessary to complete this project.

7.0 References

[1] Varonis, https://www.varonis.com/blog/company-reputation-after-a-data-breach/

[2] Cybersecurity & Infrastructure Security Agency, https://www.cisa.gov/cisa-tabletop-exercises-packages