Requirements Analysis Document

Dynamic Resource Generation
CSCI 4712 Senior Capstone Project
Spring 2021
Augusta University
Augusta, GA

Team Members

Justin Deshong
Joseph Glass
Daniel Kim
Eric Swart
Jake Tuten

Table of Contents

1 INTRODUCTION

1.1 SCOPE OF SYSTEM

The Attack Monitoring Console (AMC) is a web-based system used to dynamically generate decoy resources in the event of a cyber-attack and monitor a potential attacker's actions as they interact with decoy resources. The goal is to distract attackers from targeting real company resources and gather intel about the attacker's actions in order to identify them and prevent future attacks.

The system supports a single actor, the IT Employee. The IT Employee represents the user who is interacting with the system to generate resources and monitor activity in the event of an ongoing attack. The IT Employee interacts with the system through a web browser.

The system includes the functionality to deploy decoy resources in the form of virtual machines. The IT Employee can maintain these resources from the AMC and terminate them at any point. The AMC keeps track of the attacker's actions in the decoy environment and displays them on the homepage. The IT Employee can save this information in the form of a log after an attack has commenced. Data stores are cloud hosted and accessed via an external database. Servers are remote and accessible via the internet.

2 REQUIREMENTS OF SYSTEM

2.1 FUNCTIONAL REQUIREMENTS

AdminAttack - The attacker logs into the Admin's workspace account and uses the command prompt to go through the Admin's file systems. The attacker can also use Remote Desktop Connection to log into another workspace account.

UnanticipatedAttack - The attacker logs into a user's workspace account and uses the command prompt to go through the user's file systems.

MonitorAttack – The ITEmployee looks at the attack data from the Monitoring Console and can make a decision based on that data.

SetUpEnvrionment – The ITEmployee can create a workspace and configure it using the settings of a user, image, and size.

SaveLog – The ITEmployee can save a .json file that contains all the attack data regarding the current attack.

TerminateEnvironment- The ITEmployee can terminate workspace environment. This causes the Attacker or any other user to be directed out of the workspace environment.

2.2 NON-FUNCTIONAL REQUIREMENTS

- Platform
    - Monitoring Console is web-based and should utilize a web browser
    - Monitoring Console should be implemented using the Microsoft .Net Framework
- Safety
    - Save Log .json file is under the auto-save feature so no data is lost during the event of an attack
- Usability
    - The Monitoring Console consists of buttons, data tables, and a menu bar to help the ITEmployee make decisions
- Performance
    - The Monitoring Console will receive keylog data from a workspace every 30 seconds

2.3 USE CASES

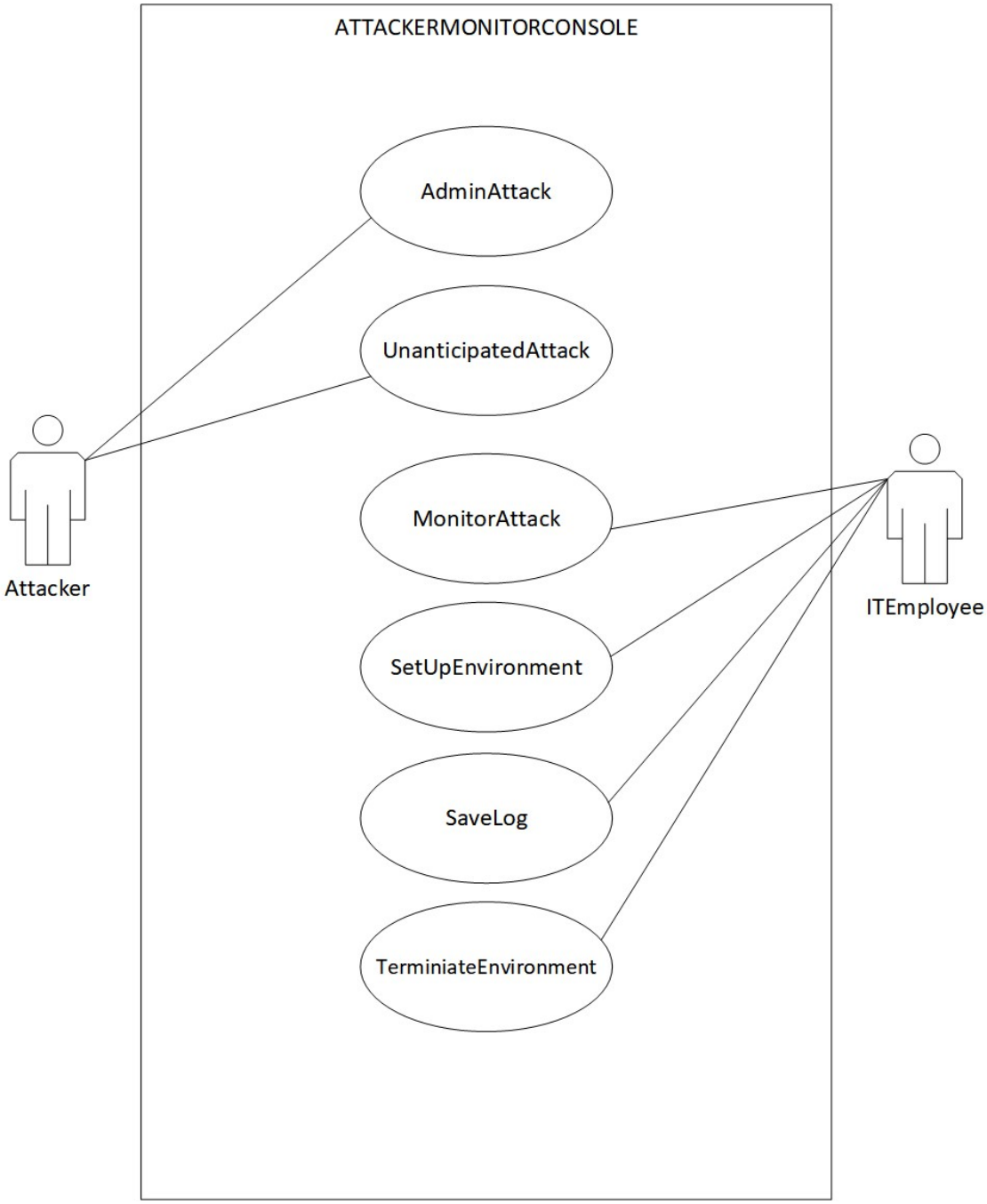


Figure 2.0: Use Case Diagram

2.4 USE CASE DESCRIPTIONS

Use case name	AdminAttack
Participating actors	Initiated by Attacker
Flow of events	1. The Attacker uses the credentials to log into the decoy Admin environment using the Command Line Interface. They begin traversing the decoy environment's file system looking for an active directory which contains login information for high level engineers of the company. **2. The AMC initiates the MonitorAttack use case**. 3. The Attacker finds and accesses the decoy active directory. From the directory, they are able to obtain a file that contains login credentials of the engineers. The Attacker uses Remote Desktop Protocol (RDP) connection to stay logged in to the Admin environment, while using the fake login credentials to log in to each of the decoy engineer environments. The Attacker traverses the file systems of each environment looking for top secret design documents. They are able to find several documents of interest regarding plane blueprints in the decoy file systems, which the Attacker then saves to their personal computer for later use. When they are done, the Attacker exits all environments.
Entry condition	1. Attacker has credentials for the fake environment.
Exit Condition	1. The Attacker leaves the environment.
Security Requirements	

Table 2.1: AdminAttack

Use case name	UnanticipatedAttack
Participating actors	Initiated by Attacker
Flow of events	1. The Attacker uses the decoy credentials to log into the fake environment using the Command Line Interface. **2. The AMC will initiate the MonitorAttack use case.** 3. The Attacker traverses the directory structure and begins looking for sensitive information about new cancer drugs . The Attacker finds a folder that includes information on all new cancer drugs coming out very soon and starts to open files. After finding some files that contain info about cancer drugs, the Attacker then downloads all the files in the folder to their personal PC. After downloading the cancer drug files, the Attacker starts to look for other vulnerable data to exploit. When they are done, the Attacker leaves the employee's environment.
Entry condition	1. An unanticipated attack has occurred on the company's systems. 2. ITEmployee has deployed a fake environment.
Exit Condition	1. The Attacker leaves the environment.
Security Requirements	

Table 2.2: UnanticipatedAttack

Use case name	SetUpEnvironment
Participating actors	Initiated by ITEmployee
Flow of events	1. The user selects the "Configure New Resource" radio button on the SetupEnvironmentForm. Beneath this button, the user enters the username of the spear-phished employee, specifies the account type that describes this employee, and enters a VM ID and expiration date. The user hits the "Continue" button. **2. The system destroys the SetupEnvironmentForm, and it generates and displays the CloneConfigForm.** 3. The user chooses a single option from a list of existing clone environments that correspond to the account type specified. **4. The system displays in a PopupMessage the cost score associated with using that clone.** 5. The user clicks the "Deploy" button on the bottom of the form. **6. The system configures and boots up a VM corresponding to the preferences the user selected. The CloneConfigForm is destroyed, and the system generates and displays the MonitoringForm.**
Entry condition	1. The ITEmployee learns of an imminent attack on a company employee; OR the ITEmployee has clicked the "Set Up Environment" button on the MonitoringForm.
Exit Condition	1. A VM is configured and booted up.
Security Requirements	

Table 2.3: SetupEnvironment

Use case name	MonitorAttack
Participating actors	Communicates with ITEmployee
Flow of events	**1. The AMC displays a notification message that an attack is occurring. The notification message consists of the directories, and other actions of the attacker. The following info will be displayed in the AMC: short description of the activity, a timestamp, the IP address of the Attacker, and the name of the account that the Attacker is trying to access. If the IP address identifies a previously attributed attacker, the system will display the attacker's information (name, location, affiliated group, tactics, & techniques). The system will update continuously based on the attacker's actions and behaviors inside of the environment (clicking on an application, opening/searching folders and documents, typing in the command prompt). The info mentioned here will also be updated in a MongoDB Atlas Database ('Auto-Save' every 10 seconds).**
Entry condition	1. An Attacker has accessed a decoy environment.
Exit Condition	
Security Requirements	

Table 2.4: MonitorAttack

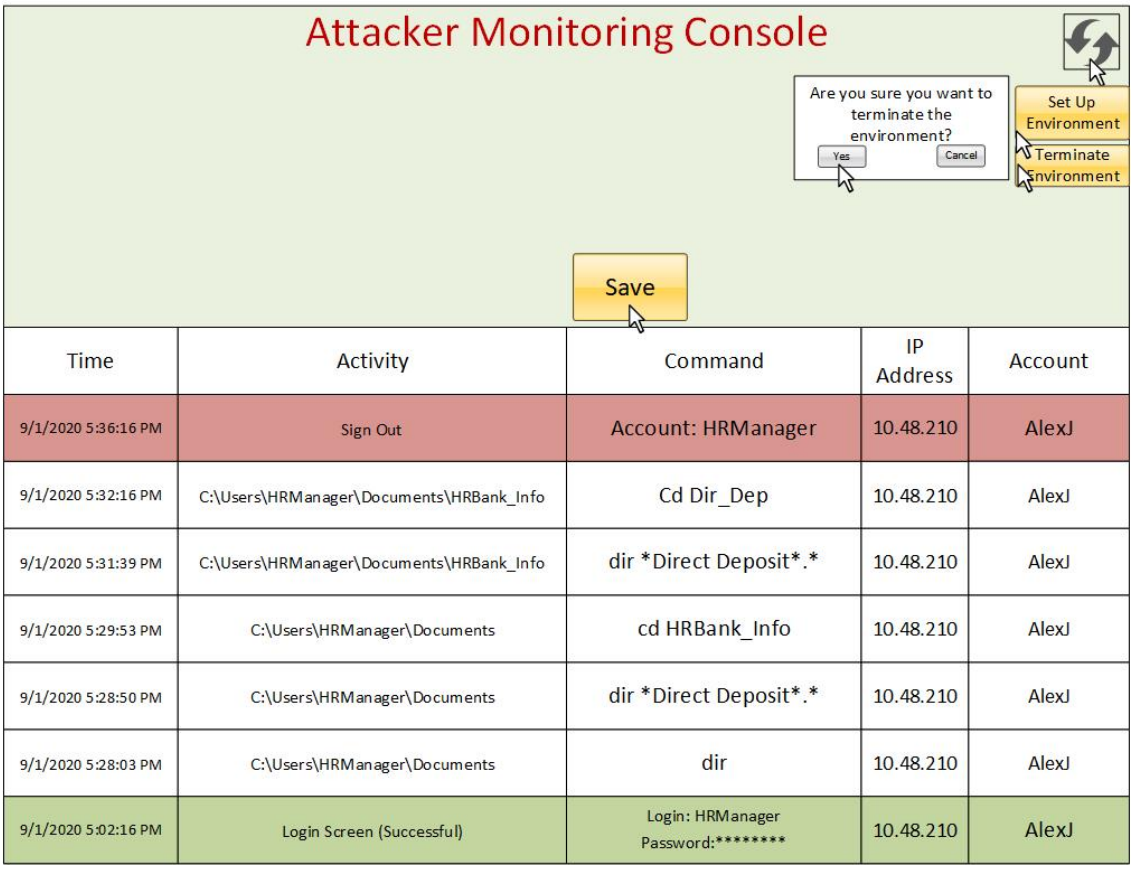
Use case name	SaveLog
Participating actors	Initiated by ITEmployee
Flow of events	1. This function is activated when the ITEmployee selects the "Save" button on the MonitoringForm. **2. In its database, the AMC stores the date of the attack, the attacker's IP address, and a string concatenating all the activities and commands of the attacker. For each VM deployed, the AMC also saves the username attached to the VM, the VM's total size (in GB), and the cost score associated with deploying it. A popup ConfirmationMessage informs the user of the successful transaction. The AMC also saves the information above in a json file (named by timestamp of current time). Since the AMC 'Auto-Saves' every 10 seconds, the json file will be updated every time by appending the information to the file.**
Entry condition	1. The attack is terminated.
Exit Condition	1. The attack data is saved to the database.
Security Requirements	Non-functional Requirement (Safety) is used here so try to prevent data from being lost after the attack has started.

Table 2.5: SaveLog

Use case name	TerminateEnvironment
Participating actors	Initiated by ITEmployee
Flow of events	1. With an active decoy environment, the ITEmployee clicks terminate environment on the MonitoringForm on the AMC. **2. The AMC responds by terminating the environment. The information from MonitorAttack will then be used to update the MongoDB Atlas database one more time before the environment is terminated.**
Entry condition	1. Decoy environment is active.
Exit Condition	1. The environment is terminated.
Security Requirements	

Table 2.6: TerminateEnvironment

2.5 USER INTERFACE MOCKUPS



Time	Activity	Command	IP Address	Account
9/1/2020 5:36:16 PM	Sign Out	Account: HRManager	10.48.210	AlexJ
9/1/2020 5:32:16 PM	C:\Users\HRManager\Documents\HRBank_Info	Cd Dir_Dep	10.48.210	AlexJ
9/1/2020 5:31:39 PM	C:\Users\HRManager\Documents\HRBank_Info	dir *Direct Deposit*.*	10.48.210	AlexJ
9/1/2020 5:29:53 PM	C:\Users\HRManager\Documents	cd HRBank_Info	10.48.210	AlexJ
9/1/2020 5:28:50 PM	C:\Users\HRManager\Documents	dir *Direct Deposit*.*	10.48.210	AlexJ
9/1/2020 5:28:03 PM	C:\Users\HRManager\Documents	dir	10.48.210	AlexJ
9/1/2020 5:02:16 PM	Login Screen (Successful)	Login: HRManager Password:********	10.48.210	AlexJ

Figure 2.8: Image of Attacker Monitoring Console

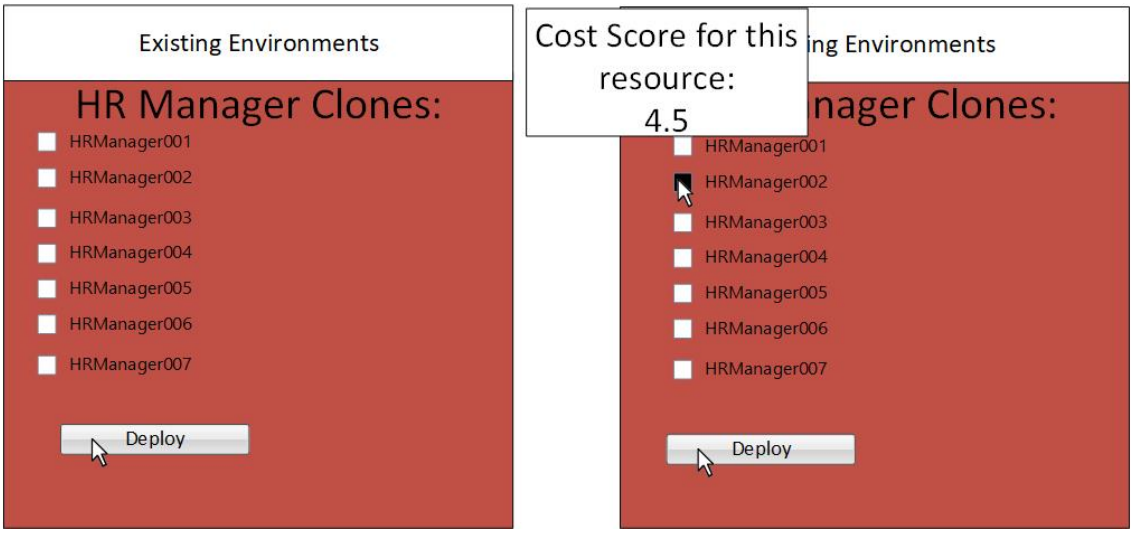


Figure 2.9: Images of Deploying Resources

Figure 2.10: Images of Deploying Resources

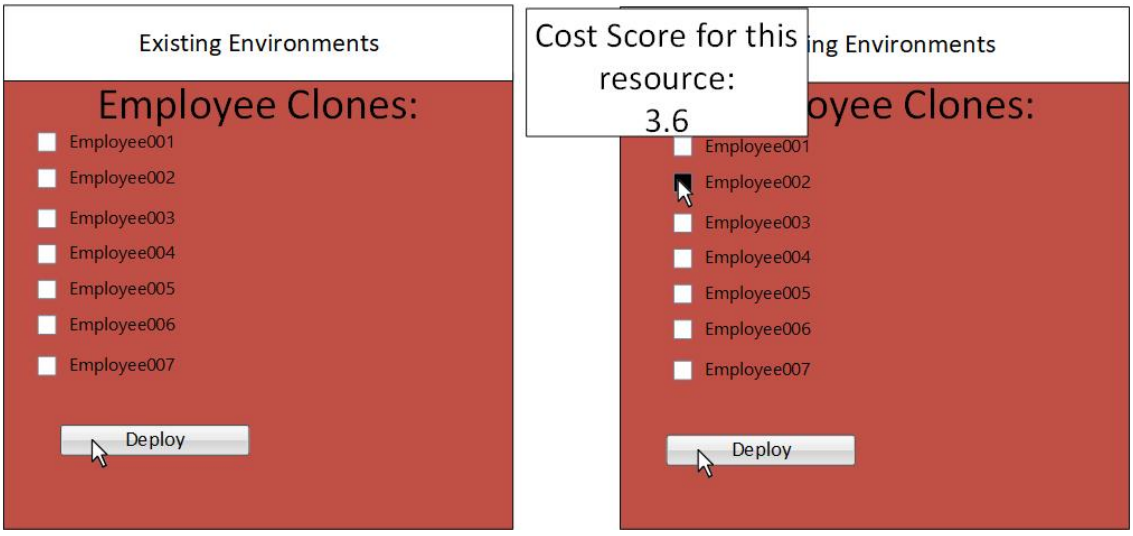


Figure 2.11: Images of Deploying Resources

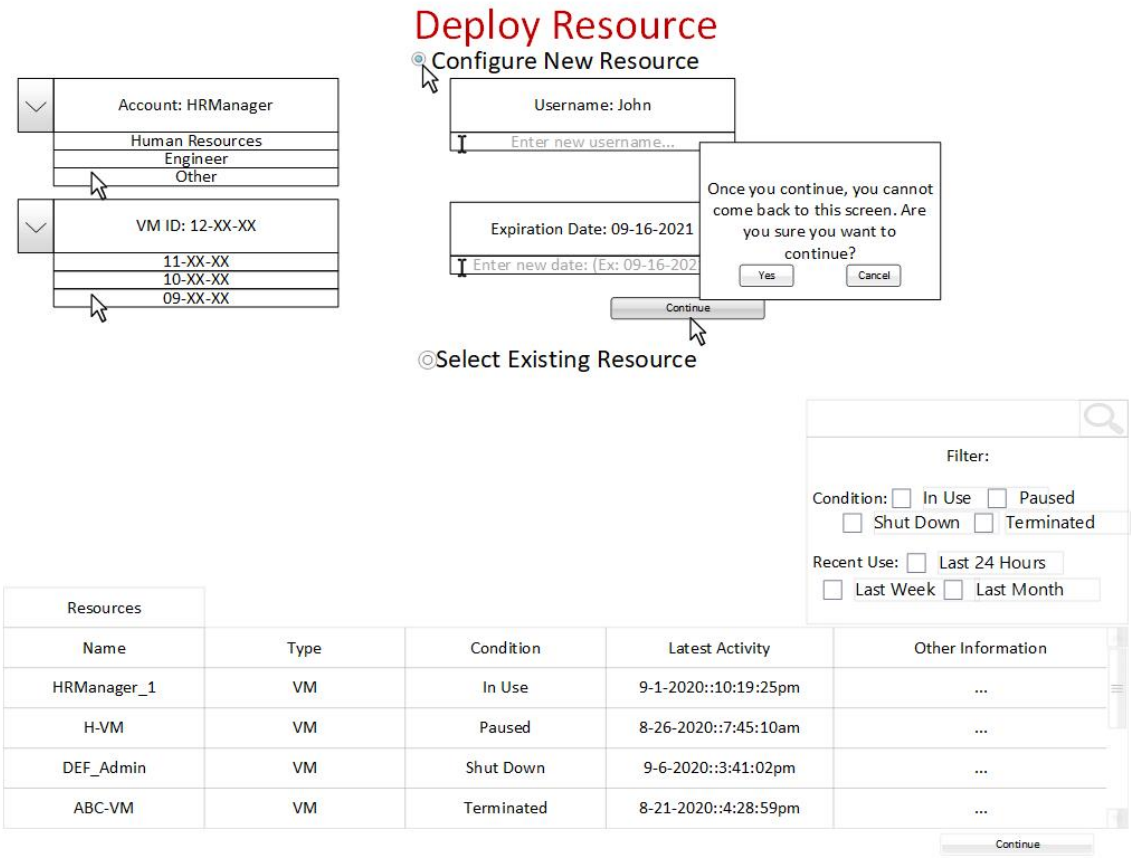


Figure 2.12: Image of Set Up Environment

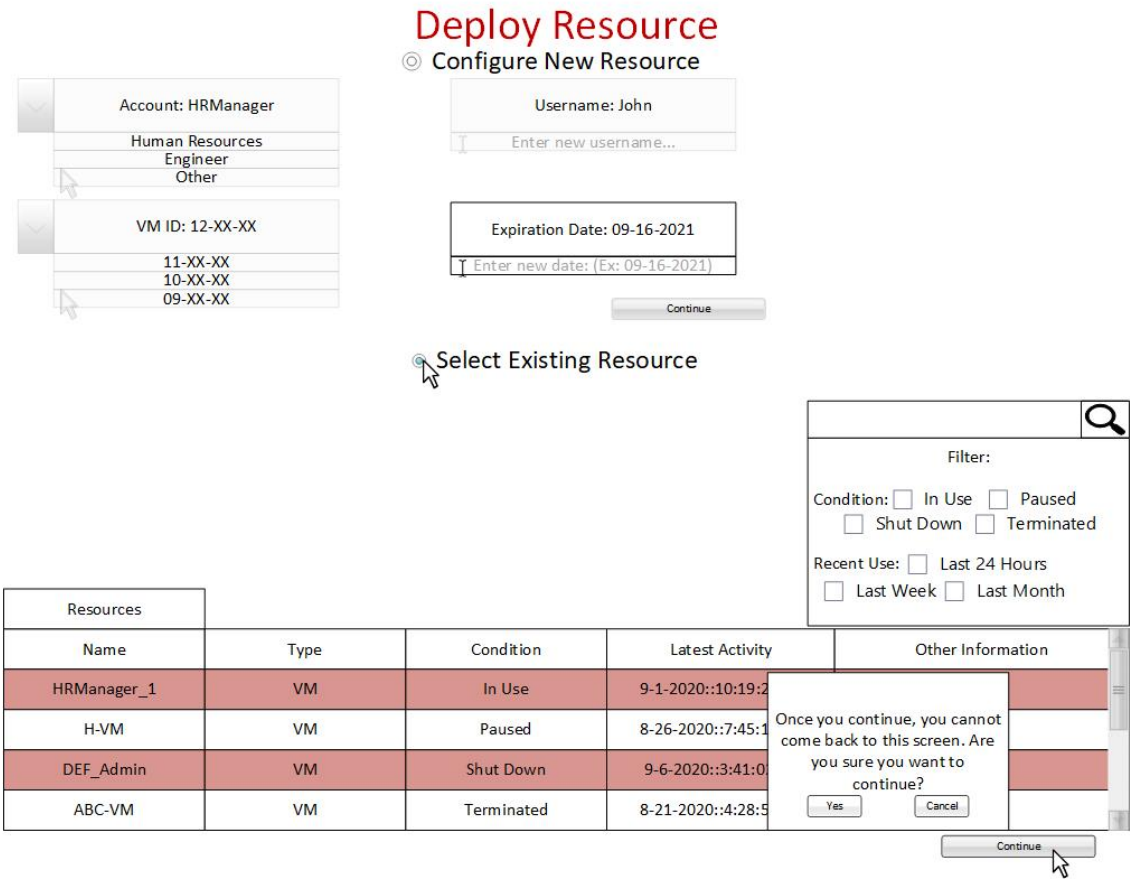


Figure 2.13: Image of Set Up Environment

3 SOFTWARE ARCHITECTURE

3.1 SUBSYSTEM DECOMPOSITION

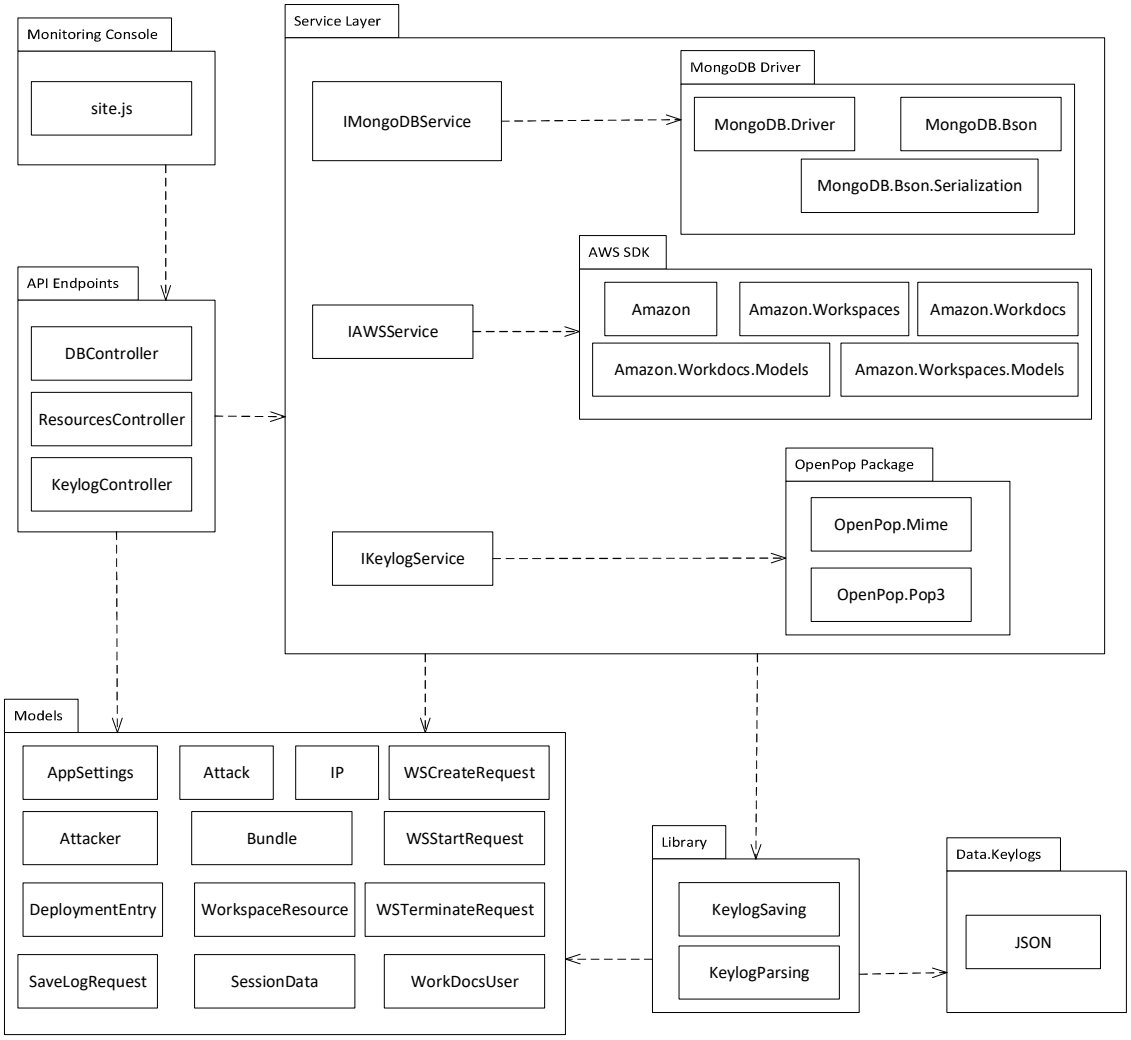


Figure 3.1: Image of Subsystem Decomposition

- Monitoring Console: This is the web page that is presented to the IT Employee.
    - site.js: this is the JavaScript script that is responsible for handling the requests and responses over http to the REST endpoints located on the project's web server.
- API Endpoints: This subsystem provides an RESTful interface to the client with three endpoints.
    - DBController: This controller calls the appropriate methods of the IMongoDBService interface to perform reads and writes to the MongoDB database.

    - ResourcesController: This controller calls the appropriate methods of the IAWSService interface to execute methods that describe the resources associated with the AWS configuration, and that create, start, stop, and terminate resources.

    - KeylogController: This controller calls the appropriate methods in the IKeylogService interface to read from the POP3 server associated with the project's configuration. This controller also handles refreshing the SessionData object at the start of a new attack.

- Service Layer: This subsystem defines three interfaces and contains three classes that realize each respective interface.

    - IMongoDBService: Classes implementing this interface call methods of the MongoDB Driver package to interact with the MongoDB database defined by the project configuration.

    - IAWSService: Classes implementing this interface call methods of the AWS SDK to elicit AWS services programmatically.

    - IKeylogService: Classes implementing this interface call methods of the OpenPop package to read from the POP3 server defined by the project's configuration. Such classes also are responsible for marking messages as deleted after they are read.

- Models: This subsystem defines entities custom to this system.

    - AppSettings: This class contains properties mapping to the keys given in the appsettings.json file. Components that depend on these settings can request an AppSettings instance via dependency injection.

    - Attack, Attacker, IP, WorkspaceResource: These classes represent objects to be stored in the database.

    - WSCreateRequest, WSStartRequest, WSTerminateRequest, SaveLogRequest: The first three classes contain the fields necessary to relay data necessary for operations on AWS resources from the client to the AWSService. SaveLogRequest performs a similar function between the client and the MongoDBService.

    - Bundle, DeploymentEntry, WorkDocsUser: These classes are the output of operations that describe AWS resources.

    - SessionData: This is a static class that contains data that persist between request to the server. Its fields include items necessary for the keylog parsing algorithm, such as a list of previously entered commands.

- Library: This subsystem defines two classes that encapsulate static methods.

    - KeylogSaving: This classes' methods are called by the DBController to save the keylogs locally on the same server that hosts the web app.

    - KeylogParsing: This classes' methods are called by the KeylogService to parse messages read from the POP3 account.

- Data: This subsystem contains the keylogs from all attacks, named by the timestamp of the starting time of the attack. These files, in json format, reside in the Keylog folder.

3.2 HARDWARE/SOFTWARE MAPPING

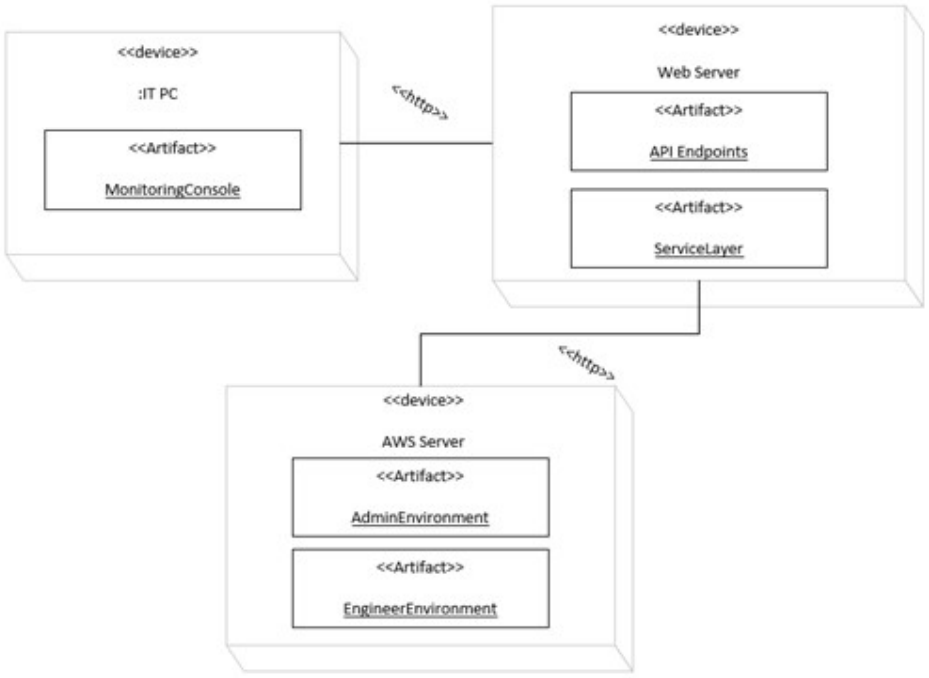


Figure 3.2: Image of Deployment Diagram

3.3 PERSISTENT DATA MANAGEMENT

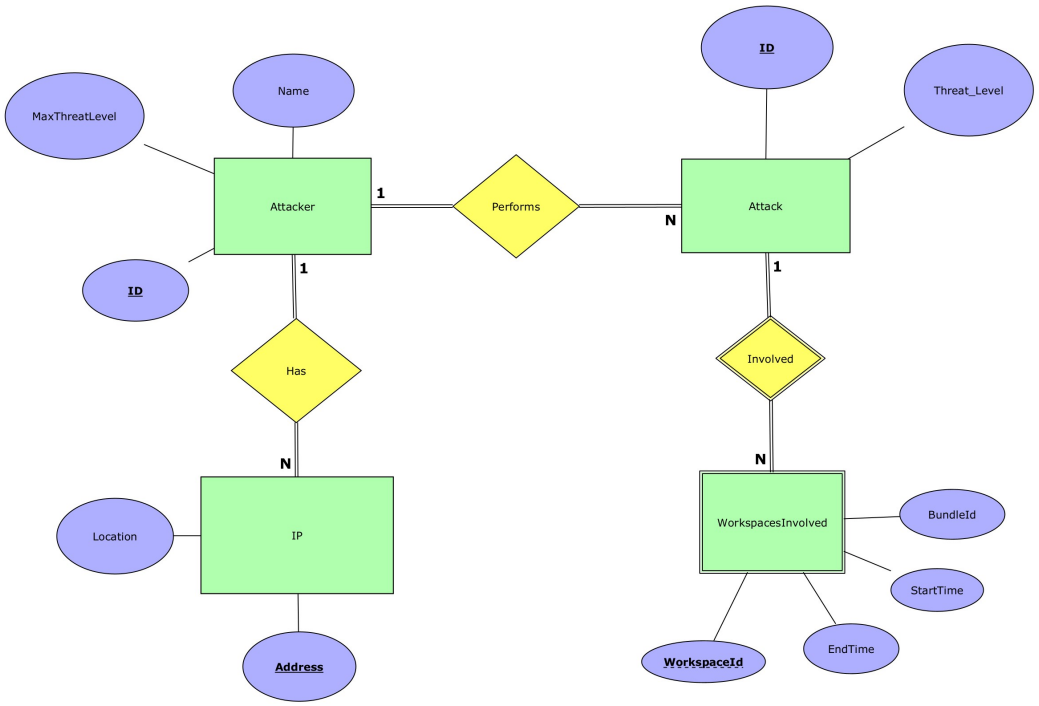


Figure 3.3: Image of ER Diagram

4 APPENDIX

4.1 APPENDIX A – SOURCE CODE

Link to GitHub repository: https://github.com/jdg1125/MonitoringConsole