4 Design

4.1 Design Content

The core of our design content will be two attack paths and the scenarios associated with them. The design elements of these attack paths will include the vulnerability or misconfiguration featured at each step of the attack and how these individual components will work together to mirror an AWS account compromise. After defining these steps, we will design scenarios or stories that mirror the users and services used at each step in the context of a hypothetical business that would be commonly found as the client of a security consulting firm, such as a hospital network or banking website.

4.2 Design Complexity

Components:

- **IAM:** complex relationships between users, resources, and services that will be defined through roles and policies
    - The modularity of our project through Cloud Formation Templates(CFT) enables it to include critical IAM vulnerabilities without jeopardizing company data. The CFT will be deployed on end user's AWS accounts which detaches our team's testing account from their own.
- **Red Team/Blue Team Operations:**
    - **Attack Paths:** Various vulnerabilities leading to unique points of entry, lateral movement, privilege escalation, and eventually to an end goal followed by unique exfiltration paths.
        - Ex: See 4.7.2 Design 1
    - **Incident Response:**
        - Creating robust logging systems. Ex: tracking Eventbridge API data with CloudTrail and viewing those data metrics in CloudWatch
        - Remediation of the vulnerabilities to effectively mitigate the attack path
- **Narrative/ User Roles:**
    - We've added another layer of complexity by defining a narrative for the project and specifying user roles, integrating real-world scenarios and business-related considerations. This entails domain-specific knowledge and creative problem-solving, including the implementation of IAM roles and permissions to mimic real-world access control
- **Risk Management:**
    - We are keenly aware of potential challenges in the project, as evidenced by our risk considerations and mitigation strategies. Successfully managing these risks requires strategic thinking and a deep understanding of AWS security policies, including IAM best practices and AWS Testing Policies.

4.3 Modern Engineering Tools

- S3 Static Website:

  A S3 bucket serves as a central resource for users to access relevant information. It will be accessed from many other resources and will serve as our storage resource for our project. It serves as an easily accessible reference for whatever purpose.

- EC2 workstation:

  EC2 instances can be used as developmental and testing environments. It might also be used to simulate certain scenarios and test our environment. Since EC2 instances provide a flexible and scalable environment, the team can create, configure, and experiment with different AWS scenarios.

- Lambda function:

  Lambda functions can be used to implement various components in our environment. They can also be configured to simulate specific behaviors and vulnerabilities. Lambda functions act as building blocks for AWS environments, allowing the team to manage all configurations and behaviors of resources.

- IAM role and policies:

  By controlling roles and policies, we can control access to different AWS resources. It could play a role in mimicking certain identity flaws. IAM roles and policies play an important role in defining and managing access control. This showcases real world security issues and helps users understand the importance of well defined roles and policies.

- AWS CLI:

  By using the AWS CLI we can interact with AWS services and resources programmatically. This can be useful to automatically set up environments to some configuration. The AWS CLI streamlines interactions with AWS resources.

- API Gateway:

  By using API Gateways we can expose AWS API vulnerabilities and misconfigurations since it acts as a front-end service that handles requests. This will play a role in triggering Lambda functions and other resources. API Gateway acts as the entry point for external users to access the AWS API, where vulnerabilities and misconfigurations can be deliberately incorporated.

- Gitlab:

  Gitlab is being used for project management and version control. It helps the team collaborate, manage tasks, and track progress throughout its entirety. The Agile-style project management in Gitlab ensures that tasks are well organized and tracked.

4.4 Design Context

Our project doesn't necessarily have a direct impact on **Public Health, Safety and welfare**. It's a tool that can be used to teach users how to exploit vulnerabilities and remediate these vulnerabilities in a controlled area. The entire purpose of this is to prepare the user for the real world in terms of cyber security, and in the case the user ends up working for a company that is in the medical field, government agency, security field or any correlating field, they now know how to find and mitigate these vulnerabilities. In doing so, they prevent attackers from gaining sensitive information, such as user and passwords, health information, government agency confidential data, the list is endless. That's the entire point of our project to inform users on how to prevent these leaks from happening, to prevent unwanted access, to prevent any data that can lead to harming people. For example, if a user utilized our project to get a good understanding of how to find vulnerabilities and mitigate them, and they work for a government agency that works in putting people in witness protection. This user's entire job is to work with the AWS services that hold data that leads to the location of these witnesses. If the user doesn't implement the right safeguards an intruder can potentially gain access to this data and sell it to anybody that is wanting to know details about these witnesses.

In terms of our project having an impact on **Global, cultural and societal**, there is not a direct impact. Looking at the impact indirectly, from a perspective of who can utilize our tool/project to help benefit themselves and their company, it would benefit those who have the privilege to access the internet, use of technology, etc. For example, If one country is not the most developed in terms of technology, it might not be as beneficial for them to use our project/tool when compared to another country that has to take into consideration the potential attacks on their cloud services, more specifically anything that is dealing with AWS.

While generally speaking our project won't have any major **environmental** impacts, the usage costs of the physical servers is a small consideration. Data Centers and cloud based providers use large amounts of local resources like power & water and while our project doesn't contribute to even 1% of that, it does subsist within these systems. This will indirectly cause more strain on power resources that, depending on where the physical server is located, could be non sustainable.

The **economic** impacts the project has, are one of the chief design considerations. Our design is intended to be economically viable for an individual within a team, and use as many free AWS resources that are relevant. This affordability aspect will be important in scaling the project up and making the development environment as accessible as possible. In general there won't be any broader economic impact and the only possible impact will be to clients using the product. In the long term, a small indirect economic factor might be increasing the knowledge and therefore value of potential security experts using our project.

4.5 Prior Work/Solutions

There are two prior solutions that we are using as learning guides and references. Flaws.cloud is a public website made with AWS which has purposeful misconfigurations and vulnerabilities. It is a good learning tool as it gives hints to help walk through how to find and exploit the misconfigurations on the site. IAM Vulnerable is similar but the user has to deploy the prebuilt environment in their own AWS account. IAM Vulnerable also has more content with 21 different attack paths. An advantage that Flaws has over the IAM Vulnerable is that it is already set up. This makes it a great beginner tool and is free to use compared to deploying the services in a personal account. On the other hand, this limits the type of vulnerabilities since

Flaws can't risk someone crashing the experience for everyone. This means certain misconfigurations and permissions can't be used in Flaws whereas IAM Vulnerable can do anything.

Our product is more similar to IAM Vulnerable than flaws. We will use Cloud Formation Templates so that it can be deployed to different AWS accounts. Additionally, we will design complete attack paths that flow together with the addition of a scenario to make it more life-like. We will also be able to implement any vulnerability, including write permissions since we don't have to worry about messing up the environment for others. One thing similar to Flaws, one of our requirements is to include documentation of how to solve and fix the issues in our attack paths. We are using the references also as ways to learn how to use AWS. Part of our weekly tasks has been going through the modules to learn the services and tools involved with AWS.

References:

IAM Vulnerable Modules: https://github.com/BishopFox/iam-vulnerable

Flaws: flaws.cloud

4.6 Design Decisions

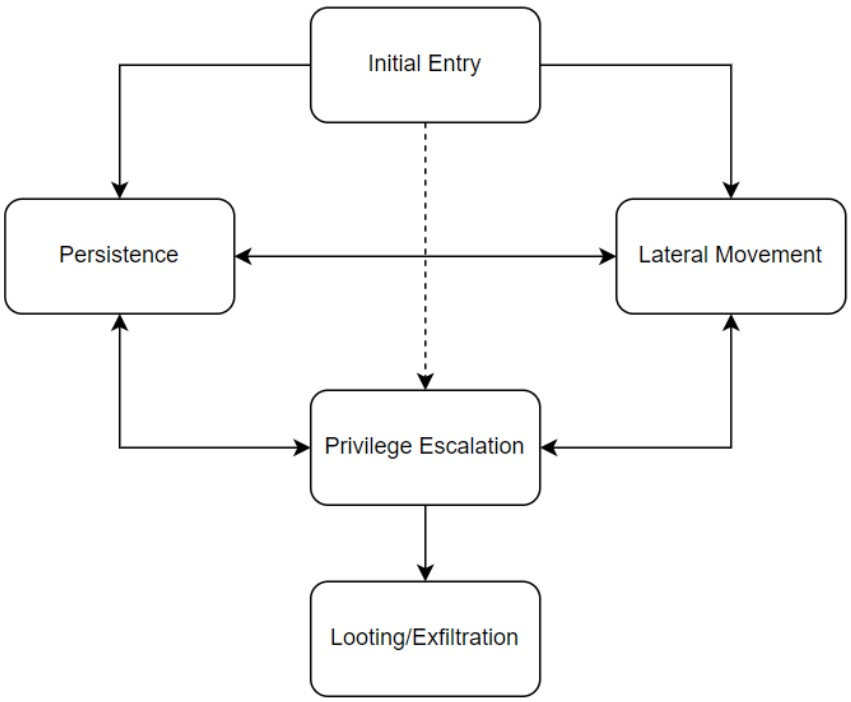
- One of our first major design decisions was to implement two distinct attack paths. These attack paths serve as the primary method by which clients will first interact with our project and allow for more versatility in the type of paths available. The decision to have two different paths was made to allow versatile options of attack, while being something that could be reasonably implemented in the time. The distinct separation of these resources also allows us to minimize resource usage because only one set of resources needs to be deployed at a time for the end user to make use of them, as opposed to all resources being deployed simultaneously.
- Another major design decision is to implement remediations for both attack paths. These remediations are to serve as guide on how to defend against these types of vulnerabilities from becoming an issue or being exploited. The decision to do this was to allow a user to understand different ways a vulnerability can be fixed.
- Another design component that we will be introducing is the logging of a user's actions. But we might not necessarily want to log every single thing the user does but rather specific actions that are geared toward the use of the attack paths or remediation efforts for each attack path. Doing this will enable the user to review what they have done to get an overall better understanding.

4.7 Proposed Design

4.7.1 Design 0 (Initial Design Template)

Design Visual and Description

This initial design iteration is a high level design that will serve as a template for future design iterations. Each component of this design represents an individual step in the generic flow of an attack on a system from the user's first interaction with the system and beyond the full system compromise. Each component's possible relationship and chronological order is represented by the arrows between each block in the visual element pictured below. These relationships are shown to be very complex here because this is a template which establishes the additional design choices that will be made in future iterations while also enabling us to provide an encompassing definition of each of these components before they are explicitly defined in terms of AWS resources.



Each component will be fulfilling at least one of the functional requirements of our project, with each requirement often being fulfilled simultaneously across multiple components. All of these components will consist of AWS Cloud Resources and associated IAM policies which will incorporate the various vulnerabilities and misconfigurations that will be used to simulate realistic production level AWS environments.

In future iterations, there will be multiple instances of components to represent multiple distinct attack paths. Each distinct attack path could also include multiple opportunities for initial entry, persistence, and other components. Each iteration may use the full template attack path or focus on certain parts leaving some components blank. At the end of the design phase, two final designs will be completed, each with a unique attack path. These final designs will be reviewed then have scenarios created for them, as defined in 4.1.

Functionality

Each component of this design iteration will function as a container to help distinguish the various AWS resources that will be used in the project in terms of their relation to the attack path.

Initial Entry:

In cyber security, Initial Entry is the method of gaining access to the system from the outside. When a user begins this educational module, they will be presented with a variety of available services that include points of 'Initial Entry'. Upon exploiting the misconfigurations included in this component, they will be presented with an opportunity to establish 'Persistence' within the environment and/or an opportunity to pivot to another resource through 'Lateral Movement'.

Lateral Movement:

Lateral movement involves pivoting to another machine or resource to obtain more information or move towards a goal. The 'Initial Entry' point will not be the final target of the attack path, thus the user has to move through the system towards other resources.

Persistence:

Persistence is the act of regaining access to the system without going through the 'Initial Entry'. The user will plant or create an entry point from one of the possible areas we incorporate. In many cases, the "Initial Entry" is a difficult process to complete and may lead to an unstable presence within the victim's system, so establishing persistence is important to ease re-entry and stabilize an attacker's foothold.

Privilege Escalation:

These paths can diverge and converge in a variety of ways that will be further defined in later iterations. The final goal of these components will be to enable further 'Privilege Escalation' which will enable larger system control, including access to hypothetical critical and confidential information that can be Looted or Exfiltrated from the system to demonstrate the severity of the risk.

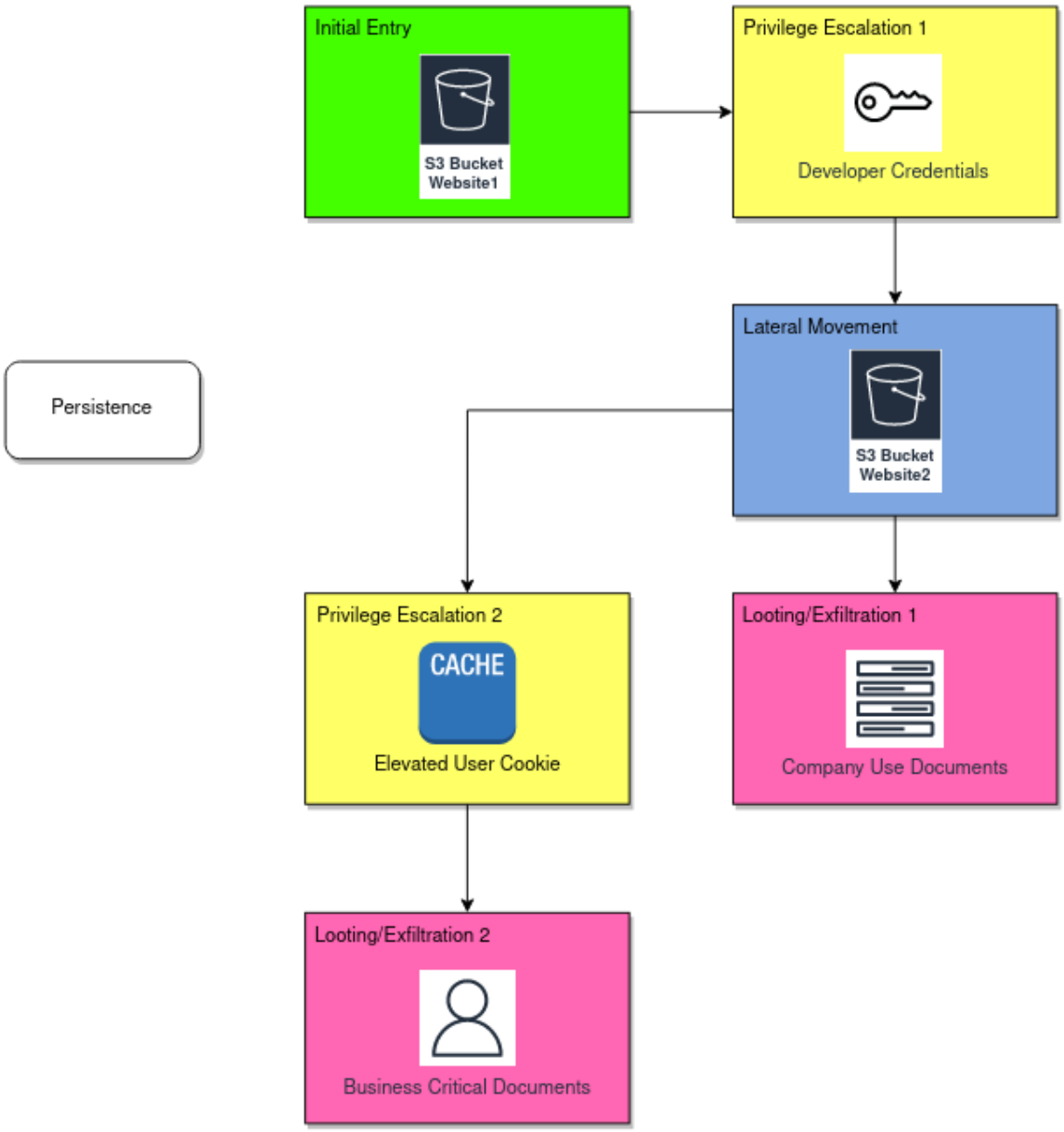
Looting/Exfiltration:

This final component of looting was not required, but we felt that adding this component would help drive home the serious nature of these seemingly innocuous vulnerabilities while giving the user an opportunity to classify and identify looted information.

4.7.2 Design 1 (Design Iteration)

Design Visual and Description

In this design iteration, we have defined a set of resources and vulnerabilities based on some of the exploits and misconfigurations we have investigated during our research. These resources have been represented inside of the core component types described in Design 0 to visually show their intended use. The relationships between each individual component type is strictly defined to represent how each resource intentionally enables the user to progress the attack.



Functionality

Initial Entry:

The initial entry component consists of an s3 Bucket, 'S3 Website1'. An s3 bucket is an AWS storage resource which will be used to host html files to be served via HTTP. This site will be **available to everyone** in this virtual environment, making it an ideal initial entry point for an attacker.

Privilege Escalation 1:

The vulnerability in this component is the misconfiguration of the S3 bucket. The misconfiguration is specifically the "GetObject" action which defines who can request what from the S3 bucket. This action will be **set to allow everyone** to access anything hosted on the bucket. An attacker can exploit this using the AWS CLI to reveal unintentionally exposed secrets. In this case the exposed secret is a set of developer credentials which can be used to access a restricted website, 'S3 Bucket Website2'. By obtaining the credentials of a privileged user, the attacker will have effectively elevated their privilege.

Persistence:

We will implement this in a future design iteration, our current plan is to add an additional privilege escalation component that will give the attacker an authorized AWS user with access to modify the Access Control List (ACL) of S3 Bucket Website 1. You can use an access control list to **allow access to S3 buckets from outside your own AWS account** without configuring an Identity-based or Resource-based IAM policy. By changing the Access Control list, the attacker will enable themselves and others to **read and write** restricted resources in the bucket even if their initial access is discovered and the compromised AWS user is removed.

Lateral Movement:

Using the **developer credentials** from 'S3 Website1', the attacker will **access an internal blog site**. The blog aspect was chosen to lay the groundwork for the privilege escalation and scenario in future iterations. The addition of this second website is to remove the components of this site from the affected areas of the vulnerability found in the initial entry section.

Looting/Exfiltration 1:

Using the developer credentials gained through 'Privilege Escalation 1', the attacker will be able to view a number of internal company documents that have been posted on 'S3 Bucket Website2'. These documents could point to other potential attack/access points, or even expose confidential client information.

Privilege Escalation 2:

On 'S3 Website2', there is an internal blog that allows privileged users to post messages on the site for others to view. This message intake will not have sufficient checks in place to prevent cross site scripting (XSS). By crafting a message using http, the attacker can make a post that will target other uses who access the site by having their browser access a url hosted by the attacker to capture the cookie being used by the victim. We plan on **enabling this attack through the use of a lambda** that will periodically make a simple GET request to this site using an admin user cookie.

Looting/Exfiltration 2:

After capturing the victim's cookie, the attacker can view the business documents of that user. This will grant the attacker access to non-public documents that contain sensitive business information such as employee records, budgets or technical data.

Changes

The additional content and resources found in this iteration arose from the research and experimentation that has been done by our team so far, which has been focused on S3 bucket deployment and misconfiguration. This design work has been accompanied by hands-on implementation of S3 buckets and accessing them through various means beyond those included in this design iteration.

NOTE: The following sections will be included in your final design document but do not need to be completed for the current assignment. They are included for your reference. If you have ideas for these sections, they can also be discussed with your TA and/or faculty adviser.

4.8 Technology Considerations

Highlight the strengths, weaknesses, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

AWS
CloudFormation template vs Terraform

4.9 Design Analysis

- Did your proposed design from 4.7 work? Why or why not?
- What are your observations, thoughts, and ideas to modify or iterate further over the design?