**CS 6348.001 - Data and Applications Security**

**Project:**

**ABAC for Role Activation Problem in Java Web App**

**Kunj Amrutbhai Patel**

**Table of Contents**

Problem Definition 2

Introduction 4

Security Principles 5

Security Layers 6

Technique Used

RBAC with ABAC 10

Output 13

Conclusion 21

**Problem Definition**

Web apps are computer programs that perform certain tasks directly via a web server, with the web browser serving as the web application's client. They are not the same as conventional desktop programs, which require software installation to run.

Web application protection, in essence, tackles questions concerning the security of web apps and utilities such as APIs and websites. They ensure that your information system is secure enough to protect valuable data and maintain operability.

Security is an important concern at all stages of the application development lifecycle, particularly when dealing with sensitive business data and resources. Implementing security strategies and checkpoints at the beginning of the software development lifecycle will help you improve and ensure safe web application development (SDLC).

When it comes to understanding the real potential of the internet, the introduction of web applications was a major advancement. They've grown over time and are now essential for companies of all sizes. As critical as these applications are to your enterprise, they also serve as entry points for threat actors. Here are three major reasons why commercial web application protection is critical:

**1. Prevents loss of sensitive data**

Cybercriminals are actively looking for confidential data to rob, networks to get into, and systems to hack. If these online apps are not rendered stable, cybercriminals will use them to steal confidential business information.

For example, malicious hackers targeted approximately 10 million websites in 2015. Organizations are vulnerable to cyber attacks 24 hours a day, seven days a week, and only about 7% of organizations perform security audits.

**2. Security is more than just testing**

Despite the fact that most vulnerability checks are automated, penetration testing methods are just as good as the individual using them. Penetration checks performed by penetration testers are linear, but confidentiality extends beyond research. As a result, web site authentication is no longer an optional security precaution.

**3. Secures business reputation and mitigates losses**

Today, there are over 1 billion websites in the world, and millions of people rely on search engines to find content for personal and commercial purposes. If your website is compromised, you will lose customer confidence, which will damage your company's credibility. Similarly, a hacked website with leaked data is likely to be blacklisted by search engines, rendering the products and services inaccessible to current and prospective consumers.

One security technique we can be used for role assignment and managing our system securely and effectively is RBAC. But it has one drawback-ROLE EXPLODING problem.

Number of roles increases as number of user increases.

Diagram

Description automatically generated

Chart

Description automatically generated with low confidence

**Introduction**

This problem of securing web application for complete security might sound like an impossible task as we may never predict Zero-day attacks but in the meantime what we can do is follow our security principles.

In this project we have focused on 2 security principles to secure our web application.

* Principle of Least Privilege.
* Defense in Depth.

While Implementing RBAC one major issue is its ROLE EXPLODING problem.

We tackle this problem by using ABAC functionality here. We use attribute to constrain under what conditions roles may be activated.

**Security Principles**

**The following security principles are used in our project**

**Principle of least privilege**

Text

Description automatically generated

**Defense in Depth**

Graphical user interface, text, application

Description automatically generated

**Security Layers**

**The following layers are used in our project**

Graphical user interface, text

Description automatically generated

A picture containing text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Technique Used**

**Overview**

**RBAC with ABAC**

Use attributes to constrain under what conditions roles may be activated.

As you can see by putting the constraint on user object itself and dynamically allocating it at runtime our power-user will effectively have to manage only few number of roles.

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Effectively Managing only 3 roles.

Table

Description automatically generated

**Web Application**

Fortress Realm

**Table

Description automatically generated**

**OUTPUT**

**User logging in**

**Graphical user interface, text, application, email

Description automatically generated**

**Add,update,delete and search button diabled as role is not activated. User clicks on page-1.**

**Graphical user interface, text, application, email

Description automatically generated**

**Activates role**

Graphical user interface, application

Description automatically generated

**Buttons are enabled as role is activated**

Graphical user interface, text, application

Description automatically generated

**Retrieves Information**

Graphical user interface

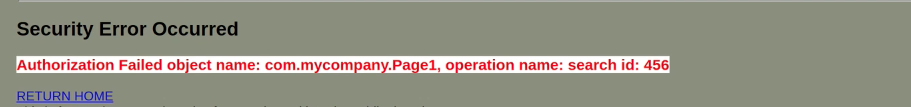
Description automatically generated

**Attempting to search data of user whose access in not provided.**

Graphical user interface

Description automatically generated

**Couldn’t Access data.**



**Attempting to activate role whose access in not provided.**

Graphical user interface, text, application

Description automatically generated

**Role assignment denied.**

Graphical user interface, application

Description automatically generated

**Attempting to search data of user whose access in not provided.**

Graphical user interface, text, application

Description automatically generated

**Attempting to activate 2 roles at a time.**

Graphical user interface, application

Description automatically generated

**Dynamic separation of duties- Can’t acticate 2 roles in one session.**

Graphical user interface, application

Description automatically generated

**Activates second role after deacticating 1st role.**

Graphical user interface, application

Description automatically generated

**Access is now permitted on this role**

Graphical user interface, text, application

Description automatically generated

**Defence In Depeth**

**Attack initiation point.**

Graphical user interface, application

Description automatically generated

**Session hijacked or attack directly access other page => *Access not permitted.***

Graphical user interface, application

Description automatically generated

**Conclusion**

Given the attacks on web applications in recent years, an increasing awareness of the limitations of earlier approaches to secure our system, and a growing realization of the power of new frameworks and libraries and related techniques to enable secure positive applications of system while mitigating negative impacts caused by attacks, the time is ripe for policymakers to work closely with researchers to jointly formulate recommendations for security and prevent maximum known attacks as possible.