UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Make today matter

Make today matter

COS 330 – Practical 2

RBAC + MFA

Due date: 3 September 2025 @ 11:00 am

Background

Software systems are typically developed with built-in access control mechanisms to restrict and manage what users can do within a system. However, these controls can be fairly limited if not structured around well-defined roles. Role-Based Access Control (RBAC) is a widely adopted security model that assigns permissions to roles rather than individual users. Users are then assigned to roles based on their responsibilities, ensuring that each user only has the necessary privileges to perform their duties. This not only simplifies management but also significantly reduces the risk of excessive or unnecessary permissions, which can be exploited by attackers. RBAC is foundational to enforcing the principle of least privilege, which is a cornerstone of secure software design. By restricting access based on roles, organizations can minimize the attack surface and better audit user actions. However, despite the robustness of RBAC, it is not a silver bullet. Attackers continue to evolve and employ sophisticated techniques such as privilege escalation, credential stuffing, and enumeration to find vulnerabilities and move laterally within systems. Privilege escalation, in particular, remains a critical threat, as attackers seek to elevate their access to administrator or root levels to gain complete control over systems.

To bolster RBAC and access controls, modern systems increasingly rely on Multi-Factor Authentication (MFA). MFA requires users to present two or more forms of verification before gaining access to a system—typically something they know (password), something they have (security token or mobile device), or something they are (biometric verification). This additional layer significantly reduces the likelihood of unauthorized access, especially in cases where credentials are stolen or compromised. MFA is now considered a baseline security measure for sensitive systems and is often mandated by security standards such as NIST 800-63 and compliance frameworks like HIPAA, GDPR, and PCI-DSS.

Instructions

For this practical, you have to implement the tasks below as well as write a report on your findings. Be sure to document all you have done, including screenshots as proof. You are restricted to using a NodeJS based backend or Django, and SQLite to perform these tasks and have free reign on MFA. You may use any front-end of your choosing if you are not using Django. You are not allowed to use 3rd party software (GitHub/Web) or AI generated code, you may use libraries provided by the programming language only.

Scenario

You have been employed by **SecureRUS** to securely develop a system that safeguards an organizations asset. They have 4 main roles, namely: Admin, User, Manager and Guest. They make use of folders to securely store data, and have 3 main assets, namely: Images, Documents and Confidential. The access rules are shown in the table below.

They require MFA to login to the system, and anyone is allowed to register, however, a manager / admin must approve the user first.

	Admin	Manager	User	Guest
Images	CRWD	CRWD	RW	R
Documents	CRWD	CRWD	RW	-
Confidential	CRWD	CRW	R	-

C – Create. R – Read. W – Write. D – Delete

Task 1 – [25 marks]

Create a registration page on the endpoint **/register** that allows a user to register, they should provide their name, surname, email address, and any other information you require. It should go without saying that there must be both client side and server-side validation. After verifying the information, you need to implement MFA. This should be done on the **/two_factor** endpoint. You will be evaluated on how secure setting up MFA is and which MFA you have chosen, as well as the whole registration process.

Next perform the login endpoint **/login** as well as the **/verify** which is used to verify MFA login. Next, an **/manage** endpoint should allow an admin to view new user requests as well as approve them, revoke any user as well as assign roles.

Note: An Admin or Manager must first approve the registration before the user can login.
Hint: Think like an attacker.

Task 2 – [35 marks]

For this task, you have to secure the digital assets as described in the Scenario. For confidential documents, they must be stored in an encrypted format on the disk, and accessible to only those authorized. Furthermore, every time an authorized party writes or creates a confidential file their identity must be validated and verified. To ease the implementation of this, there are 3 endpoints for the relevant files in which they must have an action field that indicates the action to perform (“**create**”, “**delete**”, “**write**”, “**read**”, “**list**”):

/images
/documents
/confidential

The confidential files are just text files with confidential information, and the UI should provide a means to edit the file, and no downloads should occur. For ***images*** and ***documents***, the “**read**” action can simply download the file, or alternatively allow the user to view on the page.

Hint: Think about MITM attacks.

Task 3 – [20 marks]

For this task you need to develop an **/analytics** endpoint that acts as a visualization platform to monitor all the endpoints in the form of a log. You can visualize this how you see fit. Write an algorithm that uses this log data to perform any anomaly detection, do not use AI, a simply rule based algorithm is sufficient, it also only needs to identify one item, but more items will earn you bonus marks.

Bonus – [2 marks]

Any additional or out of the ordinary security features / anomaly detection performed.

Submission

In your report, include screenshots, of positive and negative situations to demonstrate your system functionality. That is a standard user trying to access confidential files, and a manager accessing confidential files. Also include a screenshot of your algorithm code from Task 3. Furthermore, **record a short video** demonstrating your practical, it should be at most 5 minutes long and should have a file size less than 100MB. If you exceed this, use Handbrake to scale down or compress the video, as a last resort you can upload it to YouTube as an unlisted video before the due date. Uploading to Google Drive or a link will not be accepted. Uploading to YouTube will attract a 10%-mark penalty.

Submit your report as a PDF file along with your code and instructions how to run it in an archive called uXXXXXX_NAME_P2.zip to ClickUP before the deadline. Please upload well in advance, as there will be **ABSOLUTELY NO LATE SUBMISSIONS!!!**.

TOTAL: 80