International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

Strengthening Cybersecurity in Uganda's Electoral Commission through Multi-Factor Authentication and Single Sign-on Solutions across Organizational Applications

Carolyn Nasimolo, Associate Professor Dr.S.R.Raja
Department of Computer Applications,
Hindustan Institute of Technology and Science, Padur,Chennai. India.

Abstract- The increasing reliance on digital platforms for electoral processes in Uganda has brought to light significant cybersecurity issues that the Uganda Electoral Commission faces and this has made it imperative for the UEC to prioritize the strengthening of its cybersecurity. The Uganda Electoral Commission relies solely on traditional password-based authentication but hacking technologies have become more advanced and diversified. As a result, for security and authentication organizations are unable to rely on user ID and password-based authentication. (Hong, 2011). This single- factor authentication has been found to be vulnerable to attacks like malware, brute force, dictionary attacks, shoulder surfing, replay and phishing attacks etc. Much as passwords can easily be memorized and users at no cost are able to use them in their daily lives, these can be forgotten especially if the users have to log into multiple systems. (Mohammadreza Hazhirpasand Barkadehi, 2018). The UEC users log in to each of the systems independently which could lead to password fatigue. Cyberattacks can lead to unauthorized access to sensitive data, and data breaches and can therefore undermine the entire electoral process if the integrity of electoral data is compromised. This can lead to public distrust which could pose a threat to national stability. Therefore, by integrating MFA the UEC protects its sensitive electoral data and ensure secure access to its applications. Single sign-on is the ability for a user to authenticate once and access other protected resources where he has permissionwithoutlogging re- authentication. This system not only secures sensitive data but streamlines user experience through Single Sign-on (SSO) enabling users to log on once and gain access to multiple applications seamlessly.

Index Terms- Multifactor Authentication (MFA), Uganda Electoral Commission (UEC), Single sign-on (SSO), Single factor Authentication

I. INTRODUCTION

The Uganda Electoral Commission (UEC) is an autonomous constitutional body established under article 60 of the constitution of Uganda. Its primary responsibility is to organize and conducting regular free and fair elections as well as referenda in a professional, impartial and efficient manner. The EC's mandate includes ensuring the integrity, transparency and security of the electoral process which is critical in maintaining public trust and democratic governance.

In carrying out its mandate, the UEC utilizes a range of information systems and applications to manage the electoral processes. To log into these systems and applications, the users require a username and password only. This exposes the UEC systems to password related attacks like phishing, credential theft etc. Also, each application requires a separate login credential, making it cumbersome for users and increasing the potential for password related vulnerabilities because many employees may lack awareness of the latest cybersecurity threats and best practices which may lead to potential human errors that could compromise security.

Authentication is the process of identifying a user based on something he knows, has or something the user has or something that the user is (Hong, 2011).

Multifactor authentication provides layers of security that the traditional password systems lack. MFA requires the user to present multiple forms of verification before getting access significantly reducing the likelihood of breaches that may result from stolen credentials. And single sign-on (SSO)

makes it less cumbersome for the users to work seamlessly unlike the current fragmented systems.

This system is motivated by the need to protect the electoral integrity of Uganda and build public trust by modernizing the authentication framework through Multifactor authentication and SSO to safeguard the EC's sensitive data and streamline access management.

Single sign-on (SSO) works on a centralizes service concept where users are authenticated using one single set of log in credentials to gain access to more than one application (Nisha, 2020)

SSO is a mechanism that uses a single action of authentication to permit an authorized user to access all related, but independent software systems or applications without being prompted to log in again at each of them during a particular session. It reduces the risk for the administrators to manage users centrally, increases user productivity by allowing mobility and allows users to access multiple services or applications after being authenticated just once. This creates a more secure, user friendly and efficient environment that enhances productivity.

Implementing Multifactor Authentication (MFA and Single Sign On (SSO) solutions will strengthen the cybersecurity landscape of the Electoral commission. The primary objective is to protect sensitive electoral data and ensure secure access to organizational applications through the use of multi-factor authenticating to prevent unauthorized access, safeguard sensitive transactions, mitigate password related risks and improve user experience. By use of Multifactor authentication, security is enhanced by requiring users to verify their identity through multiple factors such as password combined with a One Time Password sent to their emails.

Through single sign-on (SSO) a user logs into multiple applications using a single set of credentials. This enhances security and user experience by reducing the burden of managing numerous different passwords.

II. SYSTEM ANALYSIS

In order to fulfil its mandate to maintain integrity, authenticity, confidentiality and availability of voter data, the Uganda Electoral Commission deploys various systems, tools and platforms to enable it perform these functions. These systems/applications rely exclusively on username and password to log in for user authentication which is not sufficient to safeguard the critical assets of the organization due to its tedious nature and is prone to password attacks.

The UEC users were required to login separately to each application or service that they need to access with individual credentials for each application. These separate authentication mechanisms for each system result in a fragmented user experience and can cause operational inefficiencies.

Proposed System

Multifactor Authentication (MFA)

MFA introduces an extra layer of security across EC's organizational applications with users required to enter their credentials (username and password), then system checks the credentials against stored data and generates a one- time multifactor authentication token that is sent to the user's registered email address. This token is generated by the database and when the user enters the token he/she has received in their email; it is compared to see if it matches the one generated in the database. The user is then authenticated and depending on the user's roles, a dashboard is displayed with links to the different applications that he/she is permitted to use. The token that was created in the database and is entered by the user is then discarded.

Single Sign-on

The SSO seeks to streamline the login process, enabling users access multiple applications using a single set of credentials which will reduce the risk of password reuse and log in fatigue. On registration of the user, the Administrator captures the names and URL's of the applications in the applications table.

He/she assigns rights to the user to the applications that they are permitted to access. Each use then enters their username and password for each application. There's an encrypt key created by crypto to encrypt the user's passwords in the User Applications table. When the user clicks on link on the dashboard and selects an application that they want to access, the system goes to the User Applications table and picks the name and password. It then decrypts the password using the EncrypKey saved in the database to plain text and then uses that password to log in to the respective application server.

This simplifies the user experience, enhances security, improves compliance and operational efficiency.

III. FEASIBILITY

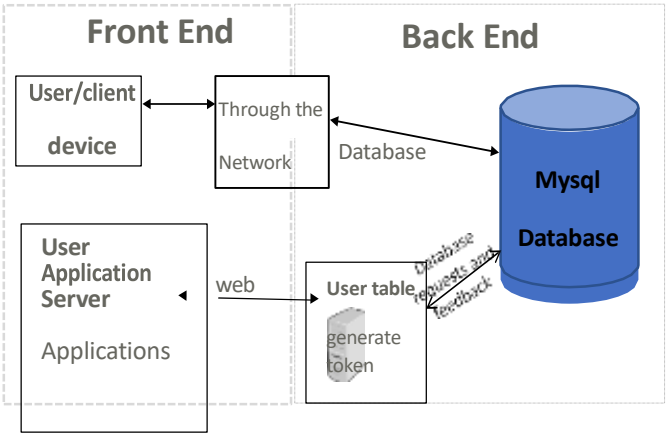
The proposed solution is technically, operationally and economically feasible with existing infrastructure Technically the system is feasible as it utilizes well established technologies and best practices recommended for cybersecurity and user convenience. It integrates what the user knows (username and password) and what they have (security token). The MFA solution will utilize secure communication protocols already in place such as TLS/SSL for transmitting tokens and is scalable to accommodate as many users as possible. This MFA leverage existing technologies such as

JWT and bcrypt which ensure compatibility with current infrastructure and crypto for SSO.

The system's operational feasibility enhances security and simplifies user experience while ensuring user acceptance and efficiency.

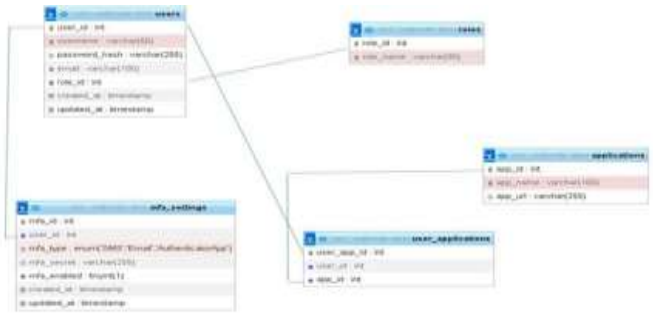
Economically, the solution is cost-effective because it's emphasis is more on software solutions with minimal additional hardware requirements.

System Architecture



The system architecture integrates Multi factor Authentication and Single Sign-on components with existing applications. It includes user management, secure communication channels, database server with user details, authentication servers.16

Entity Relationship (ER) Diagram
The ER diagram represents the relationships between users, roles, and MFA secrets.

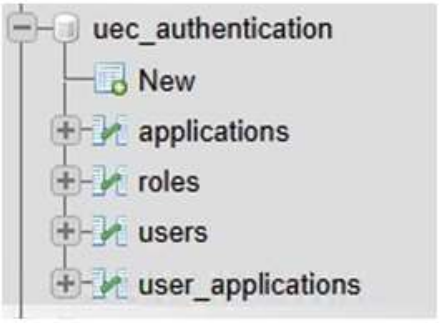


3.4 Database Design

The database tables are created using MySQL as seen below:
Creation of the database

Syntax
CREATE DATABASE uec_authentication; USE uec_authentication;

The Database has tables 4 main tables as seen in the figure below:

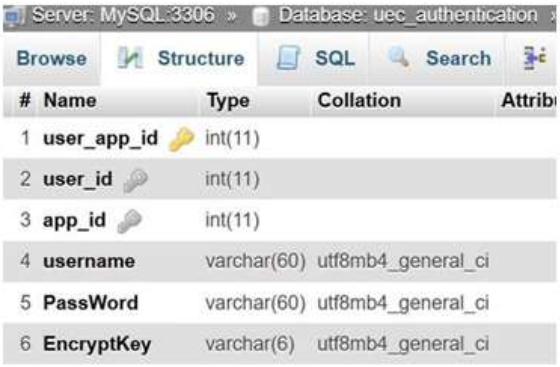


The Users database
Stores user credentials, user profiles and authentication factors for MFA.

Roles Table
Defines user roles and permissions. This table stores different roles within the UEC system, such as Admin, Staff, Observer, and Media. The roles are inserted into the by the administrator.

Applications Table
Contains all the applications in the UEC and their URLs or links.

User_applications table
Serves as an intermediary between the users and applications validating keys from the user before granting them access to applications. Stores the encryptkey which is used to gain access to the resprctive applications.



System Requirements
Software

- **Email & Messaging:** nodemailer (sending emails).
- **Database:** MySql for storing user information and authentication logs. This is platform independent and can be used on Windows, Linux server and is compatible with various Hardware.
- **Environment Management:** dotenv (loading environment variables).
- **Crypto:** password encryption and decryption while pairing the users and the applications.
- **Development Tools:** nodemon (auto- reloading server during development).
- **Authentication & Security:** bcrypt, jsonwebtoken, speakeasy (password hashing, JWTs, two-factor authentication) for secure token exchange.
- **Web Framework & HTTP Handling:** express, body parser, express-session, ejs (web routing, parsing request data, rendering views, session management).
- **Web browsers:** Compatible with Google chrome, Mozilla Firefox, Opera and Microsoft edge

Hardware

- **Database Server:** Dedicated authentication servers with multi- core processors, 16 GB RAM, 1 TB Hard Drive, Linux operating System. This computer is used for the deployment of MYSQL database, cryptographic algorithm operations and concurrent requests. (This stores user data and configuration files and logs.
- **Load Balancer:** Dual-core processor, 4GB RAM, 100 GB storage.
- **Client Devices:** desktops, laptops, and company tabs or phones.
- **Network:** Reliable network connectivity with sufficient bandwidth to handle authentication requests efficiently.

Firewall intrusion detection systems (IDS) to protect against potential threats and unauthorized access.

Security Considerations
Deployment of security best practices like the use of encrypted communication channels using HTTPS and TLS between the MFA server and the client are used. Also use of private keys between the single sign-on and the application server.

Regular updates and patching of the operating systems, software dependencies and the MFA server software will reduce security vulnerabilities. Also, implementation of access controls and logging mechanisms to audit and monitor access to the MFA server are deployed.

Implementation of regular backup mechanisms, response plans for handling security incidents and putting in place password recovery plans to ensure users can regain access to applications in case of a security incident.

Testing and Implementation
System Testing
Unit Testing: Validating each unit of the system to ensure that it works well and does what it is meant to do.

Integration Testing: This follows unit testing and verifies that all integrated components interact well together. For instance, testing how well authentication tokens are issued and validated across different applications. Also how the identity provider communicates with the application gateways and the use database. It is advisable to try out different scenarios at this stage like token expiry, invalid credentials etc.

Performance Testing: The system is tested for its stability, responsiveness, load time and maximum requests per second.

Implementation
The main challenge was intergration with integration with already existing systems since every user has to be enrolled and also resistance to change from users. The organisation therefore need to carry out continuous extensive training sessions to ensure that users appreciate the benefits of the new system.

Roll-out was phased to mitigate risks allowing for gradual adaptation and immediate troubleshooting of issues as they came up.

By methodically following the testing processes and addressing implementation challenges, organisations have successfully integrated MFA and SSO to enhance their security posture while maintaining user satisfaction.

IV RESULTS AND CONCLUSION

User experience has greatly with the adoption of SSO. Interviews conducted revealed that approximately 75% found the process of logging into multiple applications using a single set of credentials was more efficient and less frustrating than other traditional methods of logging into each application separately.

The use of SSO and MFA has led to significant improvements in both cybersecurity metrics and user experience. The systems administrators reported a 60% reduction in unauthorized access attempts post MFA deployment.

The use of MFA and SSO complies with regulatory standards which increases the organization's adherence to data protection regulations which increases public trust in the Electoral

Commission.
By embracing MFA and SSO the UEC not only safeguards its sensitive information but also empowers users. Ensuring a resilient security posture in a world of complex dynamic digital landscape of cyber threats.

Future Enhancement
As the Electoral commission strives to enhance its Cybersecurity, it should continue with training and awareness programs to improve the cyber hygiene of its staff.

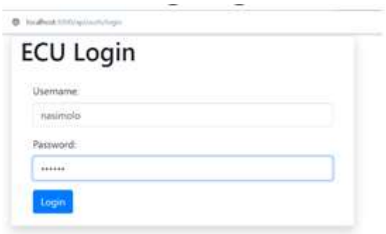
Incorporating another authentication factor like biometric (facial recognition, fingerprint, iris or hand geometry) can be considered to make the authentication process even more secure.

Cybercriminals are increasingly employing sophisticated methods such as deep fakes and automated attacks to bypass traditional security measures, enhancements in MFA and SSO should embrace advanced threat detection capabilities that can quickly identify and respond to anomalies.

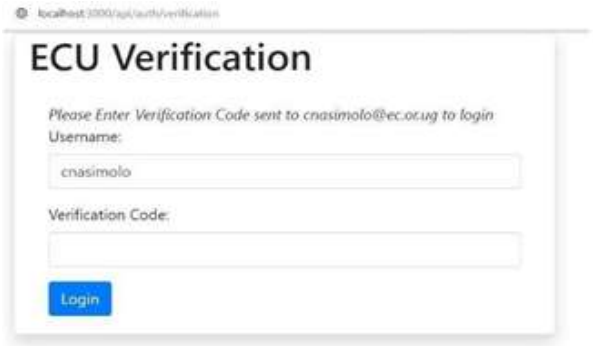
Through use of machine learning, systems can continuously assess risk and authenticate users based on their behaviour and reduce unauthorized access.

Also, through adaptive authentication, contextual information such as location, device and time of access can be used to make authentication decisions. For example, a user attempts to log in from an unusual location or device, he/she might be prompted for additional authentication factors and thus reducing the risk profile.

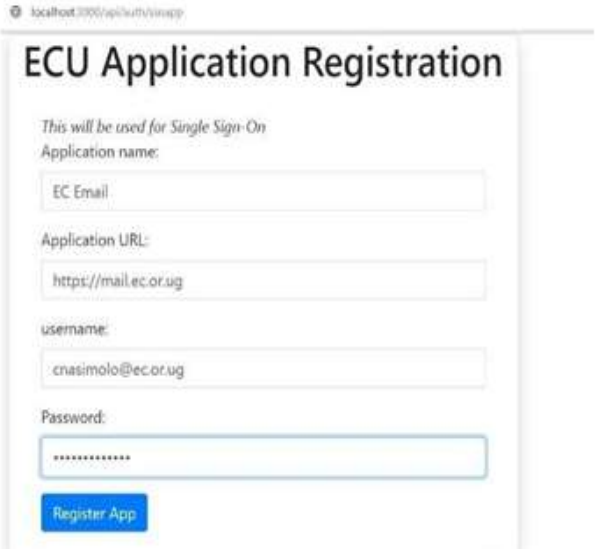
User Forms
Input Design
User Login Page



User Multifactor Authentication (MFA) Token page



Application Registration Form



User Role Registration

Output Design
Welcome Screen

User Application Dashboard

Email containing verification code/token

Source Code

The development of MFA and SSO systems involves various programming languages and frameworks. Below are some of the key source code snippets of the critical functions within this system alone with explanations of their roles

// / Register a new user

```
exports.register = (req, res) => {
const { username, password, email, role_id }
= req.body;

if (!username || !password || !email ||
!role_id) {
return res.status(400).json({ message: 'All fields are required'
});
}

bcrypt.hash(password, 10, (err, hashedPassword) =>
{
if (err) return res.status(500).json({ error: err });

const newUser = { username,
password_hash: hashedPassword, email,
role_id

};

// User login - set session after successful authentication
exports.login = (req, res) => {
const { username, password } = req.body;

User.findByUsername(username, (err, results) => {
if (err) return res.status(500).json({ error: err });
if (results.length === 0) return res.status(401).json({ message:
'Invalid credentials' });

const user = results[0];

bcrypt.compare(password, user.password_hash, (err, isMatch)
=> {
if (err) return res.status(500).json({ error: err });

db.query(query, [username,password], results) => {
if (err) throw err;

if (results.length === 0) {

async (err,

if (!isMatch) return res.status(401).json({ message: 'Invalid
credentials' });
```

• Configuratin of Nodemailer to enable database to send email to the user's preferred email:

```
// Nodemailer Configuration
const transporter = nodemailer.createTransport({
service: 'Gmail', auth: {
user: process.env.EMAIL_USER, pass:
process.env.EMAIL_PASS,
},
```

```
logger: true, // Enable logging debug: true, // Show detailed logs
});

generation of random token and emailing token to the user:
// Routes
app.get('/', (req, res) => { res.render('login');
});
app.post('/login', (req, res) => {
const { username, password } = req.body; const query = 'SELECT * FROM users
WHERE username = ? and password=?';

return res.send('Invalid username or password!');
}
//if passwords match const user = results[0];
const passwordMatch = await bcrypt.compare(password, user.password_hash);

if (!passwordMatch) {
return res.send('Invalid username or password!');
}
```

//Generation of random token, emailing the random token and it to see if what is entered matches what was generated in the database: comparing it

```
//Token generation
const verificationCode = crypto.randomBytes(3).toString('hex'). toUpperCase();

const updateQuery = 'UPDATE users SET verification = ? WHERE user_id = ?';
db.query(updateQuery, [verificationCode, user.user_id], (err) => {
if (err) throw err;

//email code to user const
mailOptions = {
from: 'carolyn.nasimolo@ec.or.ug', to: user.email,
subject: 'Your Verification Code', text:
`Your verification code is:
${verificationCode}`,
};

transporter.sendMail(mailOptions, (err, info) => {
if (err) {
console.error('Error sending email:', err); // Logs the complete error
return res.status(500).json({ message: 'Error sending verification email.', error: err.message });
}
res.status(200).json({ message: 'Verification email sent successfully.'
});
db.query(query, [userId,code], (err, results) => {
if (err) throw err;

if (results.length === 0) {
return res.send('User not found!');
}

const user = results[0];

if (user.verification !== code) { return
res.send('Invalid
verification code!');
}

const clearQuery = 'UPDATE users SET verification = NULL WHERE user_id = ?';
db.query(clearQuery, [userId], (err)
=> {
if (err) throw err;

});
});
});
});
//Verify code entered against that generated in the database
app.post('/verify', (req, res) => { const { userId, code } = req.body;

const query = 'SELECT * FROM users WHERE user_id = ? AND verification=?';

res.render('dashboard', { username: user.username });
});
});
});
```

- Use of crypto to encrypt and decrypt text for single sign on

```
const CryptoJS = require('crypto-js');
const User = require('../models/userModel');

const secretKey = '{EncryptKey}';

// Encrypting the text
const encrypted = CryptoJS.AES.encrypt('This is a secret', secretKey).toString();
console.log('Encrypted:', encrypted);

// Decrypting the text const bytes =
CryptoJS.AES.decrypt({PassWord}, secretKey);
const decrypted = bytes.toString(CryptoJS.enc.Utf8);
console.log('Decrypted:', decrypted);.
```

REFERENCES

1. SANS Institute. (2021). Implementing Multifactor Authentication: A Practical Guide. Retrieved from https://www.sans.org/white- papers/40102/
2. 2. ISO/IEC 27001: Information Security Management Systems
3. Mohammadreza Hazhirpasand Barkadehi, M. N. (2018). Authentication systems: A literature review and classification. Elservier, 1491-1511.
4. https://link.springer.com/chapter/10.10 07/3-540-45831-X_4
5. Hong, J.-J. K.-P. (2011). A Method of Risk Assessment for Multi-Factor. Journal of Information Processing Systems, 187.
6. Nisha, P. P. (2020). Challenges in Single Sign-On. Journal of Physics:Conference Series, 1.
7. National Institute of Standards and Technology (NIST).(2020). Special publication 800-63B: Digital Identity Guidelines Retrieved from https://csrc.nist.gov/publications/detail/ sp/800-63b/final
8. OWASP Foundation. (2021). Authentication Cheat Sheet. Retrieved from https://cheatsheetseries.owasp.org/cheat sheets/Authentication_Cheat_Sheet.html