Effortrak 2025 Vulnerability Report

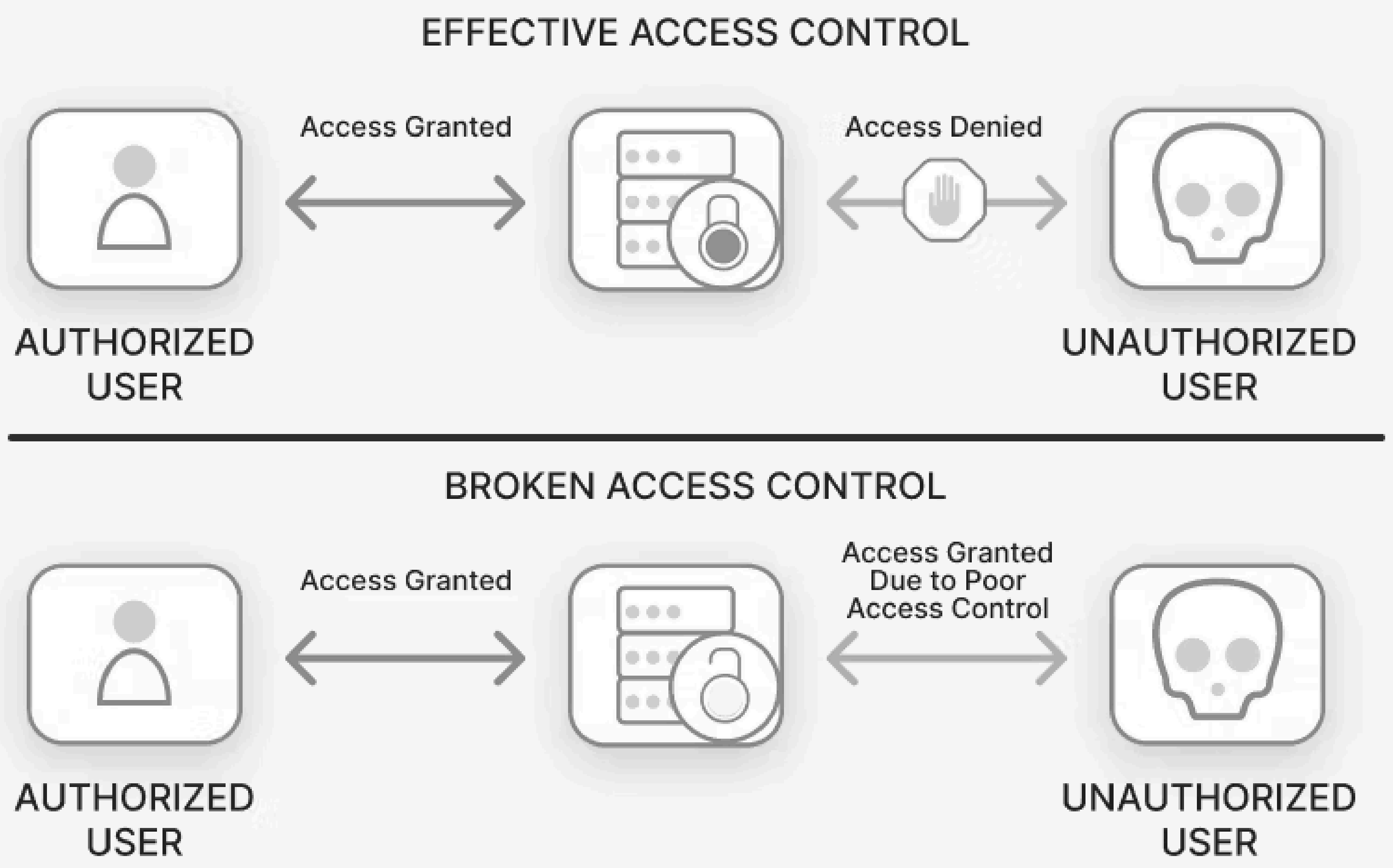
Security Researcher
Muhammet Emirhan SUMER

Contents

Vulnerability Description

Broken Access Control

Due to insufficient access control mechanisms in the system, users with the “User” role can directly access pages that should only be accessible to the “Admin” role via URL. This situation may allow unauthorized users to view or perform administrative tasks. This vulnerability poses a high risk and must be addressed immediately.



Vulnerability Description

Cryptographic Failures

Although the password field in the system appears masked as ******** in the user interface, when the HTML source code is examined, the actual password appears in plain text in the value attribute of the relevant <input type="password"> field. This situation means that even though the password is not visible in the user's browser, it can easily be obtained by third parties or malicious scripts.

SUMMARY

Summary

Broken Access Control

The URL:
https://tracker2.keylines.net/admin/clients/edit/<user_id>

is intended to be accessible only by administrator users. However, due to improper access control implementation, users with lower privileges (clients) can also access this endpoint. As a result, unauthorized users are able to modify the data of any user they choose. This poses a serious security vulnerability that compromises the integrity of the system and the confidentiality of user data.

Cryptographic Failures

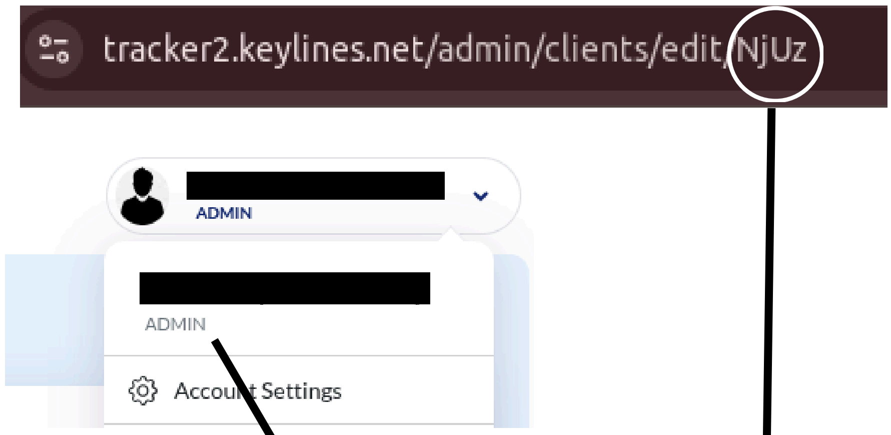
The URL:
https://tracker2.keylines.net/admin/clients/edit/<user_id>

the password field is masked (********) in the browser. However, when the page is inspected using the “Inspect Element” tool, it is observed that the actual password is stored in plain text within the value attribute of the password field. This situation enables unauthorized individuals to easily view password information, thereby seriously compromising the confidentiality of user data.

Detection Method

Broken Access Control

Due to insufficient access control mechanisms in the system, users with the “User” role can directly access pages that should only be accessible to the “Admin” role via URL. This situation may allow unauthorized users to view or perform administrative tasks. This vulnerability poses a high risk and must be addressed immediately.



tracker2.keylines.net/admin/clients/edit/NjUz

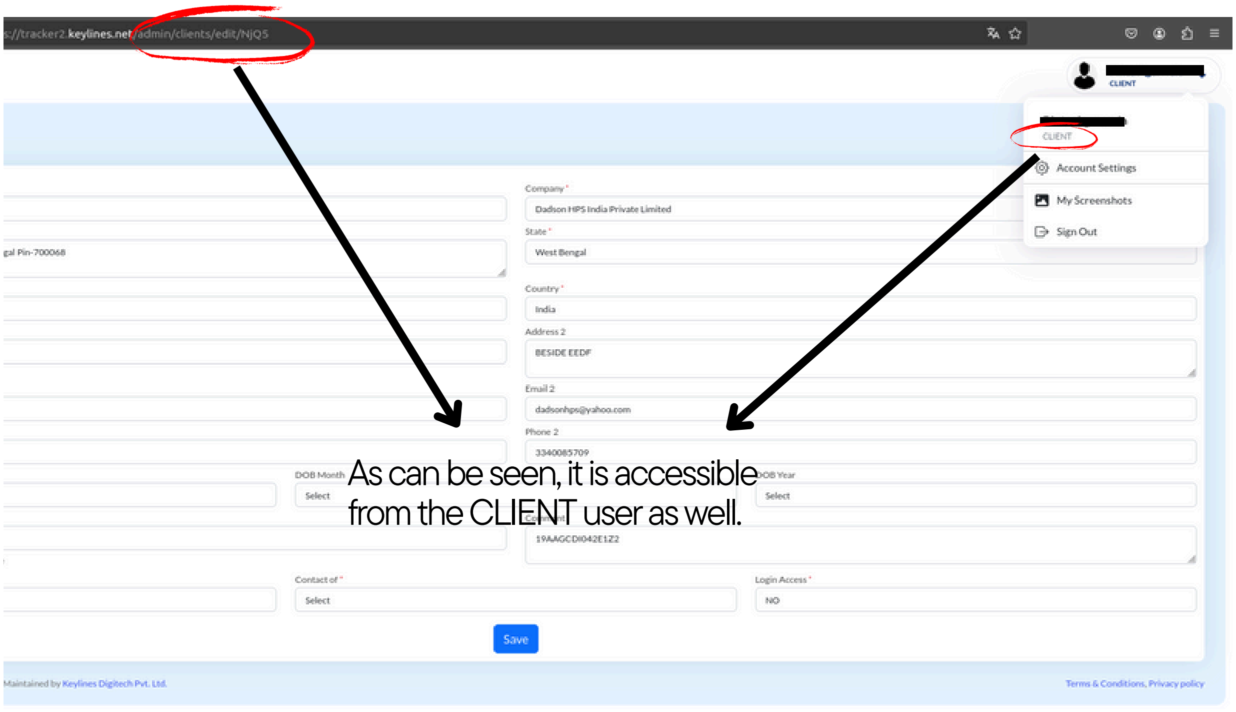
[redacted]
ADMIN

[redacted]
ADMIN

URL accessible to the admin user

Detection Method

Although this URL is only accessible after logging in, it is available to all logged in users as there is no authorization check.


s://tracker2.keylines.net/admin/clients/edit/NjQ5

[redacted]
CLIENT

[redacted]
CLIENT

Account Settings
My Screenshots
Sign Out

gal Pin-700068

Company *
Dadson HPS India Private Limited

State *
West Bengal

Country *
India

Address 2
BESIDE EEDF

Email 2
dadsonhps@yahoo.com

Phone 2
3340085709

DOB Month
Select

DOB Year
Select

Comment
19AAGCD0042E1Z2

Contact of *
Select

Login Access *
NO

Save

Maintained by Keylines Digitech Pvt. Ltd.

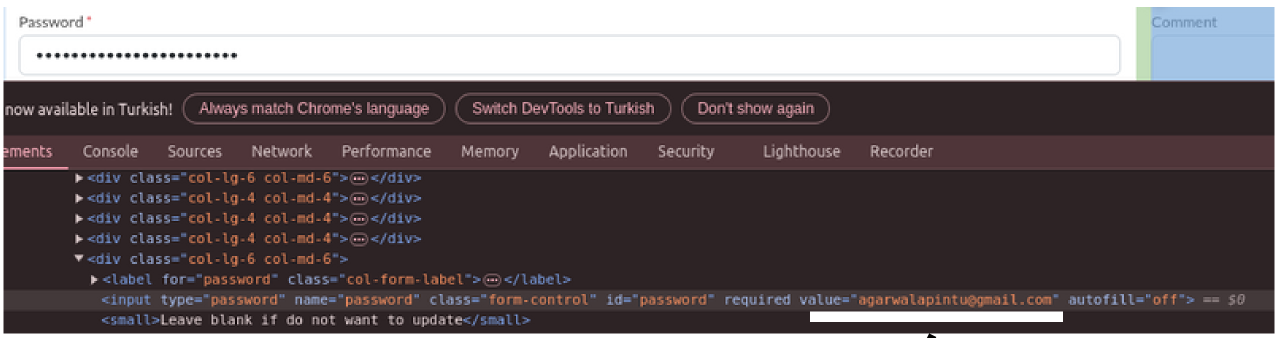
Terms & Conditions, Privacy policy

As can be seen, it is accessible from the CLIENT user as well.

Detection Method

Cryptographic Failures

This security vulnerability was detected by examining the password field in the low-privileged user profile editing screen. In the normal view, the password is displayed as *************. However, when the “Inspect” feature is used in the browser, it was observed that the actual password of the relevant user is stored in plain text within the value parameter of the input tag. Using this method, an admin user can directly obtain the actual password of any low-privileged user.



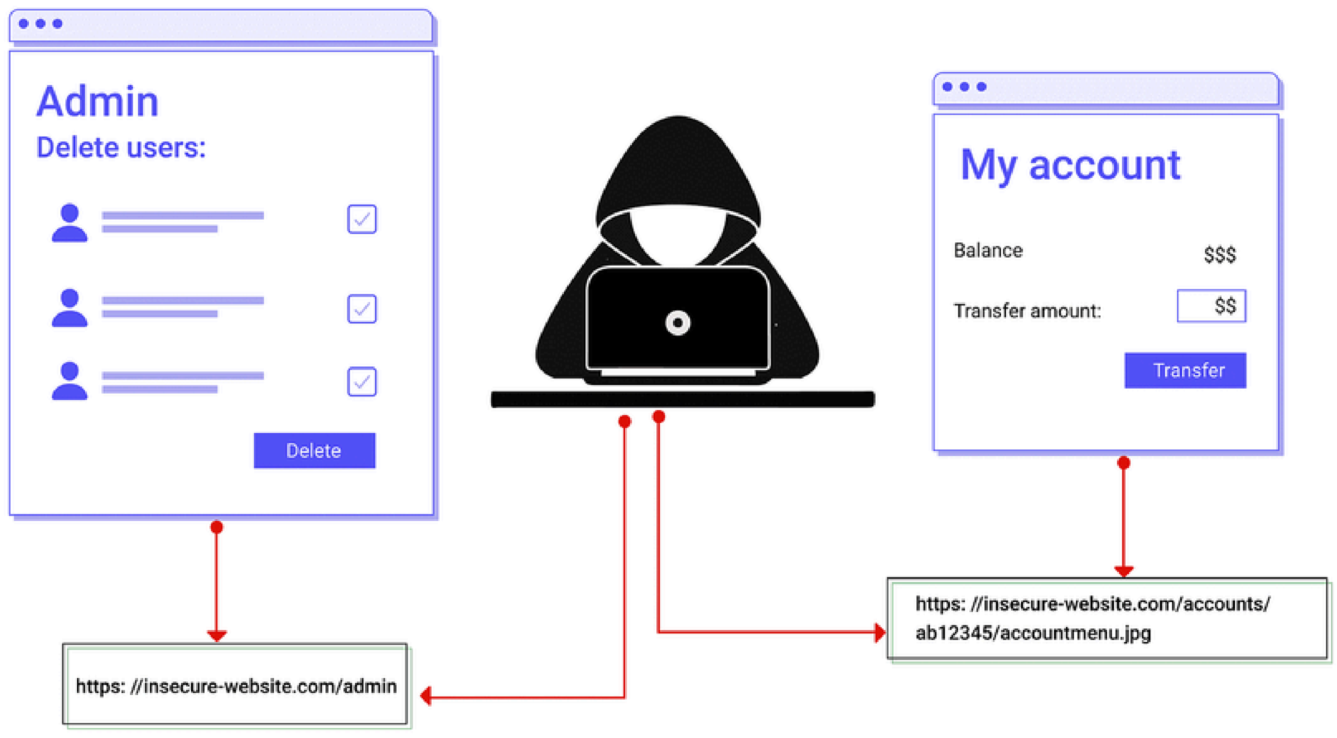
```
▶<div class="col-lg-6 col-md-6">⋯</div>
▶<div class="col-lg-4 col-md-4">⋯</div>
▶<div class="col-lg-4 col-md-4">⋯</div>
▶<div class="col-lg-4 col-md-4">⋯</div>
▼<div class="col-lg-6 col-md-6">
  ▶<label for="password" class="col-form-label">⋯</label>
   <input type="password" name="password" class="form-control" id="password" required value="agarwalapintu@gmail.com" autofill="off"> == $0
   <small>Leave blank if do not want to update</small>
```

As you can see, the password is accessible in plain text.

Impact

Broken Access Control

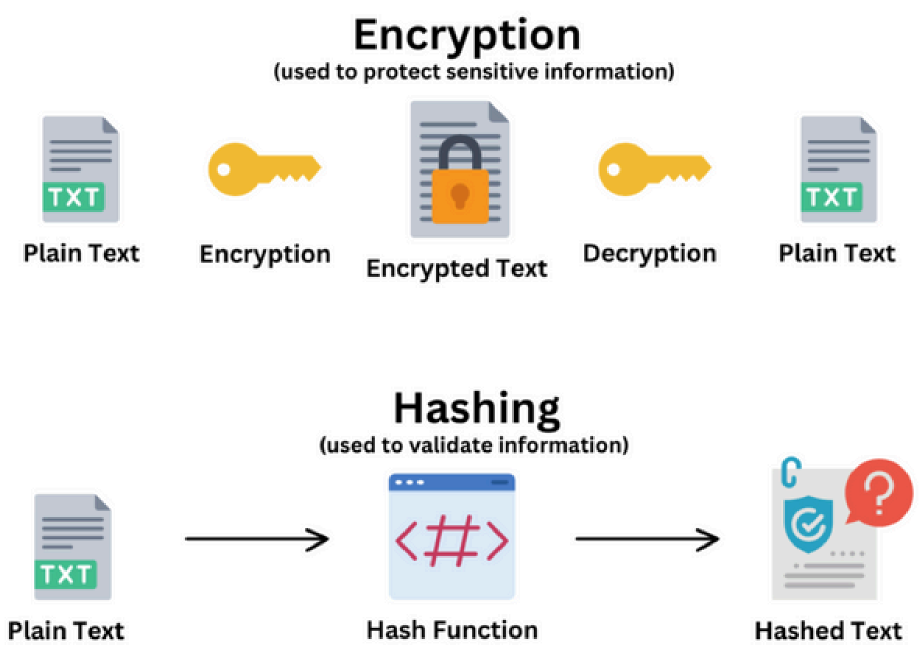
As a result of this security vulnerability, the admin user can directly obtain the password of any low-privileged user. This compromises the confidentiality of user accounts and may lead to unauthorized access. Since the compromised passwords can be reused in different systems or applications, chain security breaches related to password reuse may occur.


Admin
Delete users:
Delete
My account
Balance
$$$
Transfer amount:
$$
Transfer
https: //insecure-website.com/admin
https: //insecure-website.com/accounts/ab12345/accountmenu.jpg

Impact

Cryptographic Failures

As a result of this security vulnerability, the admin user can directly obtain the password of any low-privileged user. This compromises the confidentiality of user accounts and may lead to unauthorized access. Since the compromised passwords can be reused in different systems or applications, chain security breaches related to password reuse may occur.



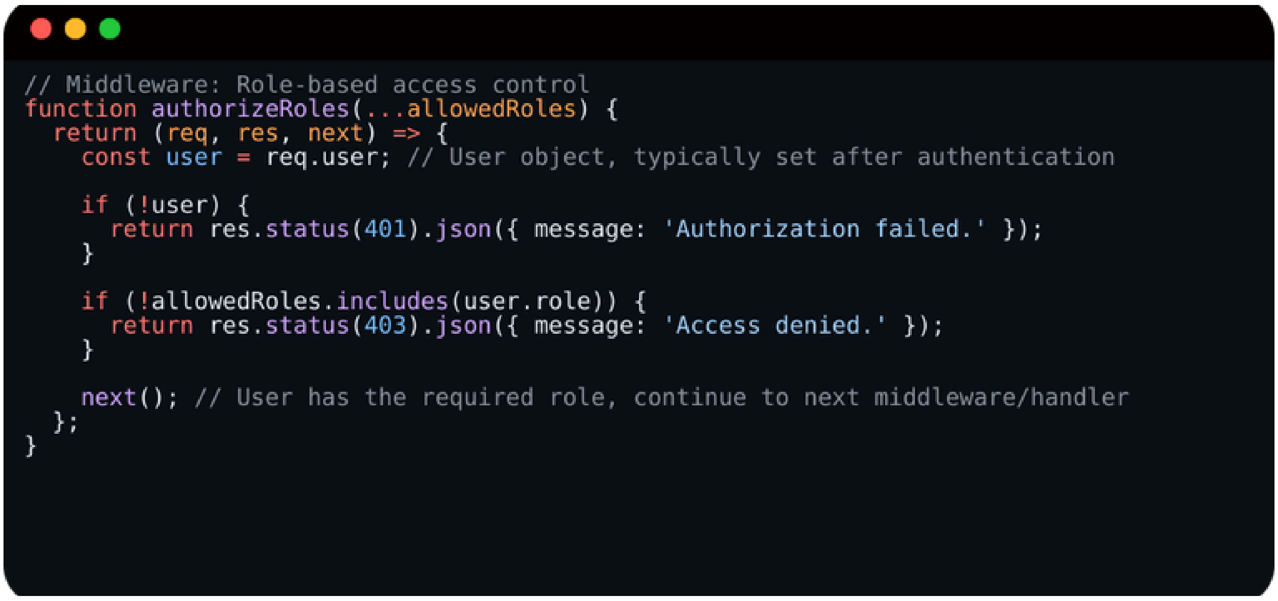
Mitigation & Recommendations

BROKEN ACCESS CONTROL

Mitigation & Recommendations

- Server-Side Role Verification Should Be Implemented For every incoming request, the user's role (e.g., “User,” “Admin”) should be verified on the server side, and only users with authorized roles should be allowed to access the relevant pages or processes.

Example Code

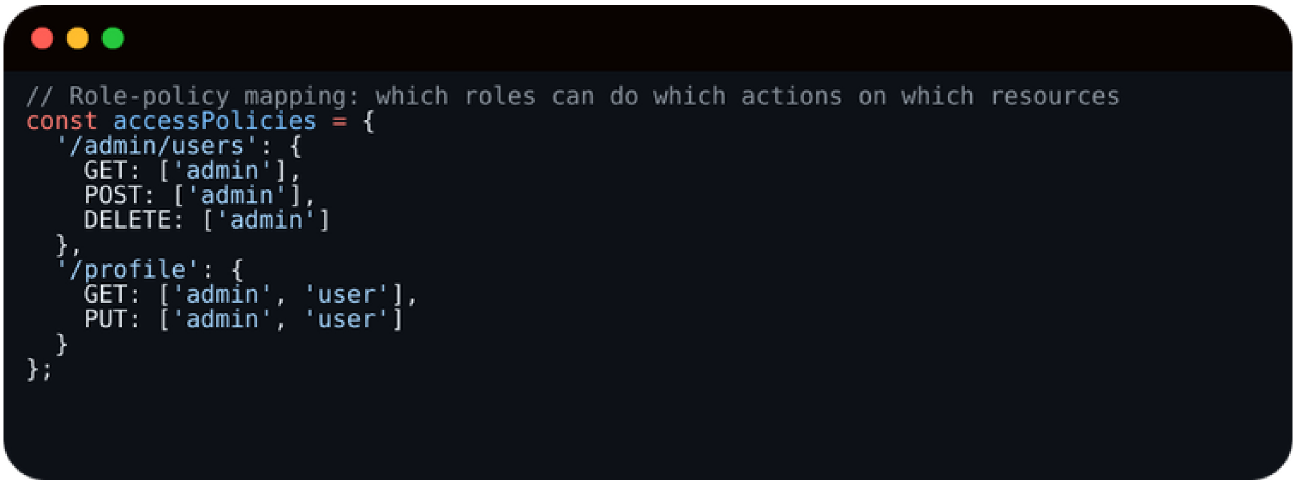


```
// Middleware: Role-based access control
function authorizeRoles(...allowedRoles) {
  return (req, res, next) => {
    const user = req.user; // User object, typically set after authentication

    if (!user) {
      return res.status(401).json({ message: 'Authorization failed.' });
    }

    if (!allowedRoles.includes(user.role)) {
      return res.status(403).json({ message: 'Access denied.' });
    }

    next(); // User has the required role, continue to next middleware/handler
  };
}
```

Mitigation & Recommendations

For more flexible structures beyond RBAC, ACL or Policy-based Access Control (PBAC) systems can be implemented. It must be clearly defined which role can perform which operations for each resource.

Example Code



```
// Role-policy mapping: which roles can do which actions on which resources
const accessPolicies = {
  '/admin/users': {
    GET: ['admin'],
    POST: ['admin'],
    DELETE: ['admin']
  },
  '/profile': {
    GET: ['admin', 'user'],
    PUT: ['admin', 'user']
  }
};
```

Cryptographic Failures

Mitigation & Recommendations

Passwords should never be displayed in plain text in the frontend.
Password fields should not contain any data in the form of value="password". Passwords should never be displayed to the user (even if they are hidden) and should only be entered manually by the user.

Password Information Should Never Be Sent to the Client Side
The server should not send password information to the frontend; this information should not be stored on the client side in any way. All password operations should only be performed on the backend side and through secure channels.

My Most Important Advice

My Most Important Advice

The mistake is at your fingertips. A tiny mistake you make can lead to big problems. Develop secure code and use **STRONG** passwords.

DON'T FORGET

The Biggest Security Vulnerability is HUMAN

End...