JUL 13 , 2023

UOTTAWA UNEVERSITY

SECURE SOFTWARE DESIGN

Deliverable F

Team 3

Student Name	Student id
Asmaa EmadEldin	300389563
Hager Mahmoud	300389397
Manar Ibrahim	300389404
Mariam Galal	300389366
Sara Galala	300389879

Severity Scaling

id	issue	description	severity	Reason for Severity Score	status
27	Easy to access the solutions of challenge from the URL (Bypassing-Authorization)	Any user can reach to the markdown solution pages directly by writing the URL of the wanted solution like(https://securedojo-team3.onrender.com/static/lessons/blackBelt/cwe311.sol.md)	2	as users could be able to read data on the server that they aren't have the access to reach to	closed
32	Easy to access the not allowed solutions (Bypassing-Authorization)	Any user can access any solution for any lesson by the url, like(https://securedojo-team3.onrender.com/challenges/solutions/cwe311) even if this challenge is forbidden for him to be shown	2	as users could be able to read data on the server that they aren't have the access to reach to	closed
28	Broken access Control leads to modify/manipulate the view of the solutions by normal user	Any normal user/student can modify/change/manipulate the view of the challenges' solutions. By the following steps: -Create a normal user/student then login. -Create an instructor and login (on another session OR private window OR different browser) -Prepare proxy and burp-suite tool to capture the (instructor traffic). -Intercept this Post request /selectedchallenges -change the original cookie of the instructor on (the burp suite's intercept proxy) with this normal/student user's Cookie, and of course make a little change in the body. -Forward the request.	1	as unauthorized users could be able to access the server and write and change the data inside it	Closed

id	issue	description	severity	Reason for Severity Score	status
29	Privilege Escalation Vulnerability: User Creation Allows Unauthorized Instructor Account Creation	Any normal user can easily create an instructor account by the following steps. 1- Create a normal user/student then log in. 2- Create a post request with body: {"newUser":{ "username":"insta22", … …. **"choice":"instuctor"** }} 3- Using Burpsuite send to this endpoint **'/puplic/instructorregister'** and change the body choice to "instructor" using the normal user session cookie and click Send.	1	as unauthorized users could be able to access the server and write and change the data inside it	closed
30	User Authorization Vulnerability: Unauthorized Creation of 'admin' Account Bypassing System Checks	Making instructor account with name admin so this new account has the privileges of the true admin of the application (can be able to add new instructors) by following steps: -Entered the admin account -Register a new account with a username “admin”. -Entering the instructor account -write this url(https://securedojo-team3.onrender.com/admin)	1	as unauthorized users could be able to access the server and write and change the data inside it	closed
31	Instructor can't change his password	Can't edit the password after logging as an instructor.	4	could be optimized further in terms of performance or security	closed
12	Missing rate limiting	The code does not include rate limiting, which can be a serious security vulnerability. Without rate limiting, an attacker could potentially send a large number of requests to the "/api/teams" endpoint, which could cause the server to become overwhelmed and potentially crash.	4	could be optimized further in terms of performance or security	Closed

Id	issue	description	severity	Reason for Severity Score	status
13	Uncontrolled data used in path expression	Accessing files using paths constructed from user-controlled data can allow an attacker to access unexpected resources. This can result in sensitive information being revealed or deleted, or an attacker being able to influence behavior by modifying unexpected files. for this path(var defs = Object.freeze(require(path.join(__dirname, getModulePath(moduleId), '/definitions.json'))))	1	as attacker could be able to access the server and write and change the data inside it	closed
14	Inclusion of functionality from an untrusted source	using a library from ustrusted source	4	could be optimized further in terms of performance or security	closed
33	Uncommon header 'alt-svc' found, with contents: h3=":443"; ma=86400	Security of the Protocol: HTTP/3 is designed to be secure, but like any protocol, it may have its vulnerabilities. Keeping the protocol and associated software up to date is essential for security.	4	Non-issue that could be optimized further in terms of performance or security.	Closed
26	Modern Web Application Using Non-Traditional Links	The web application appears to be a modern web application that uses non-traditional links, which may indicate the use of Single Page Application (SPA) frameworks or other modern web development techniques. These links do not have traditional href attributes, which means that they may not be crawled by traditional web crawlers.	4	Non-issue that could be optimized further in terms of performance or security.	Closed

Id	issue	description	severity	Reason for Severity Score	status
25	Loosely scoped cookie	The web application has set a cookie with a domain scope that may allow unauthorized access to the cookie. The domain scope determines which domains can access the cookie, and cookies scoped to a parent-level domain may be transmitted to the parent or any subdomain of the parent, potentially allowing unauthorized access to the cookie. The origin domain used for comparison is "securedojo1.onrender.com", and the cookie with the vulnerability is "__cf_bm=aAAEEZSUqnbVimXJe6so2449.dEOSIBVZbVKQ8Las2s-1687784922-0AbOjM5wdXQs7J4dw4hD8w3AzQHbuw9/irrVAjlTE13T2mRscZhh2dlkvDqLv0hqY9L5wB7y/IpYlK7m2BwIh2Tc=".	4	Non-issue that could be optimized further in terms of performance or security.	Closed
24	Suspicious Comments in HTTP Response	The HTTP response of the web application contains suspicious comments that may help an attacker. These comments may contain sensitive information, such as usernames, that can be used to launch targeted attacks on the web application.	4	Non-issue that could be optimized further in terms of performance or security.	Closed
16	Exposure of private files	The code uses the Express.js middleware function express.static() to serve static files from the specified directory. The vulnerability of "Exposure of private files" may occur if the directory specified contains private files that are not intended to be publicly accessible	4	Non-issue that could be optimized further in terms of performance or security.	Closed

Id	issue	description	severity	Reason for Severity Score	status
11	Vunerabe JS Libraray	The code uses the AngularJS library, specifically version 1.8.3, which is at risk of being compromised. This vulnerability may allow attackers to perform various types of attacks, such as cross-site scripting (XSS), cross-site request forgery (CSRF), or code injection attacks.	4	Non-issue that could be optimized further in terms of performance or security.	open
46	user data is not saved	the student data that you registered is deleted after closing the website. This can be a significant problem, especially if the data is important and needs to be accessed later	4	Non-issue that could be optimized further in terms of performance or security.	Closed
45	TCP Timestamp s Informatio n Disclosure	TCP Timestamps Information Disclosure is a security vulnerability that can occur when the remote host implements TCP timestamps, as defined by RFC1323/RFC7323. This vulnerability arises because the TCP timestamps feature includes a time stamp value in the TCP header, which can be used to estimate the uptime of the system and the duration of the connection. An attacker can use this information to gather intelligence about the system and potentially launch further attacks.	4	Non-issue that could be optimized further in terms of performance or security.	Closed
36	open ports not in use	network ports that are available for communication but have no active application or service using them. These ports can pose a security risk by providing attackers an entry point to exploit known vulnerabilities	4	Non-issue that could be optimized further in terms of performance or security.	Closed

Id	issue	description	severity	Reason for Severity Score	status
34	Uncommon header 'x-render-origin-server' found, with contents: Render	The 'x-render-origin-server' header in an HTTP response provides information about the origin server that rendered the response. However, if misconfigured or not properly validated, it can be manipulated by attackers to inject malicious code or steal sensitive information	4	Non-issue that could be optimized further in terms of performance or security.	Closed
23	Server Information Leakage via X-Powered-By HTTP response header	The web/application server is leaking information through one or more "X-Powered-By" HTTP response headers. This header provides information about the technology stack and software components used to build and run the web application, which can be exploited by attackers to identify vulnerabilities in those components.	4	Non-issue that could be optimized further in terms of performance or security.	Closed
22	Cookie SameSite Attribute Set to None Vulnerability	The web application has set a cookie with its SameSite attribute set to "none". This means that the cookie can be sent as a result of a "cross-site" request, which can potentially be exploited by attackers to launch Cross-Site Request Forgery (CSRF), Cross-Site Script Inclusion (XSSI), and timing attacks.	4	Non-issue that could be optimized further in terms of performance or security.	Closed
21	Csp: Style-Src Unsafe-Inline Vulnerability	The CSP header contains the style-src directive with the unsafe-inline value, which allows inline styles to be executed, leading to XSS attacks.	4	Non-issue that could be optimized further in terms of performance or security.	Closed

Id	issue	description	severity	Reason for Severity Score	status
20	Csp: Script-Src Unsafe-Inline Vulnerability	The CSP header contains the script-src directive with the unsafe-inline value, which allows inline scripts to be executed, leading to XSS attacks.	4	Non-issue that could be optimized further in terms of performance or security.	Closed
19	Csp: Script-Src Unsafe-Eval Vulnerability	The CSP header contains the script-src directive with the unsafe-eval value, which allows scripts to be executed from unsafe sources, leading to XSS attacks.	4	Non-issue that could be optimized further in terms of performance or security	Closed
18	Csp: Wildcard Directive Vulnerability	The wildcard directive is being used in the CSP header, which means that sources that may not have been explicitly approved by the website owner can be loaded, potentially leading to Cross-Site Scripting (XSS) and data injection attacks. The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined: style-src, img-src, connect-src, frame-src, frame-ancestors, font-src, media-src, object-src, manifest-src, prefetch-src, form-action The directive(s): frame-ancestors, form-action are among the directives that do not fallback to default-src, missing/excluding them is the same as allowing anything.	4	Non-issue that could be optimized further in terms of performance or security	Closed

Id	issue	description	severity	Reason for Severity Score	status
10	Hidden file was found	The system has been found to have an accessible sensitive file, which poses a risk of leaking important information such as administrative details, configuration settings, or credentials. This information can be exploited by malicious individuals to launch further attacks on the system or conduct social engineering campaigns.	4	Non-issue that could be optimized further in terms of performance or security	Closed
9	User Agent Fuzzer	The User Agent Fuzzer is a vulnerability check that involves testing for differences in response based on fuzzed User Agent strings. This technique is commonly used to identify if a web application is treating different User Agents differently, which could potentially be exploited by an attacker.	4	Non-issue that could be optimized further in terms of performance or security	Closed
8	HyperText Transfer Protocol(HTTP) Redirect	the remote web server issues an HTTP redirect when requesting the root directory of the web server this indicates that a web server is issuing an HTTP redirect when a request is made for the root directory of the server. While HTTP redirects can be useful, they can also be used for phishing attacks or to redirect users to malicious websites.	4	Non-issue that could be optimized further in terms of performance or security	Closed
7	HTTP Methods Allowed	It indicates that a web server is allowing certain HTTP methods on specific directories of the server. Allowing certain methods on specific directories can be useful, but it can also lead to security vulnerabilities if not properly configured	4	Non-issue that could be optimized further in terms of performance or security	Closed

Id	issue	description	severity	Reason for Severity Score	status
6	Syn Open Port Detected	The Nessus vulnerability scanner has detected an issue related to a Syn half-open port scanner. Port 8081/tcp was found to be open	4	Non-issue that could be optimized further in terms of performance or security	Closed
5	Web Server No 404 Error Code	The remote web server is configured such that it does not return '404 Not Found' error codes when a nonexistent file is requested,	4	Non-issue that could be optimized further in terms of performance or security	Closed
4	JQuery Detection	The vulnerability scanner detected that we used a jquery library which typically points to a web server running on our own machine.	4	Non-issue that could be optimized further in terms of performance or security	Closed
1	Web Application Sitemap	The remote web server contains linkable content that can be used to gather information about a target.	4	Non-issue that could be optimized further in terms of performance or security	Closed

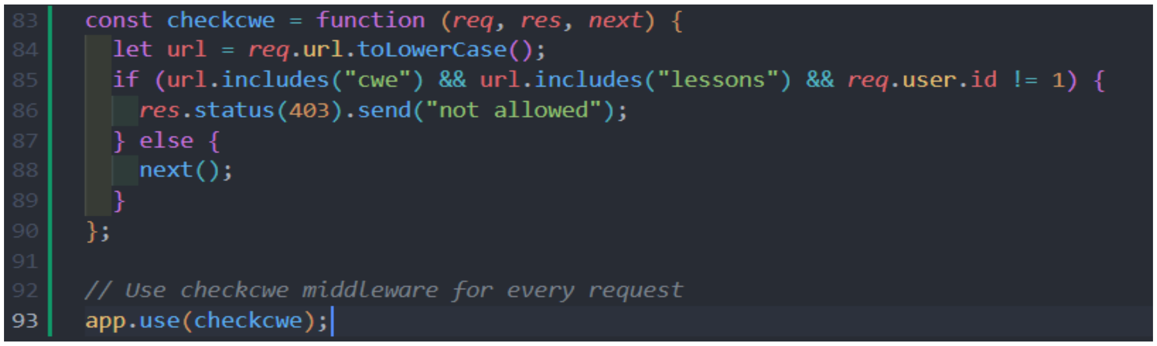
Id	issue	description	severity	Reason for Severity Score	status
44	Absence of Anti-CSRF Tokens	No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: · The victim has an active session on the target site. · The victim is authenticated via HTTP auth on the target site. · The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.	4	Non-issue that could be optimized further in terms of performance or security	Closed

Fix History

27: Easy to access the solutions of challenge from the URL (Bypassing-Authorization)

Closed on 7/11/2023.

We solved this issue by adding (checkcwe function) this function checks if the requested url contains a “cwe” word and if the user isn’t the admin. Then the user will be directed to page contains “not allowed” and will not be directed to the solution markdown page.

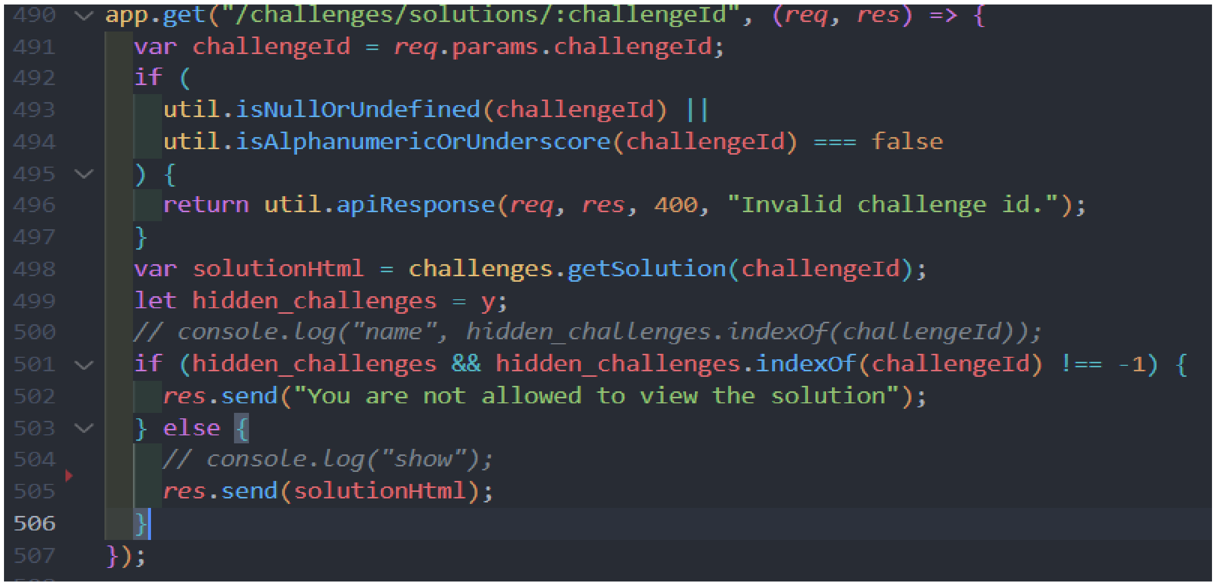


```
const checkcwe = function (req, res, next) {
  let url = req.url.toLowerCase();
  if (url.includes("cwe") && url.includes("lessons") && req.user.id != 1) {
    res.status(403).send("not allowed");
  } else {
    next();
  }
};

// Use checkcwe middleware for every request
app.use(checkcwe);
```

32: Easy to access the not allowed solutions (Bypassing-Authorization)

Closed at 7/2/2023.

We solved this issue by adding a condition to check if the requested challenge id is assigned to the hidden challenges of the logged in user or not, and if it is a hidden challenge for this user the application will redirect the user to the main page.

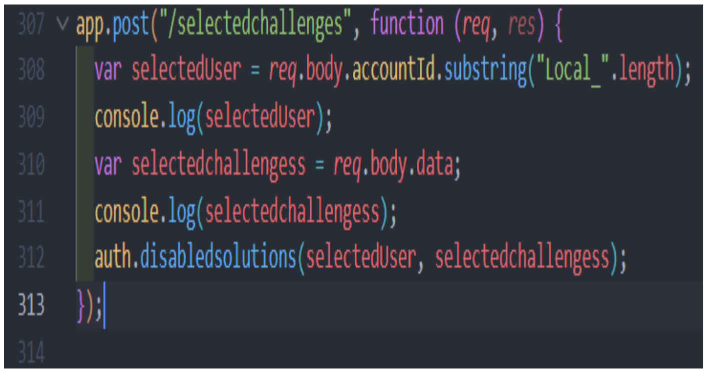


```
app.get("/challenges/solutions/:challengeId", (req, res) => {
  var challengeId = req.params.challengeId;
  if (
    util.isNullOrUndefined(challengeId) ||
    util.isAlphanumericOrUnderscore(challengeId) === false
  ) {
    return util.apiResponse(req, res, 400, "Invalid challenge id.");
  }
  var solutionHtml = challenges.getSolution(challengeId);
  let hidden_challenges = y;
  // console.log("name", hidden_challenges.indexOf(challengeId));
  if (hidden_challenges && hidden_challenges.indexOf(challengeId) !== -1) {
    res.send("You are not allowed to view the solution");
  } else {
    // console.log("show");
    res.send(solutionHtml);
  }
});
```

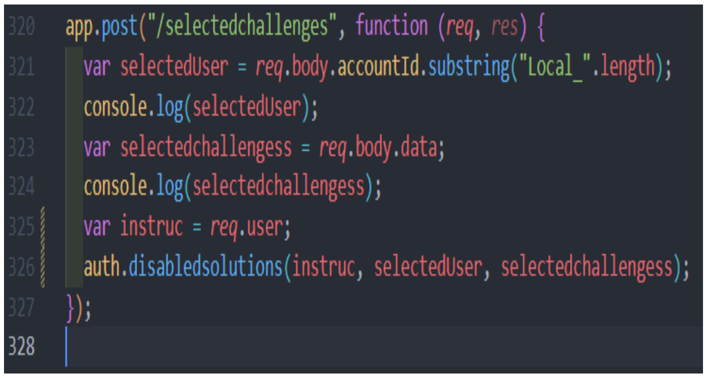
28: Broken access Control leads to modify/manipulate the view of the solutions by normal user

Closed on 7/11/2023.

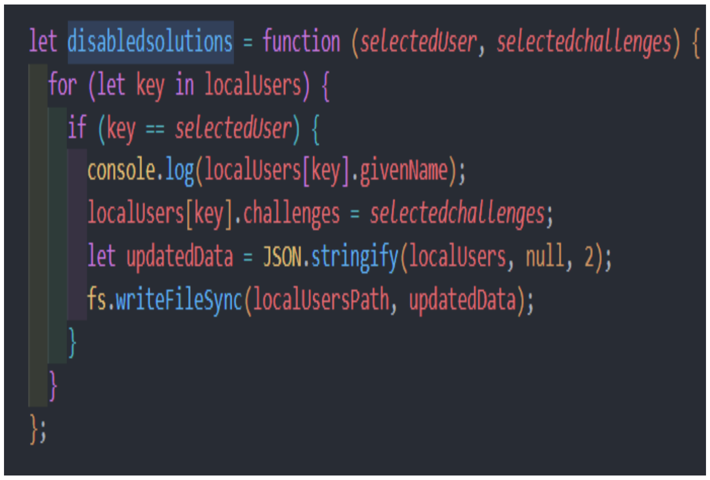
The issue was solved by adding a condition in the “/selectedchallenges” endpoint to check if the request to this endpoint is sent by the instructor or not, and if it is by the instructor he will be able to do the changes he wanted and if not, the user will be directed to not allowed page.



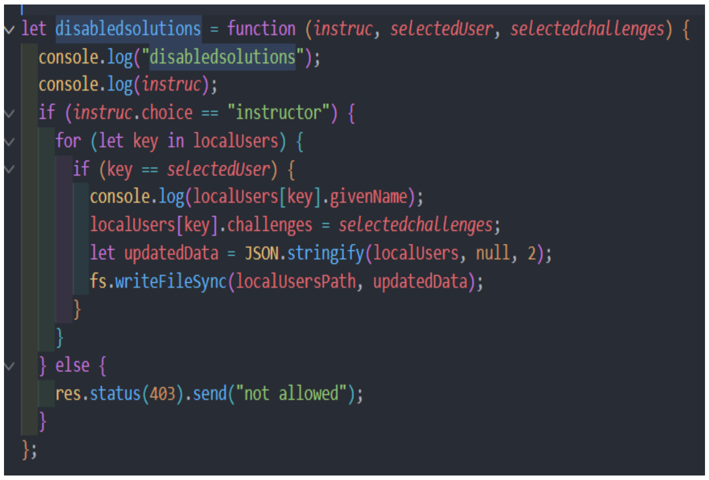
```
app.post("/selectedchallenges", function (req, res) {
    var selectedUser = req.body.accountId.substring("Local_".length);
    console.log(selectedUser);
    var selectedchallengess = req.body.data;
    console.log(selectedchallengess);
    auth.disabledsolutions(selectedUser, selectedchallengess);
});
```



```
app.post("/selectedchallenges", function (req, res) {
    var selectedUser = req.body.accountId.substring("Local_".length);
    console.log(selectedUser);
    var selectedchallengess = req.body.data;
    console.log(selectedchallengess);
    var instruc = req.user;
    auth.disabledsolutions(instruc, selectedUser, selectedchallengess);
});
```



```
let disabledsolutions = function (selectedUser, selectedchallenges) {
  for (let key in localUsers) {
    if (key == selectedUser) {
      console.log(localUsers[key].givenName);
      localUsers[key].challenges = selectedchallenges;
      let updatedData = JSON.stringify(localUsers, null, 2);
      fs.writeFileSync(localUsersPath, updatedData);
    }
  }
};
```

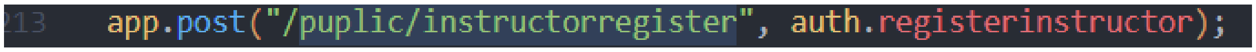


```
let disabledsolutions = function (instruc, selectedUser, selectedchallenges) {
  console.log("disabledsolutions");
  console.log(instruc);
  if (instruc.choice == "instructor") {
    for (let key in localUsers) {
      if (key == selectedUser) {
        console.log(localUsers[key].givenName);
        localUsers[key].challenges = selectedchallenges;
        let updatedData = JSON.stringify(localUsers, null, 2);
        fs.writeFileSync(localUsersPath, updatedData);
      }
    }
  } else {
    res.status(403).send("not allowed");
  }
};
```

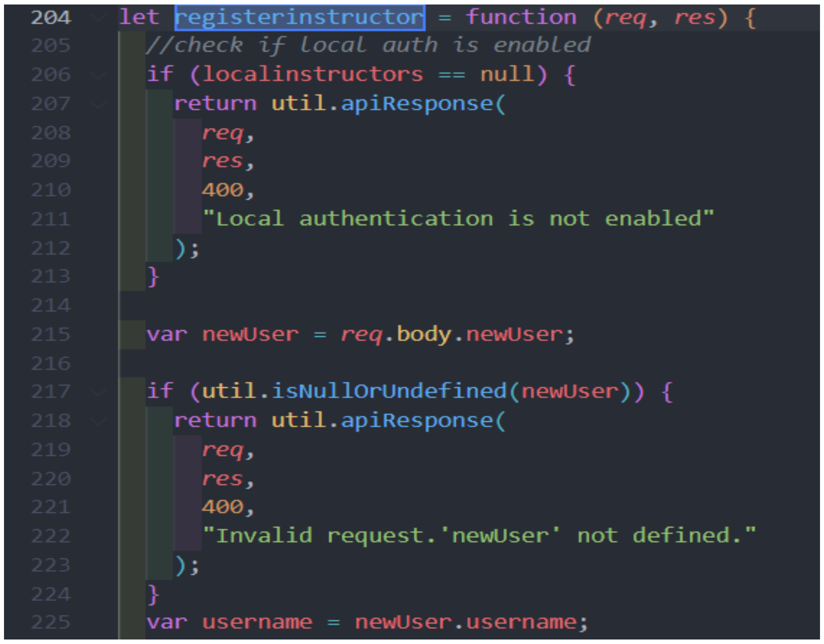
29: Privilege Escalation Vulnerability: User Creation Allows Unauthorized Instructor Account Creation

Closed on 7/11/2023

The end point" **'/puplic/instructorregister'**" shouldn't be accessed by anyone except the admin, so the issue was solved by making a check inside this endpoint that the request is sent by the admin or not and if it is sent by the admin so he will be able to add a new instructor and if not, he will be directed to

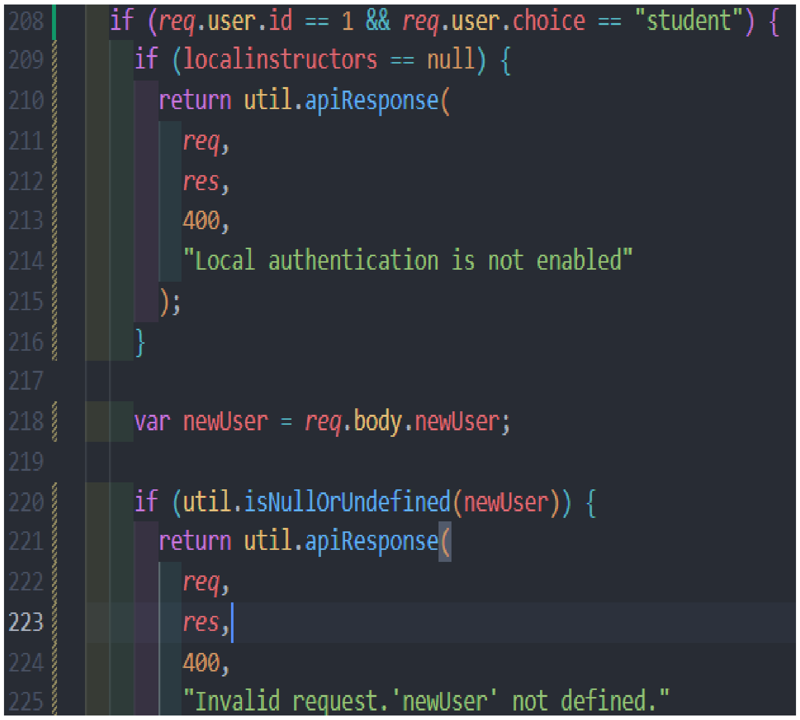


```
app.post("/puplic/instructorregister", auth.registerinstructor);
```



```
let registerinstructor = function (req, res) {
  //check if local auth is enabled
  if (localinstructors == null) {
    return util.apiResponse(
      req,
      res,
      400,
      "Local authentication is not enabled"
    );
  }

  var newUser = req.body.newUser;

  if (util.isNullOrUndefined(newUser)) {
    return util.apiResponse(
      req,
      res,
      400,
      "Invalid request.'newUser' not defined."
    );
  }
  var username = newUser.username;
```

(The id 1 is the id of the admin)



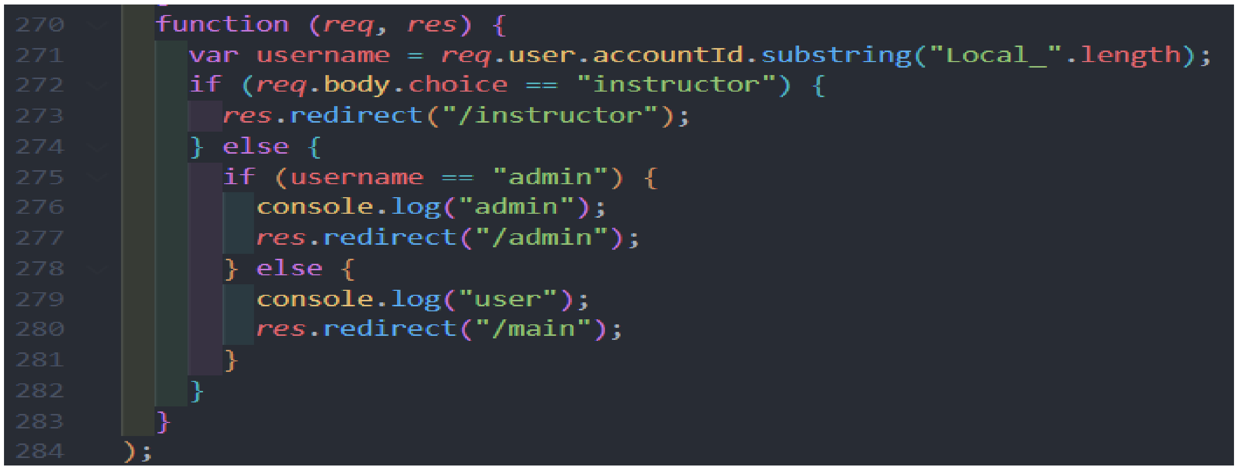
```
  if (req.user.id == 1 && req.user.choice == "student") {
    if (localinstructors == null) {
      return util.apiResponse(
        req,
        res,
        400,
        "Local authentication is not enabled"
      );
    }

    var newUser = req.body.newUser;

    if (util.isNullOrUndefined(newUser)) {
      return util.apiResponse(
        req,
        res,
        400,
        "Invalid request.'newUser' not defined."
```

30: User Authorization Vulnerability: Unauthorized Creation of 'admin' Account Bypassing System Checks.

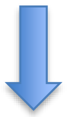
Closed on 7/2/2023.

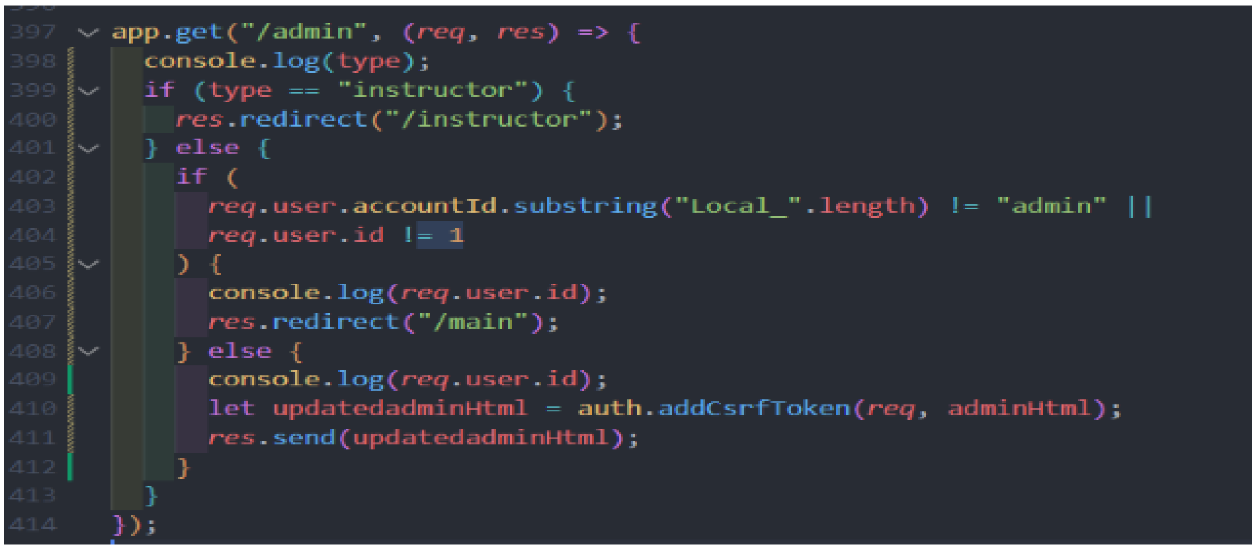
Our website has one admin and his record saved in student database (not instructor database), so the vulnerability done by creating a new instructor with a username equals admin so this instructor could get all the admin privileges.

We solved this issue by checking that no one can reach the admin page except him by making the checking by the admin date(id,choice)



```
  function (req, res) {
    var username = req.user.accountId.substring("Local_".length);
    if (req.body.choice == "instructor") {
      res.redirect("/instructor");
    } else {
      if (username == "admin") {
        console.log("admin");
        res.redirect("/admin");
      } else {
        console.log("user");
        res.redirect("/main");
      }
    }
  }
);
```



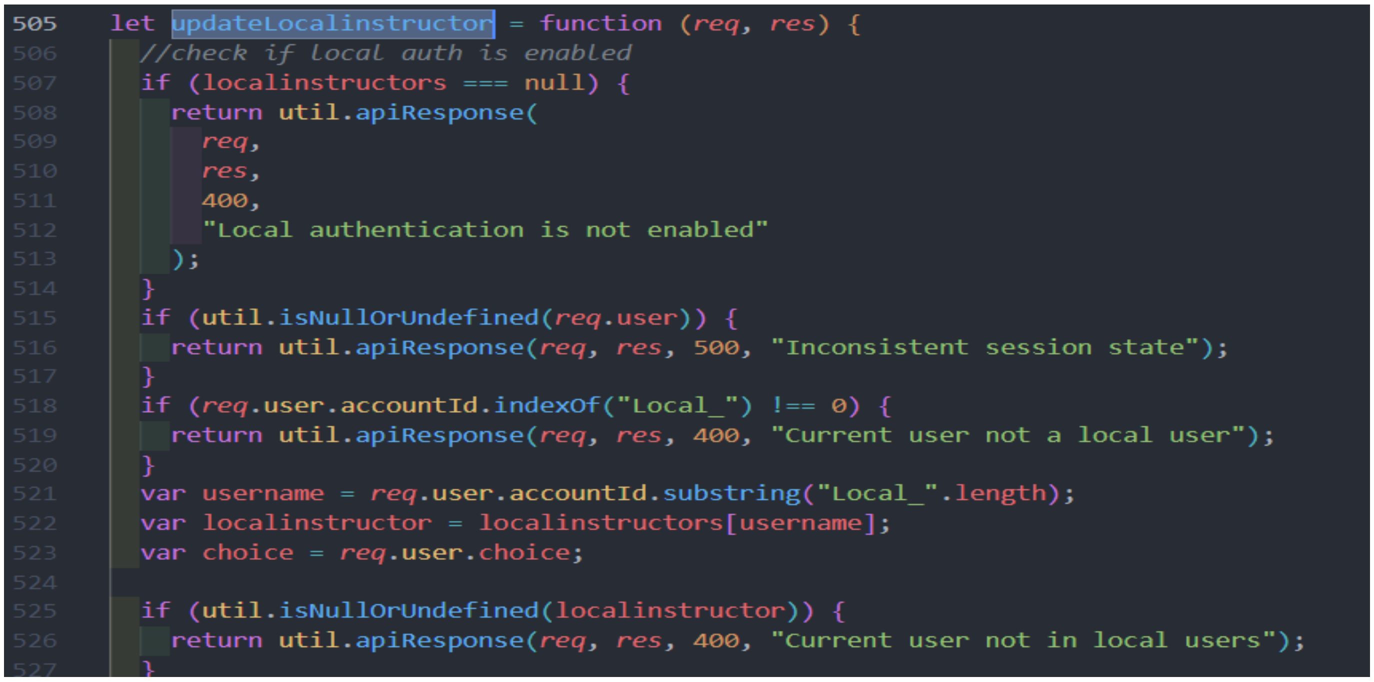


```
app.get("/admin", (req, res) => {
  console.log(type);
  if (type == "instructor") {
    res.redirect("/instructor");
  } else {
    if (
      req.user.accountId.substring("Local_".length) != "admin" ||
      req.user.id != 1
    ) {
      console.log(req.user.id);
      res.redirect("/main");
    } else {
      console.log(req.user.id);
      let updatedadminHtml = auth.addCsrfToken(req, adminHtml);
      res.send(updatedadminHtml);
    }
  }
});
```

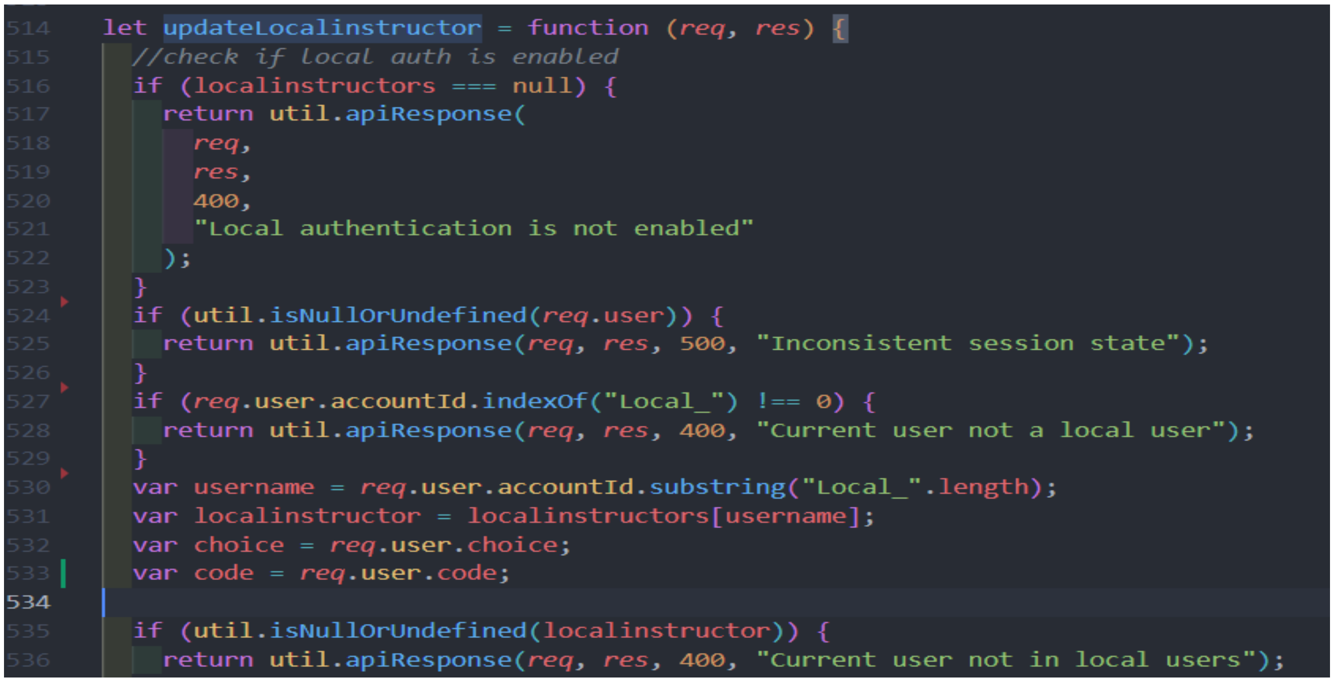
31: Instructor can't change his password

Closed on 7/2/2023.

There was a missing key(choice) of the instructor object inside the “updateLocalinstructor” function that was causing an error when trying to call this function, we solved this issue by editing this function and add the missed variable”choice”



```
let updateLocalinstructor = function (req, res) {
  //check if local auth is enabled
  if (localinstructors === null) {
    return util.apiResponse(
      req,
      res,
      400,
      "Local authentication is not enabled"
    );
  }
  if (util.isNullOrUndefined(req.user)) {
    return util.apiResponse(req, res, 500, "Inconsistent session state");
  }
  if (req.user.accountId.indexOf("Local_") !== 0) {
    return util.apiResponse(req, res, 400, "Current user not a local user");
  }
  var username = req.user.accountId.substring("Local_".length);
  var localinstructor = localinstructors[username];
  var choice = req.user.choice;

  if (util.isNullOrUndefined(localinstructor)) {
    return util.apiResponse(req, res, 400, "Current user not in local users");
  }
```

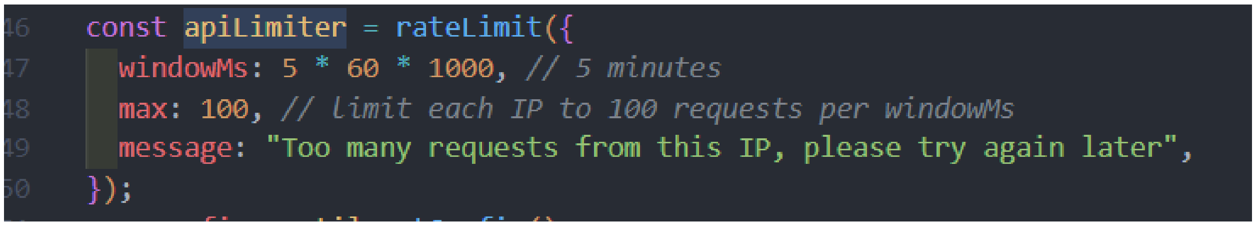


```
let updateLocalinstructor = function (req, res) {
  //check if local auth is enabled
  if (localinstructors === null) {
    return util.apiResponse(
      req,
      res,
      400,
      "Local authentication is not enabled"
    );
  }
  if (util.isNullOrUndefined(req.user)) {
    return util.apiResponse(req, res, 500, "Inconsistent session state");
  }
  if (req.user.accountId.indexOf("Local_") !== 0) {
    return util.apiResponse(req, res, 400, "Current user not a local user");
  }
  var username = req.user.accountId.substring("Local_".length);
  var localinstructor = localinstructors[username];
  var choice = req.user.choice;
  var code = req.user.code;

  if (util.isNullOrUndefined(localinstructor)) {
    return util.apiResponse(req, res, 400, "Current user not in local users");
```

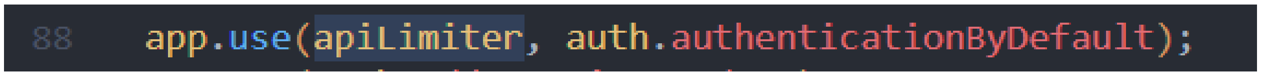
12:Missing rate limiting

closed on 6/26/2023

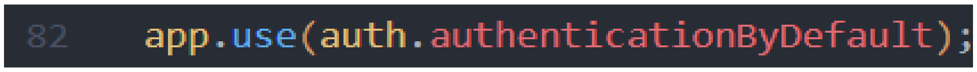
we solved the vulnerability by adding a limit for the time.



```
const apiLimiter = rateLimit({
  windowMs: 5 * 60 * 1000, // 5 minutes
  max: 100, // limit each IP to 100 requests per windowMs
  message: "Too many requests from this IP, please try again later",
});
```



```
app.use(apiLimiter, auth.authenticationByDefault);
```

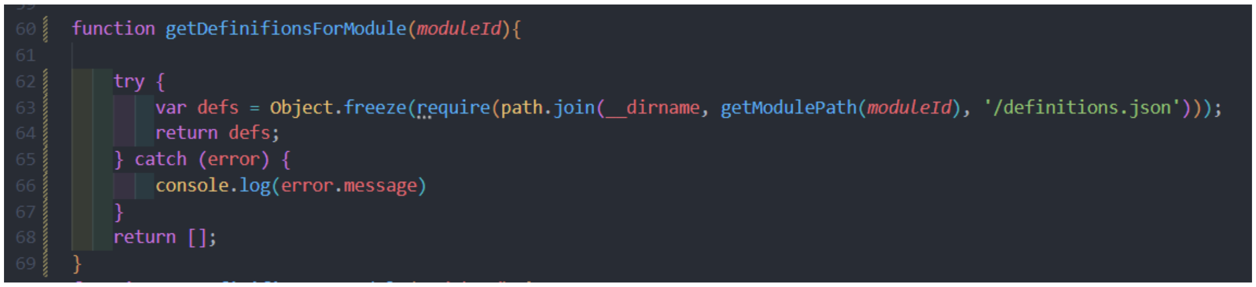


```
app.use(auth.authenticationByDefault);
```

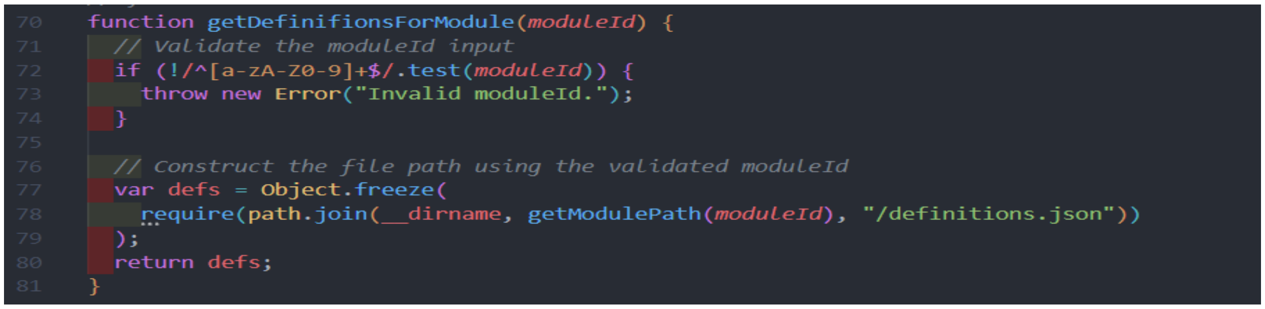
13:Uncontrolled data used in path expression

closed on 27/6/2023

we mada a validation for moduleId value to ensure that it only contains valid characters and does not allow path traversal.



```
function getDefinifionsForModule(moduleId){

    try {
        var defs = Object.freeze(require(path.join(__dirname, getModulePath(moduleId), '/definitions.json')));
        return defs;
    } catch (error) {
        console.log(error.message)
    }
    return [];
}
```



```
function getDefinifionsForModule(moduleId) {
  // Validate the moduleId input
  if (!/^[a-zA-Z0-9]+$/.test(moduleId)) {
    throw new Error("Invalid moduleId.");
  }

  // Construct the file path using the validated moduleId
  var defs = Object.freeze(
    require(path.join(__dirname, getModulePath(moduleId), "/definitions.json"))
  );
  return defs;
}
```

14-Inclusion of functionality from an untrusted source

closed on 6/29/2023

it was unused library so we removed it from our code

46: user data is not saved

Closed on 7/11/2023

We have investigated the problem and tested the web application thoroughly, but we did not find any evidence of data loss or any other issue. It is possible that the problem you experienced was due to a browser cache issue or a temporary network problem.

33: Uncommon header 'alt-svc' found, with contents: h3=":443"; ma=86400

Closed on 7/11/2023

based on our research we found that "alt-svc" header is a normal part of the HTTP/3 protocol and is not something that needs to be fixed or resolved, The "alt-svc" header message you mentioned is not an issue or error message, but rather an informational message that indicates the availability of alternative services for the requested resource. It is a standard part of the HTTP/3 protocol and is not something that needs to be fixed or resolved.

26: Modern Web Application Using Non-Traditional Links

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time

25: Loosely scoped cookie

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time

24: Suspicious Comments in HTTP Response

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time

16: Exposure of private files

Closed on 11/7/2023

this vulnerability can be harmful if the "node_modules/jquery" directory contains private files or sensitive data but there are no private files or sensitive data it is just a js library available on the internet

so it does not affect our code security.

11: Vunerabe JS Libraray

Closed on 11/7/2023

we upgraded the library to the last version but we still had the same issue when we rescaned our url.

45:TCP Timestamps Information Disclosure

Closed on 12/7/2023

After conducting a thorough investigation, we have determined that the issue is likely related to the original code and cannot be resolved through changes to our application code alone. Resolving this issue would require access to the server and changes to the server's configuration. Therefore, we believe that this issue is out of scope.

10:Hidden file was found

Closed on 12/7/2023

we were not able to resolve this issue as it was present in the original code and not introduced during the development of the project.

34:Uncommon header 'x-render-origin-server' found, with contents: Render

Closed on 12/7/2023

We have reviewed the potential risks associated with this header and its associated component "Render". While the presence of an uncommon header is not necessarily a big problem on its own, we understand that if the associated component has known security issues or is misconfigured, it could potentially lead to vulnerabilities.so we requested more information about this issue and did not get a response yet.

23:Server Information Leakage via X-Powered-By HTTP response header

Closed on 12/7/2023

After conducting a thorough investigation, we have determined that the issue is related to the server configuration and cannot be resolved through changes to our application code. The web/application server is leaking information through one or more "X-Powered-By" HTTP response headers, which provides information about the technology stack and software components used to build and run the web application. To resolve this issue, we need to modify the server configuration to remove the "X-Powered-By" header and disable server signature or version information in the HTTP response headers. Therefore, we believe that this issue is out of scope for our application.

issues related to Csp header :

21:Csp: Style-Src Unsafe-Inline Vulnerability

20:Csp: Script-Src Unsafe-Inline Vulnerability

19:Csp: Script-Src Unsafe-Eval Vulnerability

18:Csp: Wildcard Directive Vulnerability

Closed on 12/7/2023

After conducting a thorough investigation, we have determined that these issues are likely related to the original code and cannot be resolved through changes to our application code alone. Resolving these issues would require access to the server and changes to the server's configuration. Therefore, we believe that this issues are out of scope for our application

22:Cookie SameSite Attribute Set to None Vulnerability

Closed on 12/7/2023

we were not able to resolve this issue as it was present in the original code and not introduced during the development of the project.

9:User Agent Fuzzer

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time.

8:HyperText Transfer Protocol(HTTP) Redirect

Closed on 3/7/2023

After further review, we found that this is not a bug or a feature request, but rather an informational issue. We have determined that this issue does not require any action at this time.

7:HTTP Methods Allowed

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time.

6:Syn Open Port Detected

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time.

5:Web Server No 404 Error Code

Closed on 3/7/2023

Upon further investigating , this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time.

4:JQuery Detection

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time.

1:Web Application Sitemap

Closed on 3/7/2023

Upon further review, this is not a bug or a feature request, but rather an informational issue. we have determined that this issue does not require any action at this time.

44:Absence of Anti-CSRF Tokens

Closed on 12/7/2023

we were not able to resolve this issue as it was present in the original code and not introduced during the development of the project.

LESSONS LEARNED

Table 1: Lessons Learned

Lesson Learned	Description	Group-Endorsed Conclusion
Active Vulnerability Scanning	Importance of actively scanning code for vulnerabilities using a range of different tools and techniques	Following best practices is not enough to ensure the security of our code; we need to be proactive in identifying and addressing potential vulnerabilities.
Collaboration and Communication	Importance of effective collaboration and communication within the group	Effective collaboration and communication are essential for ensuring the security of our code.
Importance of Access Control	This lesson learned emphasizes the importance of implementing strong access control measures, including proper authorization and authentication procedures. Discounting the importance of access control can leave your application vulnerable to unauthorized access and potential security breaches.	We concluded that strong access control measures are essential for ensuring the security of our application and that we need to prioritize access control in our future work.

Lesson Learned	Description	Group-Endorsed Conclusion
Documentation and Knowledge Sharing	Value of documenting findings and sharing knowledge with other members of the group	Documenting findings and sharing knowledge with others is an essential part of ensuring the security of our code.
Realistic Goal-Setting	Importance of setting realistic goals and managing expectations	Setting realistic goals and managing expectations is essential for avoiding burnout and maintaining a sustainable approach to code security.
Importance of secure coding practices	The project team learned that following secure coding practices is crucial in preventing vulnerabilities in the code. This includes properly sanitizing user input, using secure authentication and authorization mechanisms, and implementing secure coding patterns.	The group concluded that implementing secure coding practices from the start of the development process is essential in preventing security vulnerabilities and should be a top priority in any software development project.

Lesson Learned	Description	Group-Endorsed Conclusion
The importance of regular vulnerability scanning	The project team learned the importance of regularly scanning the code for vulnerabilities, both in their own code and in third-party libraries and APIs. This helps to identify potential security weaknesses before they can be exploited by attackers.	The group endorsed the recommendation that regular vulnerability scanning should be a standard practice in any software development project, and that the results of these scans should be reviewed and acted upon promptly to ensure the security of the project.
The importance of patching vulnerabilities promptly	The project team learned that it is important to promptly patch any vulnerabilities that are identified, whether through a vulnerability scan or otherwise. Delaying patching can leave the project exposed to potential attacks.	The group endorsed the recommendation that patching vulnerabilities should be a top priority and should be done as soon as possible after a vulnerability is identified.
The risks of using outdated software and libraries	The project team learned that using outdated software and libraries can increase the risk of security vulnerabilities. These vulnerabilities can be exploited by attackers to gain unauthorized access to the system or to steal sensitive data.	The group endorsed the recommendation that all software and libraries used in the project should be kept up-to-date with the latest security patches and updates to reduce the risk of security vulnerabilities.

Lesson Learned	Description	Group-Endorsed Conclusion
The importance of testing for security vulnerabilities	The project team learned that testing for security vulnerabilities is an important step in the software development process. This includes both automated and manual testing, as well as testing for known attack vectors and scenarios.	The group endorsed the recommendation that testing for security vulnerabilities should be an integral part of the software development process, and that the results of these tests should be reviewed and acted upon promptly to ensure the security of the project.
The benefits of using existing code	The project team learned that using existing code, such as libraries, frameworks, and modules, can save time and effort in the development process. Additionally, many of these existing code bases have already been thoroughly tested for security vulnerabilities.	The group endorsed the recommendation that whenever possible, existing code should be used instead of implementing code from scratch. This can help to reduce the risk of security vulnerabilities and save time in the development process. However, it is important to ensure that any existing code used is up-to-date and has been thoroughly tested for security vulnerability.

Table 2: Recommendations

Detailed Recommendations	Description
Implement strong access control measures, including proper authorization and authentication procedures.	This recommendation involves putting in place robust access control measures to ensure that only authorized users can access your application and its resources. This can include implementing secure authentication methods, such as multi-factor authentication, and authorization controls to limit access to sensitive data and functionality.
Use strong password policies and regularly prompt users to update their passwords.	This recommendation involves implementing strong password policies, such as requiring users to choose complex passwords and enforcing regular password changes. This can help prevent unauthorized access to your application and its resources, as well as reduce the risk of password-related attacks, such as brute force and dictionary attacks.
Use automated and manual testing	The group recommends using both automated and manual testing to identify vulnerabilities in the codebase. Automated tools can help identify common vulnerabilities, while manual testing can identify more complex vulnerabilities that may be missed by automated tools.
Keep security in mind throughout the development process	Security should be a top priority throughout the entire software development process, from planning and design to implementation and testing. Developers should be trained in secure coding practices and given the necessary resources to ensure that security is taken seriously in all phases of development.

Detailed Recommendations	Description
Carefully consider website design and flow	We devoted a significant amount of time to carefully consider the design and flow of your website, ensuring that it is both secure and compatible with various systems. Investing the necessary time and effort upfront can help prevent potential security vulnerabilities and compatibility issues down the line, saving significant resources and time that would otherwise be spent on fixing avoidable problems.

Table 3: Things the Group Would Not Do Again and Why, What we would do differently next time

Things we would not do again	Reason	What we would do differently next time
Underestimate the Importance of Active Vulnerability Scanning	This lesson learned highlights the importance of conducting regular vulnerability scans of your application to identify potential security vulnerabilities. Relying solely on best practices and coding guidelines is not sufficient to ensure the security of your code, as new vulnerabilities are constantly being discovered and emerging threats are always a possibility. By conducting active vulnerability scans, you can identify and fix vulnerabilities before they can be exploited by attackers.	Next time, we would develop a regular schedule for vulnerability scanning, including both automated and manual scans. We would ensure that all team members are trained in vulnerability scanning techniques and understand the importance of this step in the development process. We would also prioritize fixing vulnerabilities identified in vulnerability scans, and allocate sufficient time and resources to addressing those vulnerabilities promptly. Additionally, we would establish clear guidelines for reviewing and updating vulnerability scanning procedures regularly, including staying up-to-date with the latest security threats and vulnerabilities.

Things we would not do again	Reason	What we would do differently next time
Delay addressing identified vulnerabilities	The group found that delaying addressing identified vulnerabilities can lead to increased risk of security incidents. The longer a vulnerability remains unaddressed, the more time a potential attacker has to exploit it. Therefore, the group would not delay addressing identified vulnerabilities in the future.	Next time, we would prioritize fixing vulnerabilities promptly, including establishing clear priorities for which vulnerabilities to address first and ensuring that all team members are aware of their responsibilities for addressing vulnerabilities. We would also allocate sufficient time and resources to fixing vulnerabilities, including any necessary testing and validation to ensure that fixes do not introduce new vulnerabilities. Additionally, we would establish clear guidelines for reviewing and updating vulnerability remediation procedures regularly to ensure that identified vulnerabilities are addressed promptly.

Things we would not do again	Reason	What we would do differently next time
Discounting the importance of access control	This lesson learned emphasizes the importance of implementing strong access control measures, including proper authorization and authentication procedures. Discounting the importance of access control can leave your application vulnerable to unauthorized access and potential security breaches. By implementing robust access control measures, you can ensure that only authorized users can access your application and its resources.	Next time, we would prioritize implementing strong access control measures, including proper authentication and authorization procedures, to ensure that only authorized users can access our application and its resources. We would also regularly review and update these measures to ensure that they remain effective and relevant. Additionally, we would ensure that all team members are trained in access control measures and understand the importance of this step in maintaining the security of the application.

Things we would not do again	Reason	What we would do differently next time
Use outdated or unsupported software	The group found that using outdated or unsupported software can lead to security vulnerabilities. It is important to regularly update software and libraries to ensure that security vulnerabilities are patched in a timely manner. Therefore, the group would not use outdated or unsupported software in the future.	Next time, we would establish a process for regularly checking for updates to software and libraries used in the project, and ensure that all team members are aware of the importance of keeping software up-to-date. We would also prioritize updating software and libraries promptly to ensure that any security vulnerabilities are patchedin a timely manner. Additionally, we would establish clear guidelines for reviewing and updating our software and library update procedures regularly to ensure that they remain effective and efficient. We would also consider using a tool or service that can help with identifying and updating outdated software and libraries automatically, to streamline the update process and reduce the risk of missing critical updates.