LATACORA

Fleet Device Management

Application Security Assessment

Confidential

June 2023

Executive Summary

In June 2023, Latacora performed an application penetration assessment of the application from Fleet.

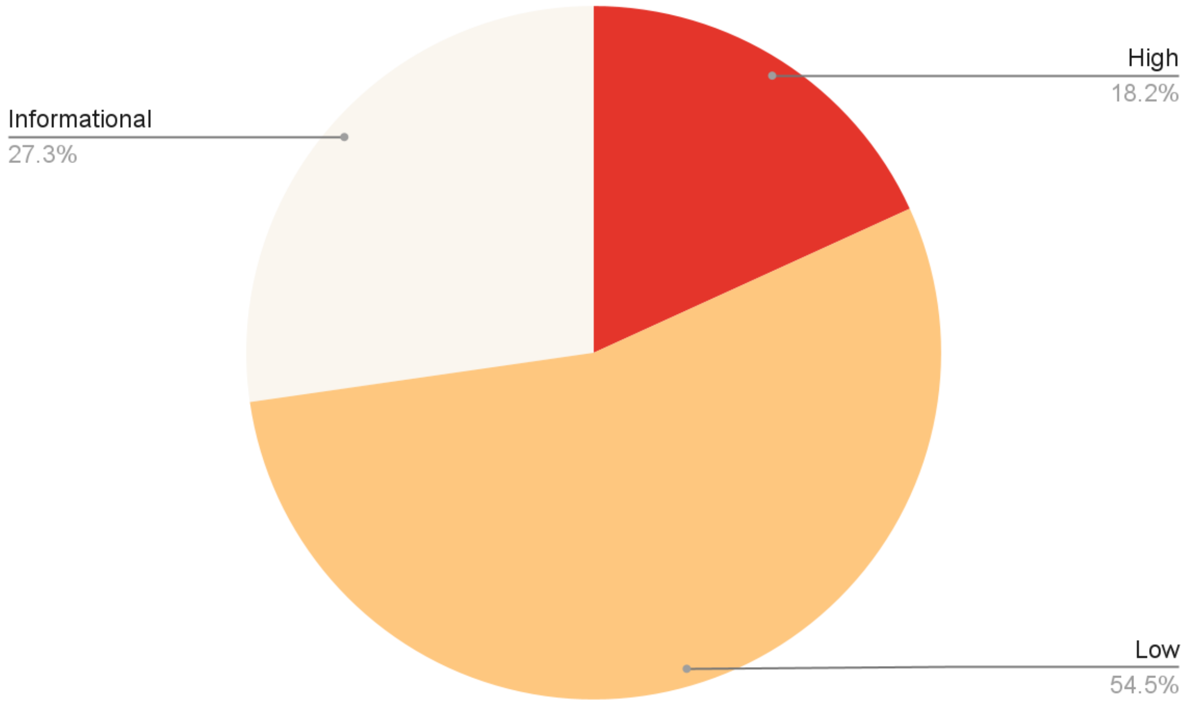
An application penetration test captures a point-in-time assessment of vulnerabilities, misconfigurations, and gaps in applications that could allow an attacker to compromise the security, availability, processing integrity, confidentiality, and privacy (SAPCP) of sensitive data and application resources. An application penetration test simulates the capabilities of a real adversary, but accelerates testing by using information provided by the target company.

This **confidential document** relates the methodology our team used to conduct the assessment, key findings, and a summary of the coverage and findings achieved. The purpose of this report is to capture the assessment work done and the outcome of that assessment.

About Latacora and the Application Security Testing Team

Latacora is a boutique security firm that provides application, network, cloud, and corporate security services to startups. The principal members of Latacora were founders of Matasano Security and executive management at NCC Group, North America's largest software security firm, and Rackspace Managed Security.

Discovered Findings



Testing identified 2 High, 6 Low and 3 Informational severity findings.

Coverage Obtained

Latacora reviewed all application targets except the backend code from a source code review perspective and performed dynamic manual testing of the applications in their respective production environment. For the Windows agent, a code review was performed.

Methodology

Logistics

Testers were given access to Fleet source code repositories as well as access to a production environment running an up-to-date version of the platform, with credentials sufficient to replicate all the levels of privilege involved in the defined testing scope.

Scope

The scope of this assessment included the following web application:

- https://sectest.cloud.fleetdm.com/

Discovery

At the start of an assessment, we identify and classify risks within the business context of the application.

In an application security assessment of a single coherent software target, the goals of Discovery include:

1. **Enumerate pre-authentication attack surface**: we seek to separate the portion of the application attack surface that doesn't require credentials from the portion that does. Pre-auth attack surface is especially exposed to "OWASP Top 10"-style web attacks.
2. **Identify administrative functionality**: we locate in the code any functionality limited to administrators and determine the code paths used to control access to that code. Administrative functionality is especially sensitive to authorization and privilege escalation attacks, and sometimes, with a threat model that includes trusted administrators, less sensitive to certain other classes of attacks.
3. **Map authorization controls**: we build an understanding of the levels of access different users have, and the application-layer controls that enforce those levels of access. A major focus of application security testing is statically and dynamically exercising these boundaries.
4. **Catalog dependencies**: modern web applications are almost invariably built on collections of third-party libraries and Fleet is not an exception. Dependencies must be analyzed for known vulnerabilities. Additionally, dependencies of security significance – those including memory-unsafe C code, or which enforce security boundaries – receive direct inspection as a task in the assessment.

5. **Classify data**: the sensitivity of data stored in financial services applications is blended. All data accesses need to be assured for integrity and database safety. Sensitive nonpublic personal information (NPI) needs further assurances, such as at-rest data protection and additional authorization controls.

Because we had access to source code, we were able to straightforwardly enumerate all exposed endpoints (frontend and API), break down which portions of the backend they implicated, and evaluate how they were authenticated.

Testing Goals

Application Security Testing Goals	
Authentication	◦ Pre-authorization attack surface is cataloged and verified ◦ Password form fields are of type password ◦ Credentials are delivered over TLS ◦ Credentials are not inadvertently disclosed to a third party, on client or server ◦ Strong password complexity guidance and validation ◦ Brute force password countermeasures ◦ Password reset hygiene ◦ Email address verification
Authorization	◦ Privilege escalation ◦ Co-tenant segregation ◦ Privilege change requires additional authentication ◦ Mass assignment ◦ Verb tampering ◦ Consistency of authorization across applications ◦ Information leakage/disclosure ◦ Rate limiting ◦ File handling
Session Management	◦ Consistent, logically centralized session management ◦ Cryptographically opaque client-visible session IDs. ◦ Proper session invalidation ◦ Session timeout ◦ Session fixation ◦ Session equivalent cookies cataloged, audited and validated ◦ SSO controls are consistent with application controls
Data Storage	◦ SQL query safety and ORM usage ◦ User inputs are typecast, sanitized and/or parameterized ◦ Encryption ◦ Non-SQL or non-database data storage (such as file systems)
Server Configuration and Deployment	◦ Exposed service ports ◦ Transport encryption, TLS and cipher suites ◦ Certificate expiry, validation and pinning ◦ HSTS ◦ Fingerprinting software versions ◦ File upload and post-processing ◦ Application extensions

	◦ Header splitting ◦ Web application firewall ◦ Host headers ◦ Server side request forgery
Application Logic	◦ Format of currency values ◦ Handling of payments and refunds ◦ Validation of awards, credits, or discounts ◦ Application can be used to validate stolen credit cards ◦ Altering prices or other payment amounts ◦ Canceling orders after fulfillment
Cryptography	◦ Encrypted data is authenticated, using a whitelist of allowed AEAD constructions. ◦ Ciphers are selected from a whitelist of allowed ciphers. ◦ Exposed encrypted tokens and capability grants are not reusable. ◦ Exposed encrypted tokens are time stamped and checked for expiry.
Application Hygiene	◦ Exposed stack traces ◦ Exposed server side artifacts such as path names ◦ Unused code paths ◦ Debugging code or other information ◦ Exposed server side source code ◦ Redirect handling
Administrative Functionality	◦ Administrative functions are out of band ◦ Administrative resources require 2FA ◦ Cross-site request forgery ◦ Reflected and stored Cross-site scripting (XSS) ◦ Audit trails and logging ◦ User impersonation features hygiene

Risk Ratings

When a risk is identified, Latacora calculates an impact or risk rating score as part of documenting the finding. Impact can be broken down into factors aligned with the traditional security areas of concern: security, availability, processing integrity, confidentiality, and privacy. The goal is to estimate the magnitude of the impact on the system if the vulnerability were to be exploited. The following areas of concern (abbreviated SAPCP) are:

- **Security** - which addresses the resistance of the application and connected systems to hostile attacks, such as DDoS attacks, injection attacks, account takeover attempts, server-side forgery, and direct object references.
- **Availability** - which addresses service interruption and the critical nature of down time, as well as the adequate provisioning of resources.
- **Processing Integrity** - which addresses complete, correct, accurate, and timely processing of data
- **Confidentiality** - which addresses storage and handling of data, as well as how much and what kind of data should be disclosed or retained.
- **Privacy** - which addresses the requirement for notice to data subjects about data collection, choice and consent to data collection and access to collected data, and the system's use, retention, disclosure, and disposal of personal information according to the applicable regulations.

Determining Severity

The severity of each risk is determined by analyzing likelihood along with impact. This table describes the severity scale used throughout Latacora's assessments:

Classification	Description
INFORMATIONAL	Not exploitable. Potentially no action required.
LOW	Limited impact on SAPCP. High awareness/easily detected. Difficult to exploit or not directly exploitable. May be useful for reconnaissance. Potentially no action required.
MEDIUM	Moderate impact on SAPCP. High awareness/possibly detected. Difficult to exploit. Some logging, some monitoring.
HIGH	Significant impact on SAPCP. Low awareness/difficult to detect.
CRITICAL	Serious impact on SAPCP. No awareness/no detection. Suggest immediate remediation.

Summary of Findings

Reference	Description	Severity
LT-APP23-1	Stored Cross-Site Scripting (XSS) in Tooltip	High
LT-APP23-2	Broken Authorization Leads to Observers Able to Add Hosts	High
LT-APP23-3	Missing Server-Side Validation on Secrets Value	Low
LT-APP23-4	Lack Of Validation on All URLs	Low
LT-APP23-5	Cookies Set to Parent Domain	Low
LT-APP23-6	IP Spoofing via X-Forwarded-For	Low
LT-APP23-7	Missing HTTP Security Headers	Low
LT-APP23-8	Systemic Blind Server-Side Request Forgery (SSRF)	Low
LT-APP23-9	Open Redirect in SSO Flow	Informational
LT-APP23-10	Observers Can See All Scheduled Queries	N/A
LT-APP23-11	Directory Listing in Assets	Informational
LT-APP23-12	Improper Validation of User Privileges in Windows System	Informational

Remediation Roadmap Guidance

Reference	Description	Timeline
LT-APP23-1	Stored Cross-Site Scripting (XSS) in Tooltip	Within 30 days
LT-APP23-2	Broken Authorization Leads to Observers Able to Add Hosts	Within 30 days
LT-APP23-3	Missing Server-Side Validation on Secrets Value	Within 60 days
LT-APP23-4	Lack Of Validation on All URLs	Within 90 days
LT-APP23-5	Cookies Set to Parent Domain	Within 90 days
LT-APP23-6	IP Spoofing via X-Forwarded-For	Within 90 days
LT-APP23-7	Missing HTTP Security Headers	Within 90 days
LT-APP23-8	Systemic Blind Server-Side Request Forgery (SSRF)	Within 90 days
LT-APP23-9	Open Redirect in SSO Flow	Within 90 days
LT-APP23-10	Observers Can See All Scheduled Queries	Within 30 days
LT-APP23-11	Directory Listing in Assets	Within 90 days
LT-APP23-12	Wrong Validation of User Privileges in Windows System	Within 180 days

Detailed Findings

LT-APP23-1 Stored Cross-Site Scripting (XSS) in Tooltip

Vulnerability Scoring

Latacora Severity	High

Finding Description

All tooltips using the `tipContent` tag are set using `dangerouslySetInnerHTML`. This allows manipulation of the DOM without sanitization. If a user can control the content sent to this function, it can lead to a cross-site scripting vulnerability. There's at least one (1) occurrence where this happens, but we suggest making sure no other React components could be vulnerable.

The occurrence is in the Bundle Identifier from the Software tab.

If a malicious user is enrolled and puts malicious content in the `Info.plist` of an installed application, this will trigger an XSS.

Here's an example of a `Info.plist` file containing a malicious payload for the `osqueryd` application.

Unset
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
	<key>CFBundleExecutable</key>
	<string>osqueryd</string>
	<key>CFBundleIdentifier</key>
	<string><![CDATA[io.osquery.agent<img src=x
onerror=alert(1)>]]></string>
	<key>CFBundleInfoDictionaryVersion</key>
	<string>6.0</string>
	<key>CFBundleName</key>
	<string>osqueryd</string>
```

```
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleShortVersionString</key>
        <string>5.8.3</string>
        <key>CFBundleVersion</key>
        <string>5.8.3</string>
</dict>
</plist>
```

Vulnerability Location

https://sectest.cloud.fleetdm.com/software/manage

Remediation Recommended

It is suggested to avoid using `dangerouslySetInnerHTML` completely. If this cannot be done, we recommend reviewing all usages of the `/frontend/components/TooltipWrapper/TooltipWrapper.tsx` component and making sure none of them contains data that could be manipulated by a user.

LT-APP23-2 Broken Authorization Leads to Observers Able to Add Hosts

Vulnerability Scoring

Latacora Severity	High

Finding Description

Observers are not supposed to be able to add hosts to Fleet. Via specific endpoints, it becomes possible to retrieve the certificate chains and the secrets for all teams, and these are the information required to add a host.
While the endpoint `/api/latest/fleet/spec/enroll_secret` has a proper authorization control and will return 401s, both endpoints `/api/latest/fleet/teams` and `/api/latest/fleet/teams/{team_id]/secrets` can be accessed by an observer. It wasn't possible to retrieve the secret for the "No team".

The endpoints to get the certificate and public and secret keys also suffer from the same authorization issue.

Vulnerability Location

https://sectest.cloud.fleetdm.com/api/latest/fleet/config/certificate
https://sectest.cloud.fleetdm.com/api/latest/fleet/teams/{team_id]/secrets
https://sectest.cloud.fleetdm.com/api/latest/fleet/mdm/apple/dep/key_pair
https://sectest.cloud.fleetdm.com/api/latest/fleet/teams

Remediation Recommended

We recommend reviewing the permissions set for those endpoints and restricting them to Maintainer and Admin users. Overall authorization posture would be improved by adding tests to cover these types of issues and making sure no regression occurs .

LT-APP23-3 Missing Server-Side Validation on Secrets Value

Vulnerability Scoring

Latacora Severity	Low

Finding Description

Clients can create multiple secrets used to enroll hosts. While the form mentions a secret must be at least 32 characters long, this validation is not enforced server-side. A user could force a secret to have a value of "1234". This could lead to easily guessable secrets and malicious users could try to bruteforce the secret to enroll any host.

Vulnerability Location

https://sectest.cloud.fleetdm.com/api/latest/fleet/teams/1/secrets

Remediation Recommended

It is recommended to also perform a server-side validation to make sure the secret contains at least 32 characters.

LT-APP23-4 Lack Of Validation on All URLs

Vulnerability Scoring

Latacora Severity	Low

Finding Description

The application lets users set a few URLs, such as URLs for Gravatar, an Organization, the Custom Transparency and the Metadata. None of these validate that the URL is well formed. Although it didn't lead to a XSS, it is still dangerous, as a user could try to inject `javascript:alert(1)` as an URL.

Vulnerability Location

https://sectest.cloud.fleetdm.com/settings/integrations/automatic-enrollment
https://sectest.cloud.fleetdm.com/settings/organization/info
https://sectest.cloud.fleetdm.com/settings/organization/sso
https://sectest.cloud.fleetdm.com/api/latest/fleet/users/{user_id}

Remediation Recommended

It is recommended to also perform a server-side validation to make sure all URLs are valid.

LT-APP23-5 Cookies Set to Parent Domain

Vulnerability Scoring

Latacora Severity	Low

Finding Description

fleetdm.com sets a few cookies related to Cookie Eyes. Because they are scoped to the parent domain, they are also sent to subdomains. Clients with a cloud-hosted instance of Fleet will send cookies to their instances.

While this doesn't lead to a security issue, it's best practice to correctly set the domain to all cookies. If a sensitive cookie was set from fleetdm.com, this could be more dangerous, as users could access it.

Vulnerability Location

All of these cookies can be accessed from subdomains:
cky-action, cky-consent, cookieyes-necessary, cookieyes-functional, cookieyes-analytics, cookieyes-performance, cookieyes-advertisement, cookieyes-other

Remediation Recommended

It is recommended to not set any domain when the cookie is set.

LT-APP23-6 IP Spoofing via X-Forwarded-For

Vulnerability Scoring

Latacora Severity	Low

Finding Description

The application doesn't properly validate the origination of the IP and it can be spoofed via the X-Forwarded-For header. This request header can be easily modified by users. Moreover, any data can be sent, so having a payload like `X-Forwarded-For: <svg/onload=alert(1)>` is accepted by the server and could trigger vulnerabilities in downstream systems.

This can become tricky when an investigation is ongoing and it is not possible to verify IP addresses.

Vulnerability Location

*.fleetdm.com

Remediation Recommended

It is recommended to use the `request.RemoteAddr` function and make sure it is a valid IP address using the `net.ParseIP(ip)` function.

LT-APP23-7 Missing HTTP Security Headers

Vulnerability Scoring

Latacora Severity	Low

Finding Description

The application server is not using some common HTTP security headers. Each of these headers instructs the user's browser to enforce additional protections that help prevent exploitation of some vulnerabilities or misconfigurations.

- `Strict-Transport-Security`: this header prevents a browser from making any future unencrypted non-HTTPS connections to the site. This prevents accidental transmission of credentials or sensitive information in plaintext if a user visited an `http://` URL pointing to the site. This header should be enabled with a suitable lifespan value, ideally 1 year for a production instance.
- `X-Frame-Options`: this header can prevent application pages from being displayed in iframes on third-party sites. Framing can be abused to trick users into taking actions in the application; this is known as a 'clickjacking' attack. Setting `X-Frame-Options: deny` prevents any framing, while `X-Frame-Options: sameorigin` only allows frames on the same domain.
- `Content-Security-Policy`: this header can enforce many browser security features, but it's most well-known as a protection against XSS because it can block execution of inline JavaScript. Enabling this header is more difficult than others as it requires trial-and-error and may require code changes, but it brings solid benefits when implemented.

The console should implement Strict-Transport-Security as soon as possible and investigate adding the X-Frame-Options and Content-Security-Policy in the mid-term.

Vulnerability Location

*.fleetdm.com

Remediation Recommended

It is recommended to implement all three headers mentioned. As the CSP could potentially break the UI, it is possible to test it first using the report-to attribute. The CSP Evaluator can also be used to test the robustness of the CSP.

LT-APP23-8 Systemic Blind Server-Side Request Forgery (SSRF)

Vulnerability Scoring

Latacora Severity	Low

Finding Description

Server-side request forgery occurs when the application is coerced into making connections to arbitrary downstream systems. This can be used to pivot into the application's internal network or leverage trust relationships that the application server may have with other systems. In a blind SSRF, the application server will send a request to an attacker-specified system but does not return the downstream system's response to the attacker.

When a user sets up SSO, the application will take the Metadata URL and perform a request. Any error is returned to the user. A user can enumerate whether the request succeeded, timed out or if there was an internal error.

The same issue is found when a user adds an Integration.

Vulnerability Location

https://sectest.cloud.fleetdm.com/settings/integrations
https://sectest.cloud.fleetdm.com/settings/integrations/automatic-enrollment
https://sectest.cloud.fleetdm.com/settings/organization/sso

Remediation Recommended

It is recommended to validate all user-supplied URLs to ensure the application does not send requests to internal systems. Validation should resolve the URL and ensure the DNS records point to an external resource directly prior to issuing the request. If possible, requests should use a proxy server with strict network controls, such as SmokeScreen (https://github.com/stripe/smokescreen), that only allows traffic to external services.

LT-APP23-9 Open Redirect in SSO Flow

Vulnerability Scoring

Latacora Severity	Informational

Finding Description

When a user logs in using the SSO flow, a POST parameter is used to redirect the user back to the application. This relay_url can be set to any URL and thus could be redirected to a malicious website.

The user is only redirected when the authentication succeeds. Since it's not possible to force a user to fill a form and do the POST request to the endpoint without the user already being on a malicious website, this isn't currently exploitable.

Vulnerability Location

https://localhost:8080/api/v1/fleet/sso

Remediation Recommended

It is recommended to make sure the user can only be redirected to the Fleet application. There's a few ways to do this:

- Force the user to be redirected to the index page.
- Save the page that the user tried to visit in the backend and then redirect them after authentication.
- Force the `Location:` HTTP header to be set to the app.fleet.com/something.

LT-APP23-10 Observers Can See All Scheduled Queries

Vulnerability Scoring

Latacora Severity	N/A

Finding Description

Observers can run scheduled queries that have the appropriate "Observers can run" attribute enabled. While the UI only shows scheduled queries having this attribute, an endpoint allows an Observer to see all scheduled queries.

Vulnerability Location

https://sectest.cloud.fleetdm.com/api/latest/fleet/schedule

Remediation Recommended

It is recommended to make sure only scheduled queries having the "Observers can run" attribute set are shown.

LT-APP23-11 Directory Listing in Assets

Vulnerability Scoring

Latacora Severity	Informational

Finding Description

The /assets endpoint reveals all existing files located in this folder and subfolders. This can aid an attacker to identify all endpoints and ease further attacks. This could also be used to get access to specific files that shouldn't be exposed.

Vulnerability Location

https://sectest.cloud.fleetdm.com/assets

Remediation Recommended

It is recommended to either create an index file or to implement a custom middleware, as Go doesn't provide an easy way to disable the directory listing.

LT-APP23-12 Improper Validation of User Privileges in Windows System

Vulnerability Scoring

Latacora Severity	Informational

Finding Description

During the enrolment process on Windows systems, the application checks if the current user has administrator privileges. The way it validates the privileges is prone to errors. At worst, it could lead to a local privilege escalation.

The application checks if the current user has read privileges on the PHYSICALDRIVE0 which is a hard drive and should be, by default, where the operating system is installed. A user could have read privileges on this drive if the OS is installed on another drive. This could lead to an ambiguous situation where the user isn't an administrator but still passes this validation.

Unset

```
// checking for administrator rights
func checkForAdmin() bool {
     f, err := os.Open("\\\\.\\PHYSICALDRIVE0")
     if err != nil {
          return false
     }

     f.Close()

     return true
}
```

Vulnerability Location

fleet/tools/mdm/windows/programmatic-enrollment/main.go, lines 29-38

Remediation Recommended

It is recommended to change the way the verification works. Windows offers a few ways to perform this validation. The `isUserAdmin` function can be used as a temporary solution as Microsoft suggests to roll out a custom function using the `CheckTokenMembership` function. Another way could be to verify the user privileges with the domain controller, using `NetUserGetInfo`. Another solution could be to change the Manifest file and add the tag `requestedExecutionLevel` and set the level to `requireAdministrator`.

Strategic Recommendations

MDM Secret Handling for Cloud

For customers using the Fleet-cloud-hosted solution, there is currently no support in the Fleet interface for uploading MDM secrets. (Self-hosted customers are intended to edit some environment variables and relaunch the fleet service.) As a result, cloud customers currently must arrange ad hoc secure transport of those secrets to Fleet Operations staff.

Latacora's testing cannot verify the consistency of handling practices in such cases, such as ensuring that secrets are not left behind in ~/Downloads on a Fleet staff laptop.

Latacora recommends something that is almost certainly already on the product roadmap: implement in-product support for the upload of such secrets. Code review/testing would be able to confirm the security of such a workflow.

Remediation Roadmap Guidance

This section encompasses our recommendations for a suggested timeline to fix the issues presented within this report. There are many factors that contribute to our overall risk ratings, and severity is not always the best indicator of how promptly one should fix identified issues.

Short Term (30-90 days)

- Stored Cross-Site Scripting (XSS) in Tooltip
  - XSS can be very dangerous, especially in the situation where the MDM might be used to access remote computers and the fix is quick to implement.
- Broken Authorization Leads to Observers Able to Add Hosts
  - As the authorization scheme is already well implemented and using a fail-close implementation, modifying the current permissions should be easy.
- Observers Can See All Scheduled Queries
  - As the authorization scheme is already well implemented and using a fail-close implementation, modifying the current permissions should be easy.
- Missing Server-Side Validation on Secrets Value
  - Adding a server-side validation should only be a few lines of code.

Medium Term (90-180 days)

- Lack Of Validation on All URLs
  - Ensuring all URLs are well formed could take some time and there's no exploitable cases.
- IP Spoofing via X-Forwarded-For
  - It could be tricky to be able to get the real IP address from a visitor, but implementing it will be really helpful in case of incident handling.
- Missing HTTP Security Headers
  - Adding security headers can really mitigate some risks, and a good CSP could prevent an XSS from being exploitable. However, it can be difficult to implement it without breaking the website.
- Systemic Blind Server-Side Request Forgery (SSRF)
  - SSRF can be really dangerous, but are also hard to fix since they require changes to the infrastructure.
- Open Redirect in SSO Flow
  - This one could be quick and easy to fix or take more time depending on how it is approached. If the SSO flow is modified, it will take more

    time, but if only the Location HTTP header is changed, it should be quick.

- Directory Listing in Assets
    - Creating a robust helper function could take some time but will guarantee a long-term fix, while adding an index file might not be ideal since it is prone to mistakes if a developer forgets to add them in subdirectories.

Long Term (180 days+)

- Cookies Set to Parent Domain
    - This may require contacting the third party to get a fix.