DEIMOS

This is an example report

Example

Penetration Test Report

<Day & Month> 2021

deimos.io

Contents

1. Executive Summary

We have conducted a black-box security audit on the staging environments specified in the target table below, by doing scans on the domain, as well as doing manual investigation and checks on the API endpoints that were exposed by interacting with the frontend applications.

The table below represents a summary of the findings and their risk levels:

Risk Level	Number of Vulnerabilities Found
High	11
Medium	5
Low	3
Info	1

The table below represents the target domains covered in the penetration test:

Target	Number of Vulnerabilities Found
Example-staging.example.co.za	18

Security Misconfiguration

Attackers will always aim to gain technical knowledge about your underlying systems in order to exploit them. By exposing error messages, API documentation, and configuration, an attacker is able to gain info on components used by your system, and in turn, find vulnerabilities in component versions and runtimes.

The latest security patches should always be applied to systems and outdated versions of components should not be run in production or public-facing environments.

The following areas we found were affected with regards to security misconfiguration:

- Web server and various library versions containing known vulnerabilities.
- Django routing error pages showing in staging, exposing all API URLs.
- Django REST interface publicly accessible showing available HTTP verbs on all endpoints.
- Error pages on staging exposing application configuration.
- Directory traversal supported on Apache2 virtual host(s).

Broken Access-Control

A common attack surface of applications would be the authentication and authorization implementation. Exploiting any functionality that is not protected by authentication, or allows for unauthorized modification can result in the exposure of sensitive information.

The following was found during testing:

- API endpoints allowing unauthenticated access.
- Exposed authentication credentials.
- Access to generated reports without authentication.
- Internal resources have public-facing IP addresses/DNS records.
- File uploads allowed on authenticated endpoints allowing for malicious attacks.

Sensitive Data Exposure

Sensitive data exposure occurs when resources leak implementation detail and show attackers' possible attack vectors that can be potentially exploited. Public-facing domains in any environment should avoid exposing implementation details that can be used to gain insight into APIs and underlying frameworks and tools. Sensitive data exposure also occurs when user personal identifiable information is compromised.

The following was found during testing:

- API (Swagger) documentation publicly available.
- List of passwords publicly exposed.
- Slack webhook service URL containing secrets is exposed.
- Admin email addresses exposed.
- Web Server and framework versions exposed.
- Personal identifiable information relating to users are exposed on unauthenticated endpoints.
- Customer information relating to policies.

Exclusions

The following was not covered in this assessment:

- Informational recommendations - Due to the amount of high and medium vulnerabilities found, we have decided to focus efforts on reporting and advising on these severities. Informational recommendations relating to security best practices have been omitted.

2. Assessment

2.1 Apache2 Version has Known Vulnerabilities

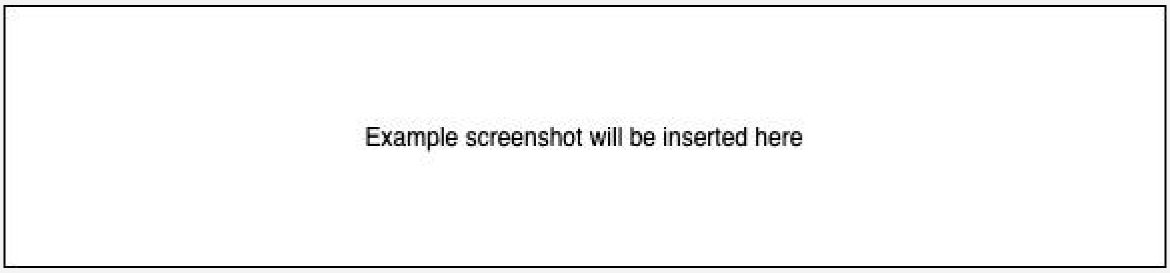
Risk	OWASP Top Ten	Target
High	A9	example-staging.example.co.za

2.1.1 Finding

The Apache2 server version currently has 18 known CVEs that range from remote code execution, denial of service, and overflow exploits.

2.1.2 Evidence

The following is returned when accessing a page that exposes default Apache2 web server pages.



Example screenshot will be inserted here

2.1.3 Risks

Since the web server's version is detectable, this will make for an attack vector where there are exploit tools commonly available. Running outdated vendor software in an environment puts the infrastructure at risk of attacks. The other risk of these types of vulnerabilities is that automated scanners are able to easily detect these kinds of vulnerabilities with minimal to no manual checking.

2.1.4 Recommendations

Always ensure that security patches are applied to web servers that are public-facing. Use managed services over self-hosted platforms where possible to avoid security maintenance whereby periodic patching is required and response to released vulnerabilities.

Apache2 Security Advisories

2.2 Nginx Version has Known Vulnerabilities

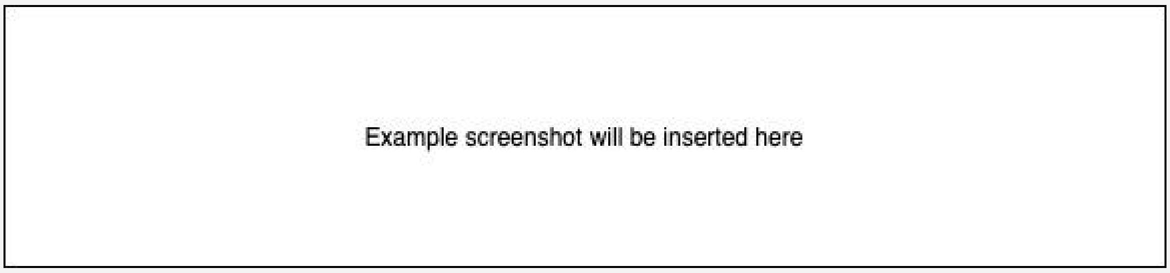
Risk	OWASP Top Ten	Target
Medium	A9	example-staging.example.co.za

2.2.1 Finding

The Nginx server version currently has 3 known CVEs that relate to Denial of Service exploits.

2.2.2 Evidence

The following shows the version of Nginx used, as well as the HTTP/2 protocol enabled which is susceptible to multiple exploits for the version in use.



Example screenshot will be inserted here

2.2.3 Risks

As with the version of Apache2, the same risks apply whereby outdated vendor software will be susceptible to exploits due to the vulnerabilities being public knowledge and has automated exploits available.

2.2.4 Recommendations

Always ensure that security patches are applied to web servers that are public-facing. Use managed services over self-hosted platforms where possible to avoid security maintenance whereby periodic patching is required and response to released vulnerabilities.

Nginx Security Advisories

2.3 Exposure of Sensitive Credentials

Risk	OWASP Top Ten	Target
High	A3	example-staging.example.co.za

2.3.1 Finding

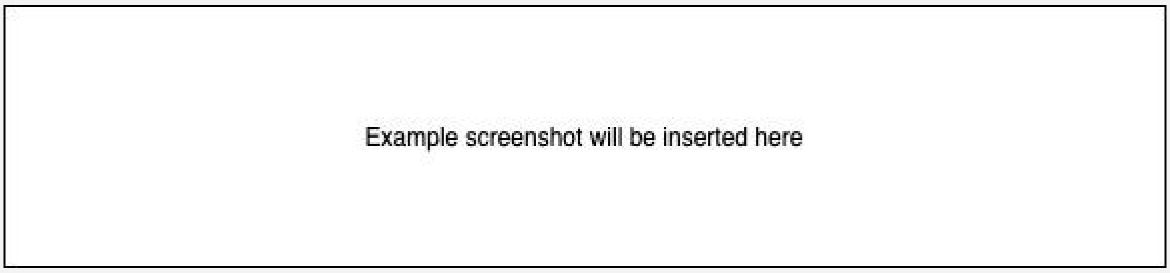
It has been discovered that RabbitMQ credentials for staging have been leaked via application configuration being exposed to the frontend.

2.3.2 Evidence

Doing an HTTP POST on the following API endpoint results in an error page displaying application configuration:

```
https://example-stg3.example.com/api/v1/hello-world
```

Upon triggering a 5xx status code on the API throws an error page exposing the configuration object, the RabbitMQ credentials are not masked as it is present in the connection string (password redacted).


Example screenshot will be inserted here

2.3.3 Risks

Leaking of sensitive credentials can lead to the underlying component being compromised and used for by an attacker maliciously.

2.3.4 Recommendations

It is recommended that AMQP credentials be stored in a connection object instead of existing in the connection string. Django error logging should not be verbose in any environment that is public-facing. To mitigate potential risk, RabbitMQ should only be accessible over a private network and should not have a public Elastic IP.

RabbitMQ credentials should be assumed compromised and be rotated as soon as possible.

OWASP - Plaintext password exposure

2.4 Exposure of Backend Implementation

Risk	OWASP Top Ten	Target
High	A3	example-staging.example.co.za

2.4.1 Finding

It has been discovered that application configuration is publicly exposed, showing application implementation details and server configuration.

2.4.2 Evidence

Doing an HTTP POST on the following API endpoint results in an error page displaying application configuration:

```
https://example-staging.example.com/api/v1/hello-world
```

The following is exposed in configuration, and was made visible via blackbox testing:

Database	RDS endpoint is exposed, database engine is PostgreSQL. Database name and user is exposed.
Stack	Using Python 3, Django REST, Apache Mod WSGI.
Email addresses	Multiple admin email addresses are exposed.
AMQP	Using rabbitmq for pub-sub.
Notifications	Using a slack bot to send notifications to an administration channel
Session	Using a slack bot to send notifications to an administration channel.

The following outdated code repository in GitHub is publicly accessible. Although containing very little around configuration, it does share the same static folder structure as the sites that were tested and align with the technology stack mentioned above:

```
https://github.com/example/example
```

2.4.3 Risks

An attacker can reverse engineer the full stack and apply exploits based on the technologies. Exposing managed service URLs and 3rd party integration points also puts the system at risk for being targeted.

2.4.4 Recommendations

All environments should not expose verbose logging, and leak implementation detail to the frontend. All backend service implementation details like language used, database used and all components should not be public knowledge as it imposes a risk on the security posture of the system as it can be used to exploit the system.

2.5 Unauthenticated API Endpoints

Risk	OWASP Top Ten	Target
High	A2	example-staging.example.co.za

2.5.1 Finding

There are multiple API endpoints without authentication required. This varies with certain HTTP verbs allowed while others are restricted. All endpoints allow for introspection by means of OPTIONS requests, and can indicate what content types, and HTTP verbs are supported by all endpoints.

2.5.2 Evidence

The following API endpoints were found to have no authentication on the staging site tested:

https://example-staging.example.com/api/v1/hello-world/

The following API endpoints are publicly accessible on staging and allows all HTTP verbs (GET, POST, PATCH) possible:

https://example-staging.example.com/api/v1/hello-world/
https://example-staging.example.com/api/v1/users/
https://example-staging.example.com/api/v1/settings/
https://example-staging.example.com/api/v1/documents/

2.5.3 Risks

API endpoints can be used to gain insight into business logic and processes, and allow for retrieval of sensitive information, or add malicious data to the system that can lead to cross-site scripting attacks.

2.5.4 Recommendations

An API that is only consumed by authenticated applications should not contain any public endpoints. Exposing endpoints without authentication puts the API at risk of having mixed authentication rules, and endpoints relating to sensitive information being exposed by misconfiguration.

It is recommended that an API gateway be used to be a single point of entry for an API, and abstract authentication so that all endpoints are covered by authentication.

The Developer and
Security Operations Company

—

50 Canterbury St
Zonnebloem
Cape Town

deimos.io