Supplier Shield Web Application
Gray Box Penetration Testing Report

CyberNeith

Table of contents

Executive summary

Introduction

In accordance with requested services, we conducted gray box penetration test of the Supplier Shield web application and its public facing infrastructure associated with the application.

The purpose of this penetration testing was to determine security vulnerabilities.

Scope

Scope has been defined as the web application in a staging environment at staging-app.suppliershield.com, and its API located at stg-api.suppliershield.com.

The application allows creating new users, so some users were created during the penetration testing for conducting some of the tests.

staging-app.suppliershield.com is using Amazon Cloudfront, and stg-api.suppliershield.com is using Amazon API gateway. The real IP addresses of the front end and the API servers have not been found, because of which they were not scanned for open ports and services.

Approach

The objective is to assess the security and the effectiveness of security controls of the web application. The penetration testing used a combination of attacks and penetration techniques and technology exploits to uncover different kinds of vulnerabilities from the application and the underlying API.

Methodology

All testing was executed in several related phases.

1. In the planning phase, the rules of engagement, scope, and testing goals were identified and set.

2. The discovery phase included vulnerability scanning along with manual testing to explore and understand the testing target and find vulnerabilities.

3. The attack phase comprised efforts to exploit any vulnerabilities detected, and to synthesize knowledge gained about the environment, its technology, users and function into an escalation of privilege beyond that intended by the Client.

4. The final phase documented all findings in a manner that supports risk assessment and remediation by the Client.

The categorization of the web-based vulnerabilities is done based on OWASP web application security testing methodology.

Brief details of the vulnerabilities, as well as remediation suggestions are presented in the vulnerability summary section.

Vulnerability rating system

Each vulnerability is rated according to the risk it poses, as well as the risk of being dangerous in combination with other vulnerabilities. The rating of vulnerability is Informational, Low, Medium, High, Critical.

Summary of results

During the penetration testing some Informational vulnerabilities were found. The vulnerabilities found in the previous penetration testing have been fixed.

Below the vulnerability numbers per category and risk level are presented.

	Informational	Low	Medium	High	Critical
Application vulnerabilities	2	0	0	0	0

General conclusions

We conclude that overall security level can be graded as satisfactory for the application and its public facing infrastructure as only some Informational severity vulnerabilities were found. It is important to note that all the previous vulnerabilities (except two Informational) from the previous penetration testing report have been fixed.

Although no direct way was found to fully compromise the system or to access users or data of another company, the sections accessible only for administrators or accessible only for the professional evaluation team have not been tested.

It has been found that the backend uses Python/Django stack, as well as Django Rest Framework, but no exact versions were found. It is advised to keep all software up to date.

It should be taken into consideration that this is a combination of black and gray box penetration testing with limited user privileges, so there is some possibility that there can be vulnerabilities in the sections with more privileged access.

Penetration testing results

Overview

An intercepting proxy has been used for intercepting the requests and responses of the web application. The revealed API endpoints were tested for different vulnerabilities.

The front end is written in Vue.js framework, and the source code is minified and packed into a single file.

Web application backend is written in Python/Django stack, Django Rest Framework is also being used. The servers of the application are hosted on Amazon, the server that serves front end files uses Amazon Cloudfront, and the backend API server uses Amazon API gateway. The real servers behind these services were not uncovered.

Amazon S3 is being used as a file storage for supplier logo, company logo and user profile image, as well as the attachments in the assessments. Because of the observed response headers, it is possible that S3 is also being used to host and serve front end files.

For convenience, the vulnerabilities are sorted based on the vulnerability rating.

Javascript files have outdated libraries, which may have vulnerabilities (Informational)

Some of the dependency libraries (like jQuery) included with the application are outdated and may have vulnerabilities. Although in practical sense exploiting such vulnerabilities may be very hard or even impossible, and the impact will be limited to front end, it is always a good idea to update all pieces of software, including client-side libraries.

All Javascript files are shown on the front page without requiring authentication (Informational)

The minified Javascript files that are shown on the unauthenticated front page of the web application contain the code for the whole application. Thus, anyone can download and inspect the front end part of the application, without needing any access. It would be better to have the unauthenticated part of the application separate, then load the full Javascript code after the user is authenticated.

Vulnerability summary

Vulnerability	Description	Host/endpoint	Rating	Impact	Remediation
Javascript files have outdated libraries, which may have vulnerabilities	Some of the libraries used in front end are old and may have vulnerabilities.	Client-side Javascript files	Informational	In some rare cases it may be possible to exploit known vulnerabilities in the old Javascript libraries.	If possible, update the Javascript libraries to latest versions.
All Javascript files are shown on the front page without requiring authentication	All Javascript code is visible to everyone, without any authentication.	Client-side Javascript files	Informational	This can help unauthenticated attacker to analyze the client-side code to understand the web application without having any access, as well as finding client-side vulnerabilities.	Separate the unauthenticated sections into separate files and show those in the unauthenticated parts and provide full Javascript to authenticated users.