

Black Hat Books

Security Audit

6005CEM

Saad Iftikhar

9789180

Contents

[Index of Figures 3](#_Toc89971204)

[Index of Tables 4](#_Toc89971205)

[Introduction 5](#_Toc89971206)

[Audit methods 5](#_Toc89971207)

[Automated security audit 5](#_Toc89971208)

[Bandit 5](#_Toc89971209)

[OWASP ZAP 5](#_Toc89971210)

[Manual security audit 6](#_Toc89971211)

[Audit Results 6](#_Toc89971212)

[Automated security audit 6](#_Toc89971213)

[OWASP ZAP 6](#_Toc89971214)

[Bandit 7](#_Toc89971215)

[Manual security audit 7](#_Toc89971216)

[A detailed description of SQL injections 9](#_Toc89971217)

[conclusion 11](#_Toc89971218)

[References 12](#_Toc89971219)

[Appendix A 13](#_Toc89971220)

[OWASP ZAP results 13](#_Toc89971221)

[Bandit output 14](#_Toc89971222)

[Appendix B 16](#_Toc89971223)

[SQL injection login 16](#_Toc89971224)

[Informative Errors Using Burp 17](#_Toc89971225)

[Multiple login 17](#_Toc89971226)

# Index of Figures

[Figure 1 OWASP ZAP Audit results 13](#_Toc89971227)

[Figure 2 OWASP ZAP risk ratting 13](#_Toc89971228)

[Figure 3 OWASP ZAP ALERT 1 13](#_Toc89971229)

[Figure 4 OWASP ZAP ALERT 2 14](#_Toc89971230)

[Figure 5 OWASP ZAP ALERT 4 14](#_Toc89971231)

[Figure 6 Bandit SQL injection 14](#_Toc89971232)

[Figure 7 usage of temp file 15](#_Toc89971233)

[Figure 8 User Settings URL 16](#_Toc89971234)

[Figure 9 XXS 16](#_Toc89971235)

[Figure 10 SQL injection 17](#_Toc89971236)

[Figure 11 Using Burp 17](#_Toc89971237)

[Figure 12 Informative error 17](#_Toc89971238)

[Figure 13 Multiple logins 18](#_Toc89971239)

# Index of Tables

[Table 1 OWASP ZAP Vulnerabilities high risk 6](#_Toc89971240)

[Table 2 OWASP ZAP Vulnerabilities low risk 6](#_Toc89971241)

[Table 3 Bandit Vulnerabilities 7](#_Toc89971242)

[Table 4 OWASP Top Ten Web Application Security Risks 8](#_Toc89971243)

[Table 5 Manual security risk assessment 8](#_Toc89971244)

[Table 6 OWASP Risk Assessment Calculator 9](file:///C:\Users\User\Desktop\6005\Course%20work\6005_CEM\Website_Audit.docx#_Toc89971245)

# Introduction

This security audit looks for and evaluates security flaws in the Black Hat Books website, which sells ethical hacking books and resources. It also offers suggestions on how to secure your system from specific threats and allows website owners to act before the vulnerability is publicly disclosed. A range of technologies were used in this audit (see Appendix C).

# Audit methods

This audit used both automatic and manual audits intending to give quality and time-efficient security audits of the Black Hat Books website.

## Automated security audit

Many organisations try to automate aspects of the pen testing process to save time, however, the pen tests are still supervised by a security analyst (Manual vs Automated Penetration Testing, 2021). It can be an expensive option to perform manual penetration testing as it requires experts’ testers for the duration of the test. Most of the test is automated hence automated testing can be a simple, secure, and less time-consuming process than manual testing [(Samant, 2011)](#Samant). This audit will use a Bandit and OWASP ZAP as the automated testing tools. These tools are used as they can cover both the code and website audit and could potentially detect issues missed with manual audit methods.

### Bandit

Bandit is a tool for finding common security flaws in Python code. It discovers the flaws in each composition of Python code and reports these to the user. As stated in [(Bandit Test Plugins — Bandit documentation, 2021)](#Bandit) it tests for the following vulnerabilities.

* B1xx – misc tests
* B2xx -application/framework misconfiguration
* B3xx – blacklists (calls)
* B4xx – blacklists (imports)
* B5xx – cryptography
* B6xx – injection
* B7xx – XSS

### OWASP ZAP

The OWASP ZAP tool is an automated website security testing tool that may be used to detect flaws in each website. As stated in [(OWASP ZAP – ZAP Alert Details, 2021)](#ZaproxyAlerts) it can be deployed to a website, test, and reported on the following website security vulnerabilities.

* SQL injection
* Broken Authentication
* Sensitive data exposure
* Broken Access control
* Security misconfiguration
* Cross-Site Scripting (XSS)
* Insecure Deserialization
* Components with known vulnerabilities
* Missing security headers

After testing for the vulnerabilities, it classifies them in different threat levels High, Medium, Low, informational, and it also accounts for False positives [(OWASP ZAP – Getting Started, 2021)](#ZaproxyGettingStarted). It allows its users to open the code is used for penetration testing and deploy it on the given website for a manual test if required.

## Manual security audit

In manual audits, skilled pen testers manually test the vulnerability and risk of a system or programme [(Manual vs Automated Penetration Testing, 2021)](#BlueFort) . Automated testing tools are not intelligent and cannot perform complex troubleshooting steps to reveal errors on this site. Use manual testing to find obvious security issues. To achieve this, look at the website source code and the website itself to see if there are any easily accessible and vulnerable features.

# Audit Results

When using the manual and automated audit methods a total of 35 vulnerabilities on the website were discovered.

## Automated security audit

### OWASP ZAP

OWASP ZAP detected 18 vulnerabilities on the website. 2 of high severity (see in Table 1 OWASP ZAP Vulnerabilities), 5 of medium severity (see in Table 1 OWASP ZAP Vulnerabilities), 6 of low severity and 5 informational as ranked by OWASP ZAP (Appendix A).

|  |  |  |
| --- | --- | --- |
| vulnerabilities | Severity |  |
| Cross-site scripting | High |  |
| SQL injection | High |  |
| Buffer overflow | Medium |  |
| CSP: Wildcard Directive | Medium |  |
| Cross-Domain Misconfiguration | Medium |  |
| Example Medium-Level Notification | Medium |  |
| X-Frame-Options Header Not Set | Medium |  |

Table 1 OWASP ZAP Vulnerabilities high risk

The low and informational raked vulnerabilities are as follows (see Table 2 OWASP ZAP Vulnerabilities low risk).

|  |  |  |
| --- | --- | --- |
| vulnerabilities | Severity |  |
| Absence of Anti-CRF Tokens | Low |  |
| Application Error Disclosure | Low |  |
| Cross-Domain JavaScript File Inclusion | Low |  |
| Example Low-Level Notification | Low |  |
| Incomplete or No Cache-Control Header Set | Low |  |
| X-Content-Type-Options Header Missing | Low |  |
| Example informational Alert Notification | Informational |  |
| HUD Tutorial Page Alert | Informational |  |
| Information Disclosure – Suspicious Comments | Informational |  |
| Timestamp Disclosure - Unix | Informational |  |

Table 2 OWASP ZAP Vulnerabilities low risk

### Bandit

Using Bandit, the results showed a total of 2 vulnerabilities in the code of medium severity as classified by bandit (see Table 3 Bandit Vulnerabilities and Appendix A).

|  |  |
| --- | --- |
| vulnerabilities | Severity |
| SQL injection | Medium |
| Usage of temp directory | Medium |

Table 3 Bandit Vulnerabilities

## Manual security audit

In the manual audit, a total of 15 vulnerabilities were found they are as follows.

* The Website doesn’t use HTTPS but rather uses HTTP having an insecure connection without reliable encryption.
* There is no Transport Layer Security (TLS). This can lead websites to have a higher susceptibility to attacks, data breaches, the identity of parties exchanging information cannot be confirmed exchanging information and data could be forged or tampered with.
* Cross-site scripting (XSS) is possible on the website which can lead to exposing a user's session cookie, allowing an attacker to hijack the user's session and seize control of the account, revealing end-user data, the installation of Trojan horse programmes, the redirection of the user to another website or site, and the modification of content display [(Cross Site Scripting (XSS) Software Attack | OWASP Foundation, 2021)](#OwasporgXSS).
* SQL injection can take place on the website so it could be possible to access sensitive database data, modify database data, perform database administration activities, retrieve the content of a specific file on the DBMS file system, and in certain situations issue commands to the operating system [(SQL Injection | OWASP, 2021)](#OwasporgSQLInjection).
* There is no login time out – so a password can be entered as many times as possible which could lead to brute force attacks.
* sha512 is mathematically secure but brute force can be applied to it repeatedly.
* No Salt on the hashing which makes using brute force attacks easier.
* Credit card numbers are not hashed or salted.
* Password can be of any length, just numbers, special characters or alphabets making.
* Single-factor authentication is used.
* By using the extension user/<Number>/settings the attacker can view the settings of not only a user but also the admin without logging in.
* Errors on the website are overly informative (see Appendix B).
* The website does not keep track of a user’s malicious activities on the site penetration testing and scans using OWASP ZAP are quite extensive, performed using the same machine and do not trigger a stop response from the website not limiting an attacker attempts at hacking the Website.
* Users can log in using multiple accounts/logins at the same time so the website is open and free to someone who is not the proper user and can abuse the login.
* The website currently has no method of creating a backup of itself, making it difficult to restore the site if any issues occur.

|  |  |
| --- | --- |
| Ranking 2021 | Security Risk |
| A01 | Broken Access Control |
| A02 | Cryptographic Failures |
| A03 | Injection |
| A04 | Insecure design |
| A05 | Security Misconfiguration |
| A06 | Venerable and outdated components |
| A07 | Identification and Authentication Failures |
| A08 | Software and data integrity failures |
| A09 | Security logging and monitoring failures |
| A010 | Server-side requests Forgery |

Table 4 OWASP Top Ten Web Application Security Risks

The manual audit issues have been classified in different threat levels (see Table 5 Manual security risk assessment) using [(OWASP Top Ten Web Application Security Risks | OWASP, 2021)](#OwasporgTopTenSecurityRisks) (see Table 4 OWASP Top Ten Web Application Security Risks).

|  |  |  |
| --- | --- | --- |
| vulnerabilities | Security risks present (OWASP Top Ten Web Application Security Risks | OWASP, 2021) | Total security risks |
| No use of HTTP | A02, A04, A05, A06 | 4 |
| Not using TLS | A02, A04, A05, A06 | 4 |
| Cross-site scripting | A01, A03, A04, A05, A08, A09 | 6 |
| SQL injection | A01, A03, A04, A05, A08, A09 | 6 |
| No login time out | A03, A05, A07, A09 | 4 |
| Using sha512 | A03, A05, A06, A07 | 4 |
| No Salting | A02, A04, A05, A06, A07 | 5 |
| Credit card not hashed or salted | A02, A04, A05, A06, A07 | 5 |
| Password length setting | A03, A05, A07 | 3 |
| Single-factor authentication | A03, A05, A06, A07 | 4 |
| Viewing settings without logging in  user/<Number>/settings | A01, A04, A05, A08, A09 | 5 |
| Overly informative error messages | A05, A09 | 2 |
| No stop response triggered after multiple attacks | A01, A03, A04, A05, A07, A09 | 6 |
| Multiple logins | A01, A04, A05, A07 | 4 |
| No backup | A04 | 1 |

Table 5 Manual security risk assessment

# A detailed description of SQL injections

SQL injection attacks are a severe security risk for Web applications because they allow attackers to get uncontrolled access to the databases that underpin the web applications, as well as the potentially sensitive information contained inside those databases [(Halfond et al., 2006, pp. 13-15)](#Halfond). Sensitive data, including usernames, passwords, names, addresses, phone numbers, and credit card numbers can be exposed using SQL injection [(Clarke, 2009)](#Clarke). SQL Injection attacks are limited in their severity by the attacker's skill and inventiveness, and to a lesser degree, defence in depth remedies such low privilege connections to the database server. Consider SQL Injection to have a high effect severity [(SQL Injection | OWASP, 2021)](#OwasporgSQLInjection).

SQL injections can be used in various ways some of the most common ways are as follows.

* Attackers exploit specially engineered user input to inject SQL statements [(Halfond et al., 2006, pp. 13-15)](#Halfond).
* An attacker might simply submit an attack by embedding it in the cookie if a Web application utilises the cookie's contents to create SQL queries [(Halfond et al., 2006, pp. 13-15)](#Halfond).
* Attackers can inject through server variables by altering the values in HTTP and network headers, and they can take advantage of this flaw by inserting an SQLIA right into the headers. The attack in the forged header is triggered when the database query to log the server variable is sent [(Halfond et al., 2006, pp. 13-15)](#Halfond).
* In second-order injections, attackers’ plant malicious inputs into a system or database to cause an SQLIA when that input is utilised later [(Halfond et al., 2006, pp. 13-15)](#Halfond).

By using the OWASP Risk Assessment Calculator [(Olmedo, 2021)](#RiskCalculator) it was determined that SQL injection had high severity with a high likelihood of occurring and will have a significant impact on the given target (see Table 6 OWASP Risk Assessment Calculator).

Graphical user interface, application

Description automatically generated

Table 6 OWASP Risk Assessment Calculator

A technique to combating SQL injection attacks is to treat them like an input validation problem, accepting only characters from an allowed list of safe values or identifying and escaping potentially dangerous data from a deny list. An allow list may be a powerful tool for implementing tight input validation requirements, but parameterized SQL statements are easier to maintain and provide additional security assurances. Using stored procedures is another popular approach for dealing with SQL injection threats [(SQL Injection | OWASP, 2021)](#OwasporgSQLInjection).

# conclusion

After performing an audit, it is evident that Black Hat Books is an insecure website, but it can simply be modified and most vulnerabilities can be patched. Some vulnerabilities put the website at a higher risk of exploitation and should be dealt with straight away for example attackers or users should not be able to open their or other users settings by simply using the user/<Number>/settings extension, user input could be better filtered to reduce the chance of XSS and SQL injection, user passwords can be salted, better hashing algorithms could be used for the user passwords and credit cards information, a timeout could be implemented, a length and rules could be set for passwords, two-factor authentication could be implemented to reduce the risk of brute force attacks, HTTPS with TLS should be used on the website, errors should not be very informative, users should not be allowed to login in with more than one account per session and a backup should be kept as a need could arise. The vulnerabilities with a higher score should be addressed right once since they have the most potential to harm the website and its visitors. However, all the vulnerabilities identified in this audit can be secured, and it should be a top priority to do so.

# References

Bandit.readthedocs.io. 2021. *Bandit Test Plugins — Bandit documentation*. [online] Available at: <https://bandit.readthedocs.io/en/latest/plugins/index.html> [Accessed 8 December 2021].

BlueFort Security. 2021. *Manual vs Automated Penetration Testing*. [online] Available at: <https://www.bluefort.com/news/latest-blogs/manual-vs-automated-penetration-testing/> [Accessed 7 December 2021].

Clarke, J. (2009). *SQL injection attacks and defense*. Elsevier.

Halfond, W. G., Viegas, J., & Orso, A. (2006, March). A classification of SQL-injection attacks and countermeasures. In *Proceedings of the IEEE international symposium on secure software engineering* (Vol. 1, pp. 13-15). IEEE.

Owasp.org. 2021. *Cross Site Scripting (XSS) Software Attack | OWASP Foundation*. [online] Available at: <https://owasp.org/www-community/attacks/xss/> [Accessed 8 December 2021].

Owasp.org. 2021. *SQL Injection | OWASP*. [online] Available at: <https://owasp.org/www-community/attacks/SQL\_Injection> [Accessed 8 December 2021].

Owasp.org. 2021. *OWASP Top Ten Web Application Security Risks | OWASP*. [online] Available at: <https://owasp.org/www-project-top-ten/> [Accessed 8 December 2021].

Olmedo, J., 2021. *OWASP Risk Assessment Calculator v2021*. [online] Javierolmedo.github.io. Available at: <https://javierolmedo.github.io/OWASP-Calculator/> [Accessed 8 December 2021].

Samant, N. (2011). Automated penetration testing.

Zaproxy.org. 2021. *OWASP ZAP – ZAP Alert Details*. [online] Available at: <https://www.zaproxy.org/docs/alerts/> [Accessed 8 December 2021].

Zaproxy.org. 2021. *OWASP ZAP – Getting Started*. [online] Available at: <https://www.zaproxy.org/getting-started/> [Accessed 8 December 2021].

# Appendix A

## OWASP ZAP results

The OWASP ZAP audit results are classified by high, medium, low, informational, and False positive (see Figure 2 OWASP ZAP risk ratting ) after running the OWASP ZAP tool 18 vulnerabilities were found (see Figure 1 OWASP ZAP Audit results). The tool makes it easy to understand these vulnerabilities and can allow the user to implement them on the website (see Figure 3 OWASP ZAP ALERT 1, Figure 4 OWASP ZAP ALERT 2 and Figure 5 OWASP ZAP ALERT 4).

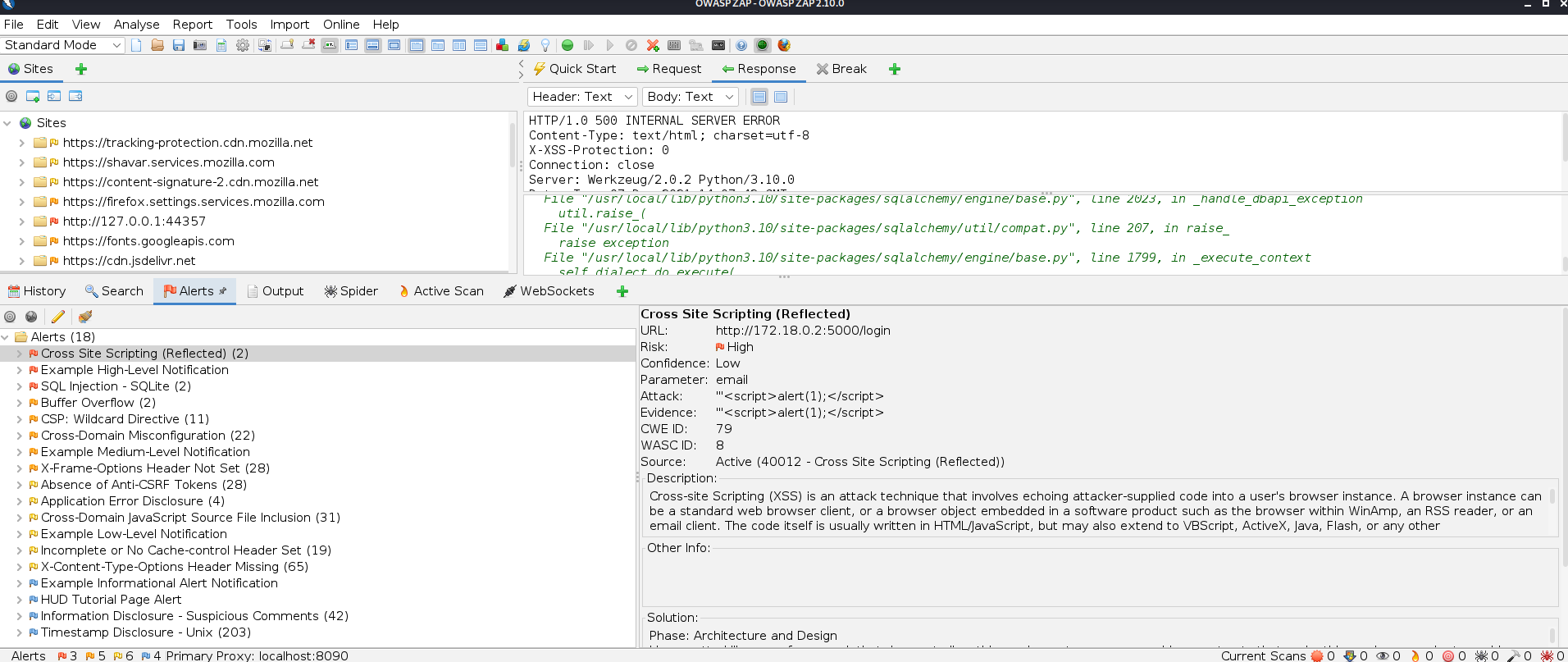


Figure 1 OWASP ZAP Audit results

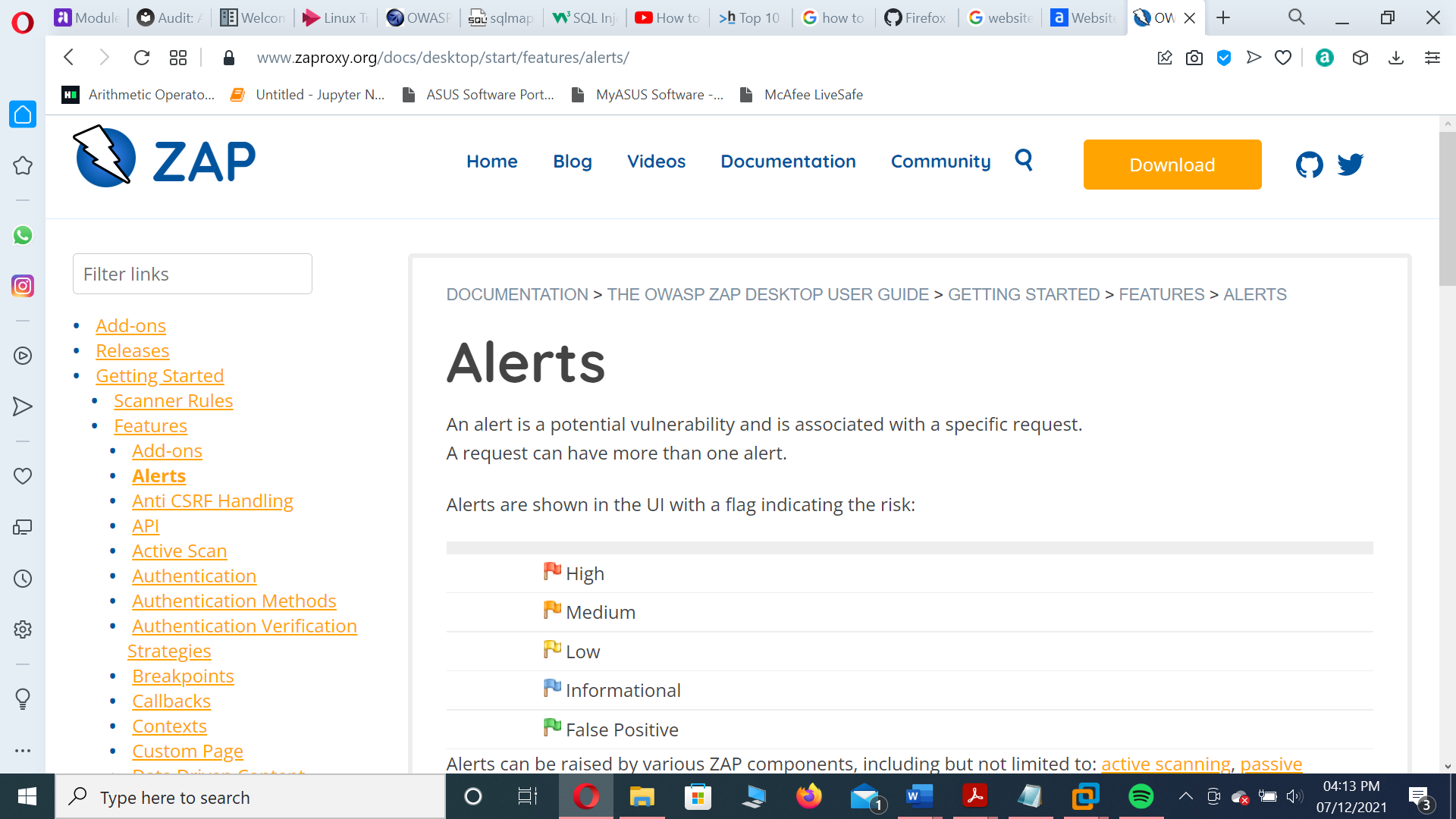


Figure 2 OWASP ZAP risk ratting

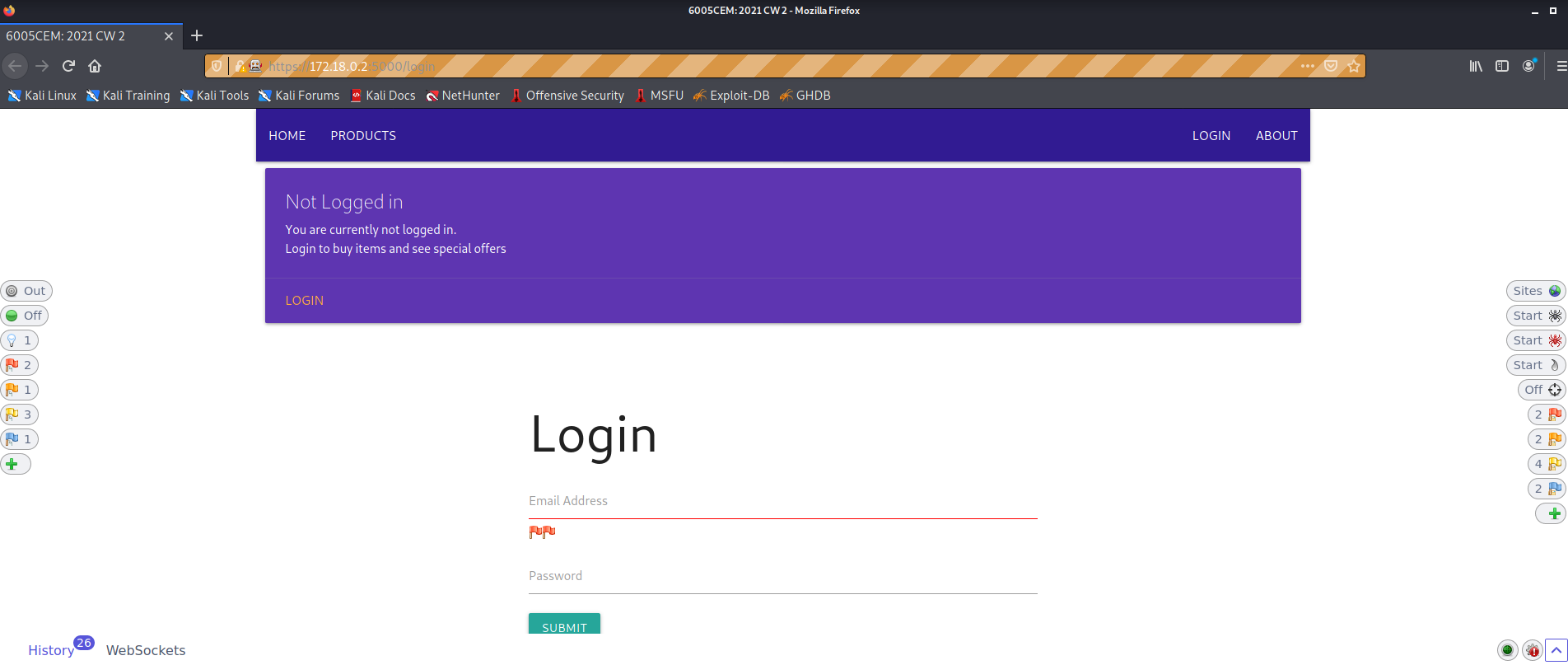


Figure 3 OWASP ZAP ALERT 1

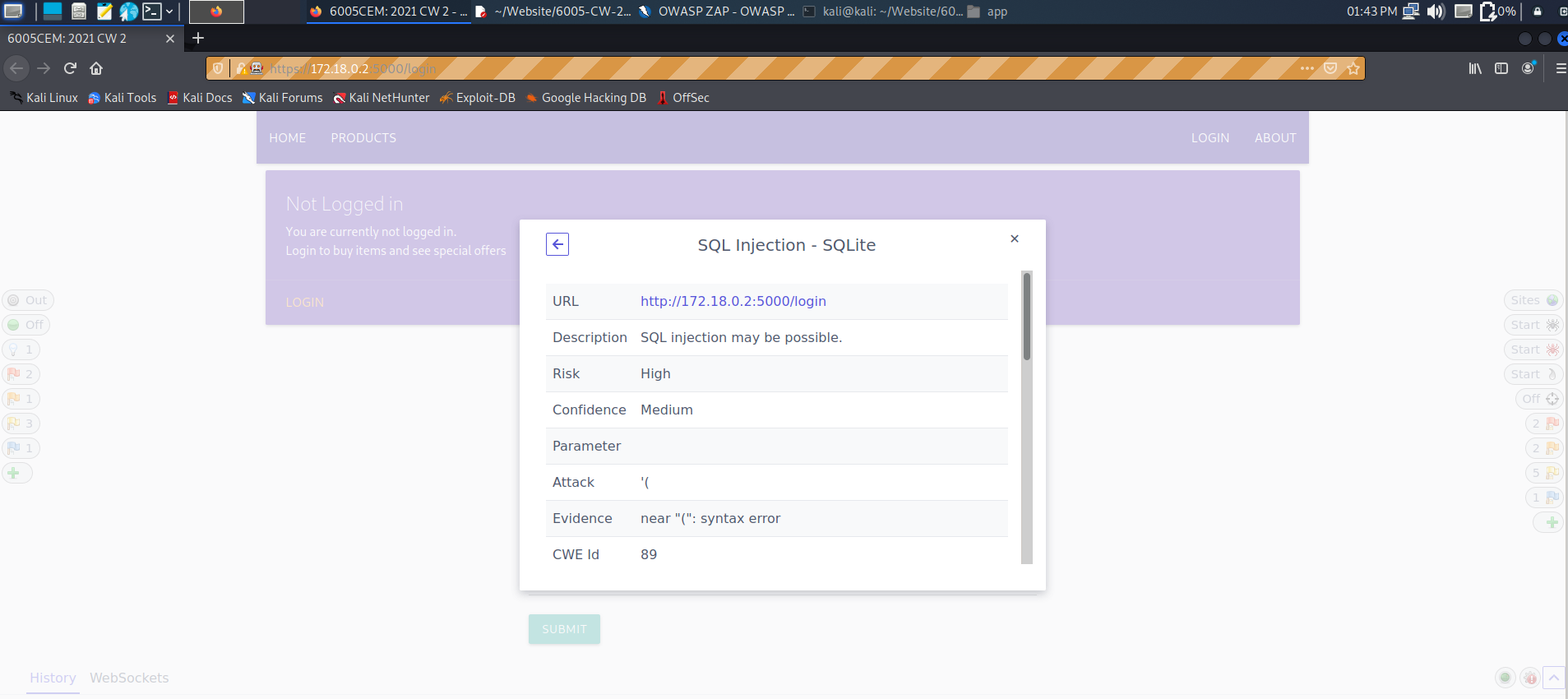


Figure 4 OWASP ZAP ALERT 2



Figure 5 OWASP ZAP ALERT 4

## Bandit output

The code bandit -r app was used to run bandit to perform the automated audit on the website's code. The results from the audit gave a SQL injection (see Figure 6 Bandit SQL injection) and usage of temp file (see Figure 7 usage of temp file) vulnerabilities.

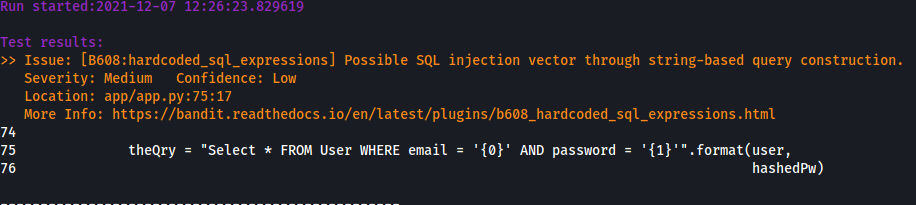


Figure 6 Bandit SQL injection

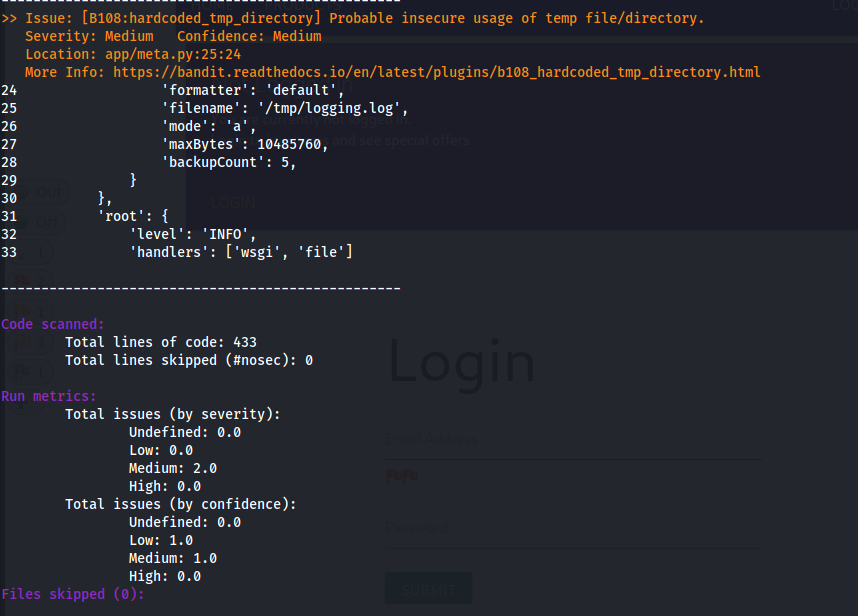


Figure 7 usage of temp file

# Appendix B

Logic Login Error

User or attackers may look at other users’ settings using the 172.18.0.2:5000/user/1/settings URL without needing to sign in (see Figure 8 User Settings URL).

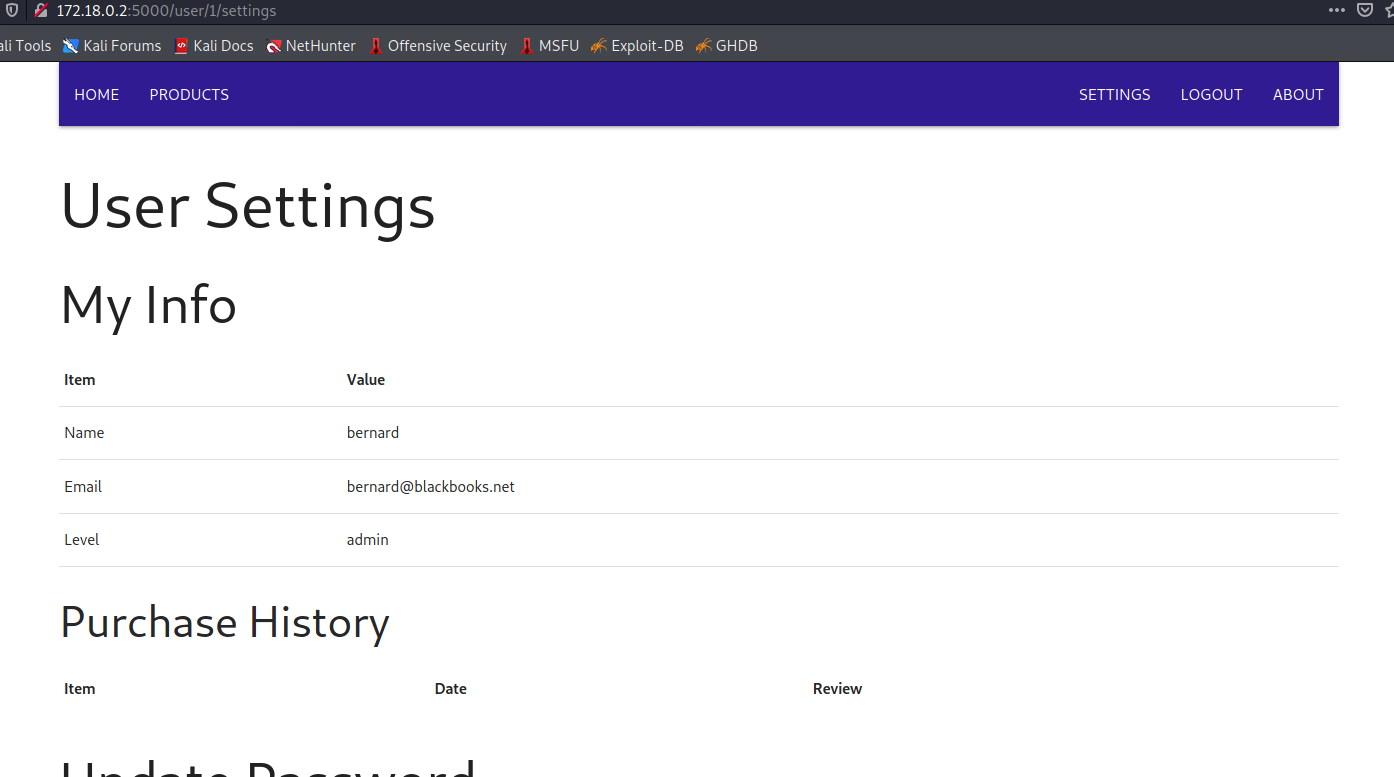


Figure 8 User Settings URL

### SQL injection login

Changing the blue highlighted type field to “text” (see Figure 9 XXS) a SQL injection can be made (see Figure 10 SQL injection).

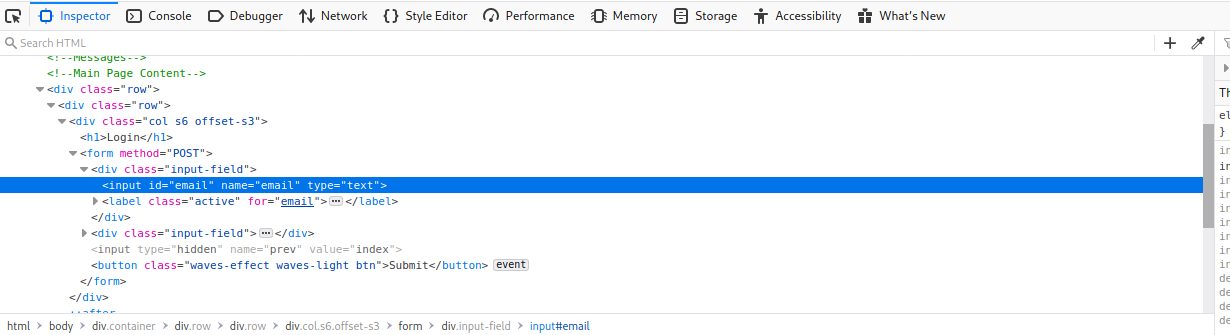


Figure 9 XXS

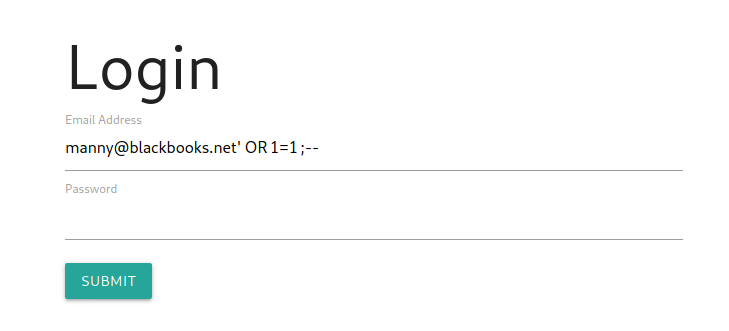


Figure 10 SQL injection

### Informative Errors Using Burp

Changing the yellow password in line 15 from = to != (see Figure 11 Using Burp) opens a very descriptive error page giving the type of hashing algorithm used (see Figure 12 Informative error).

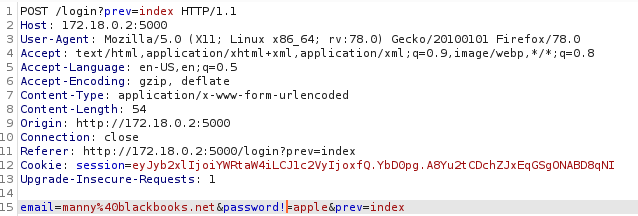


Figure 11 Using Burp

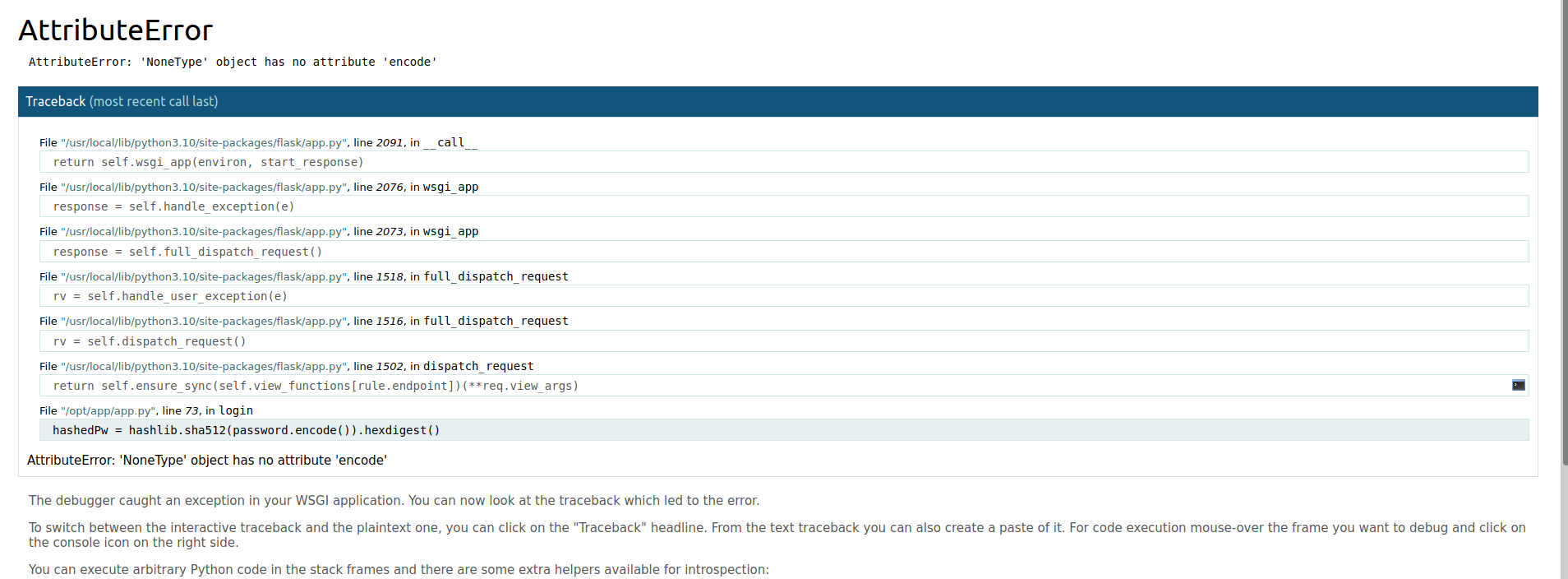


Figure 12 Informative error

### Multiple login

Users at login from more them one account using the same machine and even using the same browser (see Figure 13 Multiple login).

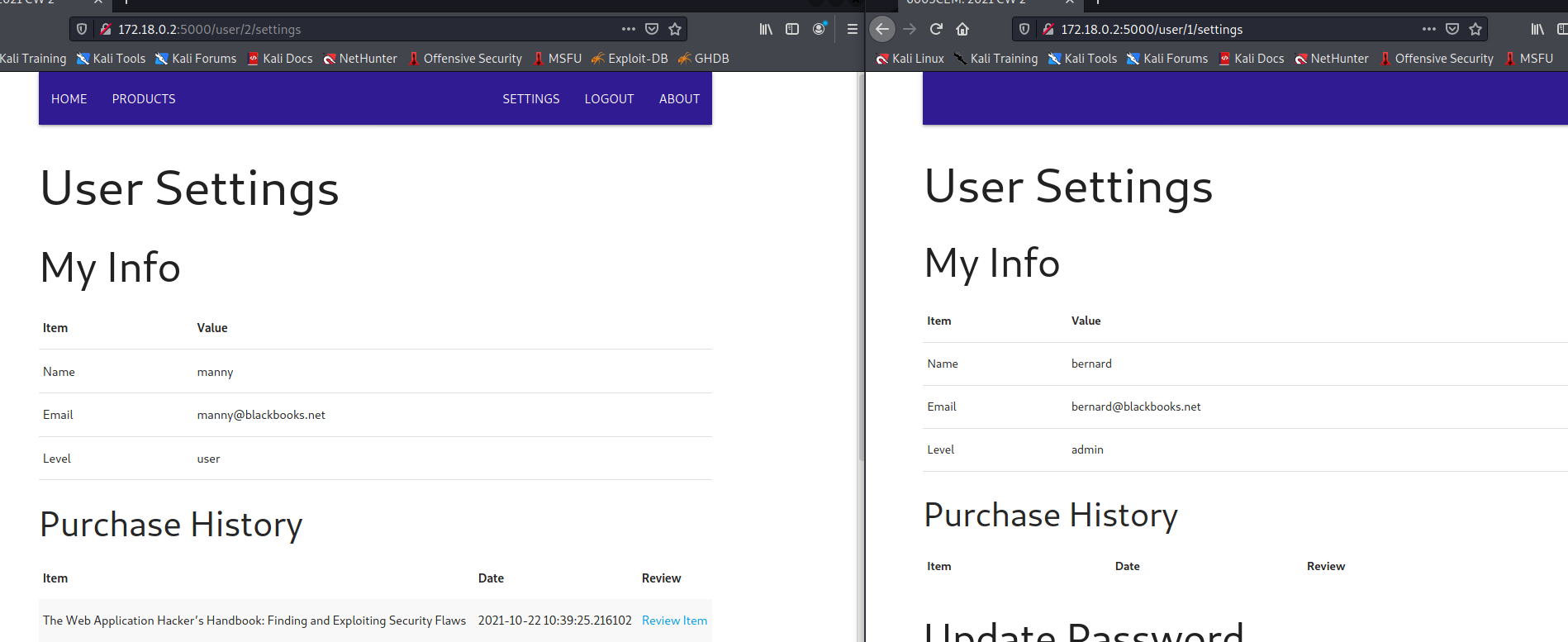


Figure 13 Multiple logins

# Appendix C

## Technologies used

A list of the technologies used in this audit.

* Kali Linux
* Python3
* Bandit
* OWASP ZAP
* Burp suit
* SQL
* JavaScript