

Отчёт по анализу защищённости Juice Shop

Введение

OWASP Juice Shop — это веб-приложение с открытым исходным кодом для демонстрации самых распространенных уязвимостей. Поддерживается некоммерческой организацией Open Web Application Security Project (OWASP) и волонтерами.

Результаты статического анализа

Проведём статический анализ кода приложения – SAST. Будем использовать бесплатный облачный инструмент Semsgrep. Для этого предварительно поместим исходный код приложения на платформе GitHub. После этого запустим сканирование. Результаты сканирования:

The screenshot shows the Semsgrep web interface for a static analysis of the Juice Shop application. The interface is divided into a sidebar on the left and a main content area on the right.

Sidebar (Left):

- Navigation menu: Dashboard, Projects, Code (82), Secrets, Supply Chain, Rules.
- Project selection: "elseire..." (dropdown).
- Status: Open (82), Ignored, Fixed.
- Category: All categories.
- Severity: High, Medium, Low.
- Component: All components.
- Confidence: High, Medium, Low.
- Recommendation: Fix, Ignore.
- Action: Monitor, Comment, Block.
- Rule: All rules.
- Ruleset: All rulesets.
- Branch: All branches, refs and PRs.
- Reset filters.
- Get Started: 41%.
- Settings, Docs, Help, Updates.

Main Content Area (Right):

82 Matching Findings

Group by Rule: All time

Findings:

- path-join-resolve-traversal** (Security: Low, </> Javascript)
 - Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be
 - data/datacreator.ts:41
 - lib/codingChallenges.ts:24
 - lib/codingChallenges.ts:24
 - lib/startup/restoreOverwrittenFilesWithOriginals.ts:30
 - lib/startup/validatePreconditions.ts:116
- express-mongo-nosqli** (Security: Medium, </> Javascript)
 - Detected a `$IMPORT` statement that comes from a `$REQ` argument. This could lead to NoSQL injection if the variable is user-controlled and is not properly sanitized. Be sure to properly sanitize the data if you absolutely
 - routes/dataExport.ts:61
 - routes/likeProductReviews.ts:18
 - routes/likeProductReviews.ts:25
 - routes/likeProductReviews.ts:31
 - routes/likeProductReviews.ts:42
- express-sequelize-injection** (Security: High, </> Javascript)
 - Detected a `sequelize` statement that is tainted by user-input. This could lead to SQL injection if the variable is
 - data/static/codefixes/dbSchemaChallenge_3.ts:5
 - data/static/codefixes/dbSchemaChallenge_3.ts:11
 - data/static/codefixes/unionSqlInjectionChallenge_3.ts:6
 - data/static/codefixes/unionSqlInjectionChallenge_3.ts:10
 - routes/login.ts:36
- express-fs-filename** (Security: Medium, </> Javascript)
 - The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the file path which the application uses to access files. If the application does not
 - routes/profileImageUpload.ts:28
 - routes/profileImageUrlUpload.ts:31
 - routes/vulnCodeFixes.ts:79
 - routes/vulnCodeFixes.ts:80
 - routes/vulnCodeSnippet.ts:93
- express-path-join-resolve-traversal** (Security: Medium, </> Javascript)
 - Possible writing outside of the destination, make sure that the target path is nested in the intended destination
 - routes/dataErasure.ts:69
 - routes/keyServer.ts:14
 - routes/rogfileServer.ts:14

Обнаруженные уязвимости из OWASP Top-10

В результатах статистического анализа кода приложения Juice Shop мы обнаружили следующие уязвимости:

1. CWE-22: Path Traversal (OWASP Broken Access Control)

13

path-join-resolve-traversal

Security Low </> Javascript

Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be

[Show more](#)

	3h	data/datacreator.ts:41	juice-shop	main	Details
	3h	lib/codingChallenges.ts:24	juice-shop	main	Details
	3h	lib/codingChallenges.ts:24	juice-shop	main	Details
	3h	lib/startup/restoreOverwrittenFilesWithOriginals.ts:30	juice-shop	main	Details
	3h	lib/startup/validatePreconditions.ts:116	juice-shop	main	Details

[Show 8 more findings](#)

2. CWE-918: Server-Side Request Forgery (OWASP Broken Access Control)

1

ssrf-deepsemgrep

Security High </> Javascript

Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in some cases, allow the attacker to control applications or systems that trust the vulnerable service. To prevent this vulnerability, avoid allowing user input to craft the base request. Instead, treat it as part of the path or query parameter and encode it appropriately. When user input is necessary to prepare the HTTP request, perform strict input validation. Additionally, whenever possible, use allowlists to only interact with expected, trusted domains.

[Hide](#)

	3h	routes/profileImageUrlUpload.ts:23	juice-shop	main	Details
--	----	------------------------------------	------------	------	-------------------------

3. CWE-611: XML External Entities (OWASP Security Misconfiguration)

1

express-libxml-vulnerable

Security Low </> Javascript

Detected use of `parseXml()` function with the `noent` field set to `true`. This can lead to an XML External Entities (XXE) attack if untrusted data is passed into it.

	4h	routes/fileUpload.ts:80	juice-shop	main	Details
--	----	-------------------------	------------	------	-------------------------

4. CWE-79: Cross-site Scripting (OWASP Injection)

1

open-redirect-deepsemgrep

Security High </> Javascript

The application builds a URL using user-controlled input which can lead to an open redirect vulnerability. An attacker can manipulate the URL and redirect users to an arbitrary domain. Open redirect vulnerabilities can lead to issues such as Cross-site scripting (XSS) or redirecting to a malicious domain for activities such as phishing to capture users' credentials. To prevent this vulnerability perform strict input validation of the domain against an allowlist of approved domains. Notify a user in your application that they are leaving the website. Display a domain where they are redirected to the user. A user can then either accept or deny the redirect to an untrusted site.

[Hide](#)

	3h	routes/redirect.ts:19	juice-shop	main	Details
--	----	-----------------------	------------	------	-------------------------

5. CWE-89: SQL Injection (OWASP Injection)

6

express-sequelize-injection

Security High </> Javascript

Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use

[Show more](#)

	3h	data/static/codefixes/dbSchemaChallenge_1.ts:5	juice-shop	main	Details
	3h	data/static/codefixes/dbSchemaChallenge_3.ts:11	juice-shop	main	Details
	3h	data/static/codefixes/unionSqlInjectionChallenge_1.ts:6	juice-shop	main	Details
	3h	data/static/codefixes/unionSqlInjectionChallenge_3.ts:10	juice-shop	main	Details
	3h	routes/login.ts:36	juice-shop	main	Details

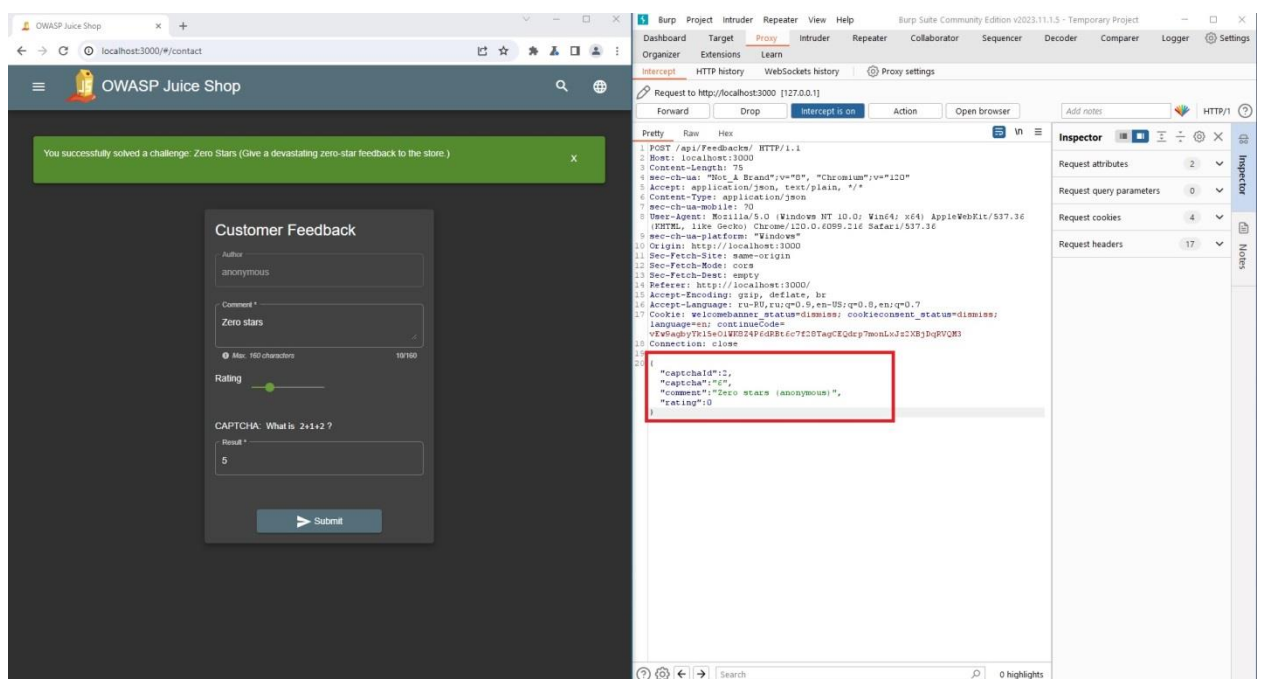
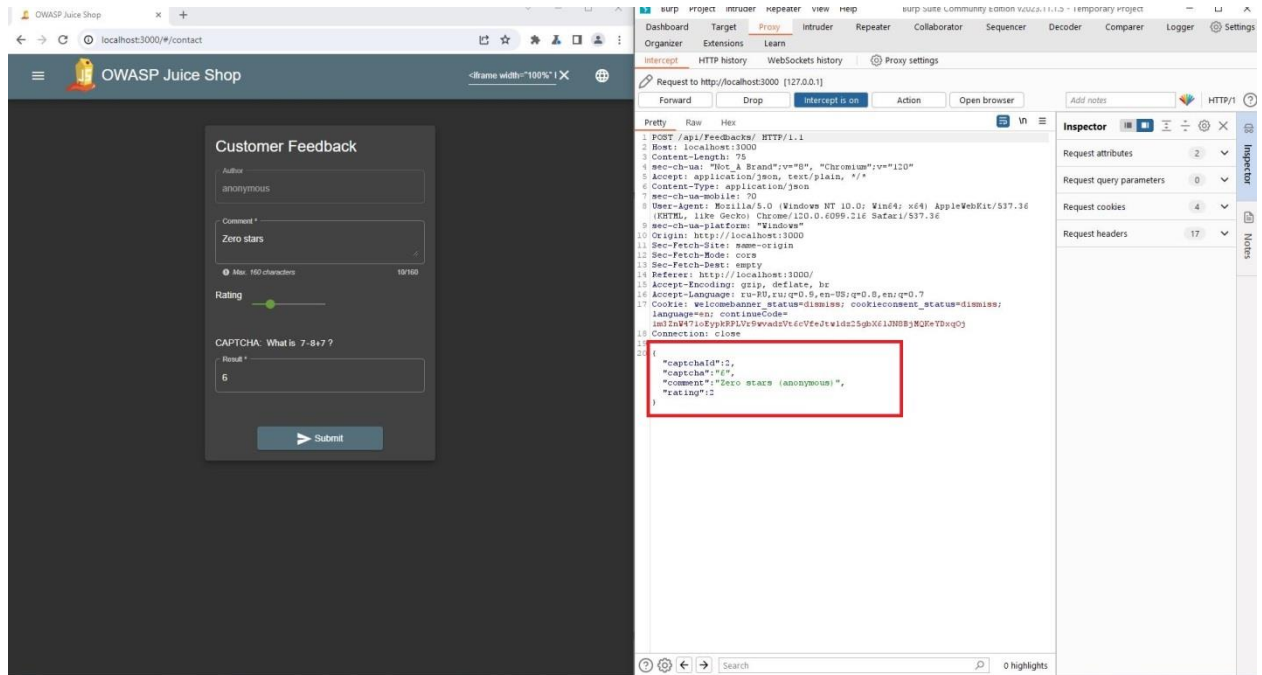
[Show 1 more findings](#)

Эксплуатация уязвимостей из OWASP Top-10

Покажем, как можно эксплуатировать найденные уязвимости в приложении Juice Shop:

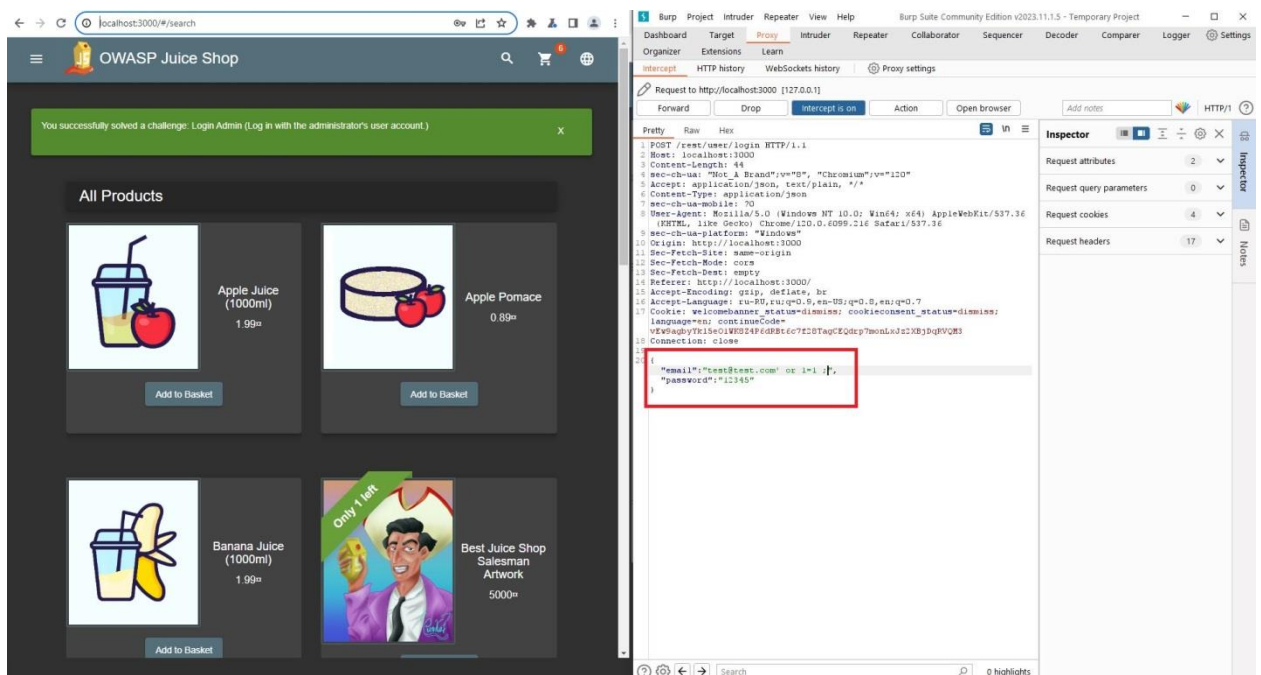
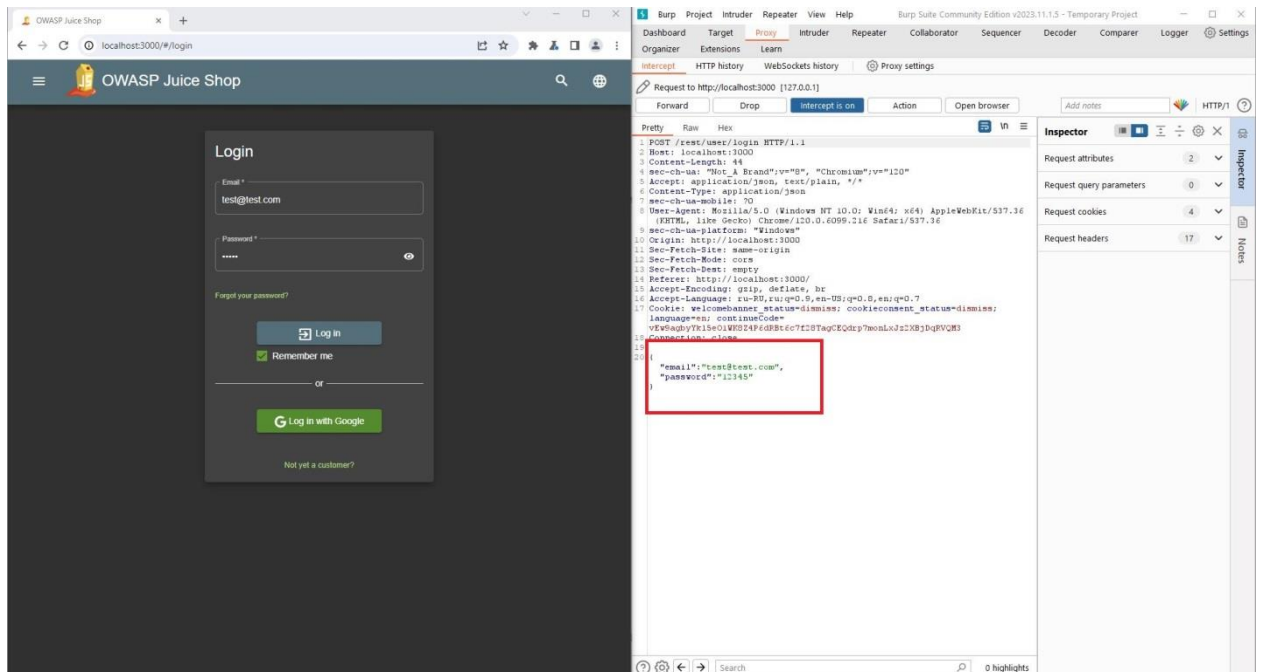
1. CWE-20: Improper Input Validation (OWASP Injection)

Заходим на страницу для отправки отзывов (<http://localhost:3000/#/contact>), где пишем текст и решаем задачу для капчи. Количество звёзд ставим любое. После чего включаем перехват запросов в Burp Suite. В перехваченном запросе нам надо поменять значение "rating" на цифру 0. После чего отправить изменённый запрос.



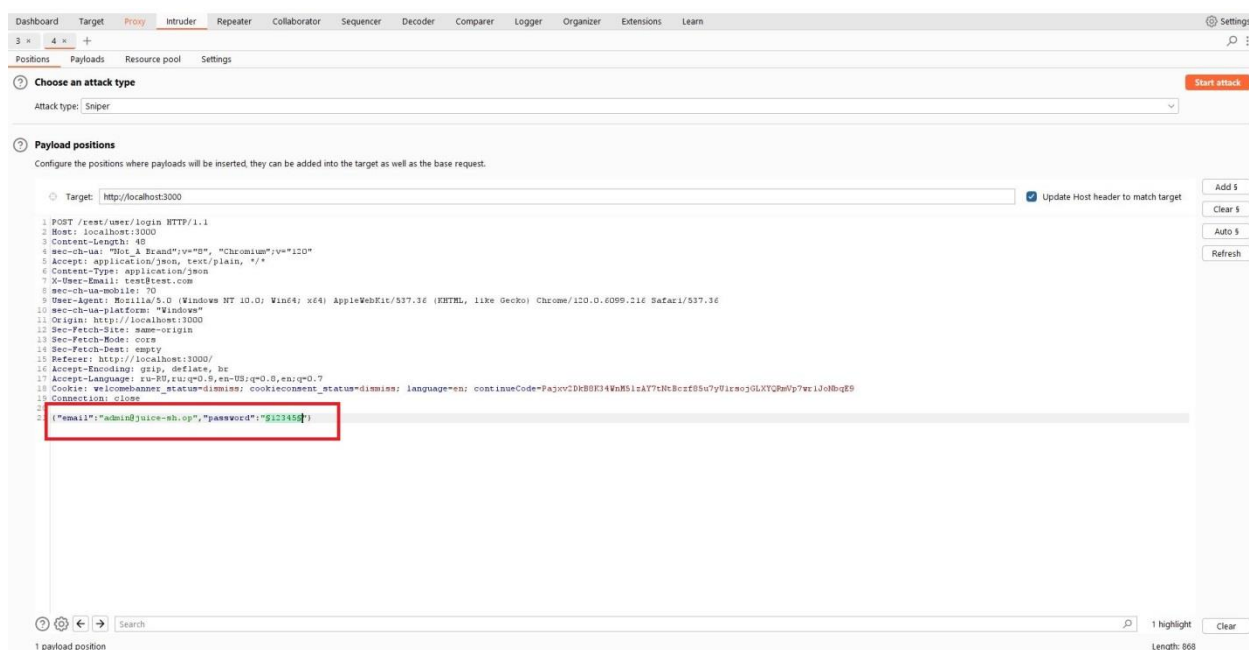
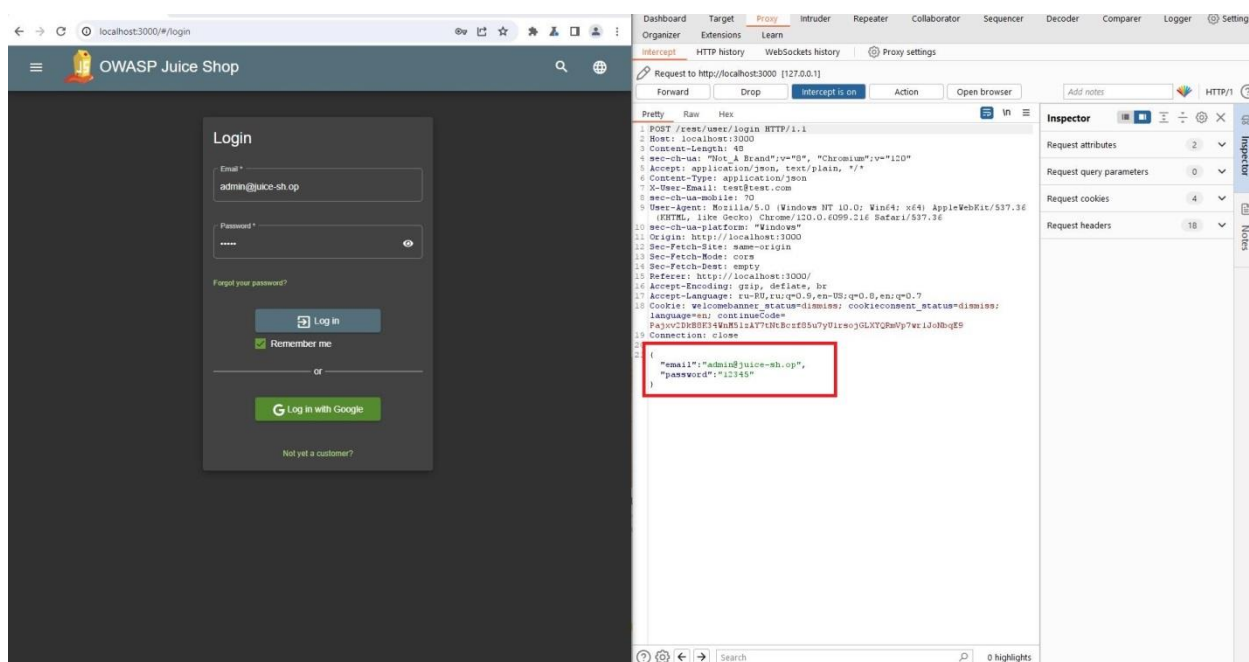
2. CWE-89: SQL Injection (OWASP Injection)

Заходим на страницу авторизации (<http://localhost:3000/#/login>), где указываем любой адрес почты и пароль. После чего включаем перехват запросов в Burp Suite. В перехваченном запросе нам надо дописать в значение "email" следующие символы "email' or 1=1 ;". После чего отправить изменённый запрос.



3. CWE-307: Improper Restriction of Excessive Authentication Attempts (OWASP Identification and Authentication Failures)

Теперь попробуем подобрать пароль администратора, зная его электронную почту admin@juice-sh.op. Также заходим на страницу авторизации (<http://localhost:3000/#/login>), где указываем почту администратора и любой пароль. После чего включаем перехват запросов в Burp Suite. Передаём перехваченный запрос в Intruder. Так указываем, что значение "password" будет переменной, которую мы будем перебирать. Настраиваем полезную нагрузку, указывая возможные варианты паролей для перебора. После этого запускаем атаку перебора. Смотрим успешный ответ с заголовком 200. Видно, что успешный ответ на пароль admin123.



The screenshot displays the Burp Suite interface with the 'Payloads' tab selected. The 'Payload sets' section shows a configuration for a 'Simple list' payload type with 9 payloads. The 'Payload settings [Simple list]' section shows a list of payloads: 123456, admin, aaadmin, password, admin1, admin12, and admin123. The 'Payload processing' section shows a list of processing rules. The 'Payload encoding' section shows a checkbox for 'URL-encode these characters' which is checked.

A 'Results' window is open, showing a table of attack results. The table has columns: Request, Payload, Status code, Error, Timeout, Length, and Comment. The results show that the payload 'admin123' (Request 7) resulted in a 200 status code and a length of 1197, which is highlighted with a red box.

Request	Payload	Status code	Error	Timeout	Length	Comment
0		401			413	
1	123456	401			413	
2	admin	401			413	
3	aaadmin	401			413	
4	password	401			413	
5	admin1	401			413	
6	admin12	401			413	
7	admin123	200			1197	
8	admin1234	401			413	
9	admin12345	401			413	

Рекомендации по устранению уязвимостей из OWASP Top-10

Рекомендации по устранению к трём продемонстрированным уязвимостям:

1. CWE-20: Improper Input Validation (OWASP Injection)

Для защиты от этой атаки необходима дополнительная проверка отправляемых данных на стороне сервера. Даже если злоумышленник подменит значения, генерируемые формой, то логика на стороне сервера не даст вставить «некорректные» данные.

2. CWE-89: SQL Injection (OWASP Injection)

Для защиты от этой атаки необходимо дополнительно экранировать передаваемые данные, чтобы помещать злоумышленнику внедрять управляющие символы SQL в обычные данные. Это помещает выполнению вредоносной команды, и она будет обработана как текстовые данные.

3. CWE-307: Improper Restriction of Excessive Authentication Attempts (OWASP Identification and Authentication Failures)

Для защиты от этой атаки необходимо предусмотреть временной лимит между отправками запросом на авторизацию и блокировки учётной записи на некоторое время при нескольких неудачных попытках подряд. Ещё можно обезопасить форму авторизации, добавив капчу, что усложнит перебор.