

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

По дисциплине «Информационная безопасность»
Лабораторная работа №4
Анализ уязвимостей веб-приложения с помощью OWASP ZAP

Студент:

Дениченко Александр Олегович Р3412

Практик:

Маркина Татьяна Анатольевна

Санкт-Петербург
2025 г.

Цель

Освоить базовые навыки динамического тестирования безопасности (DAST) на примере тестового приложения.

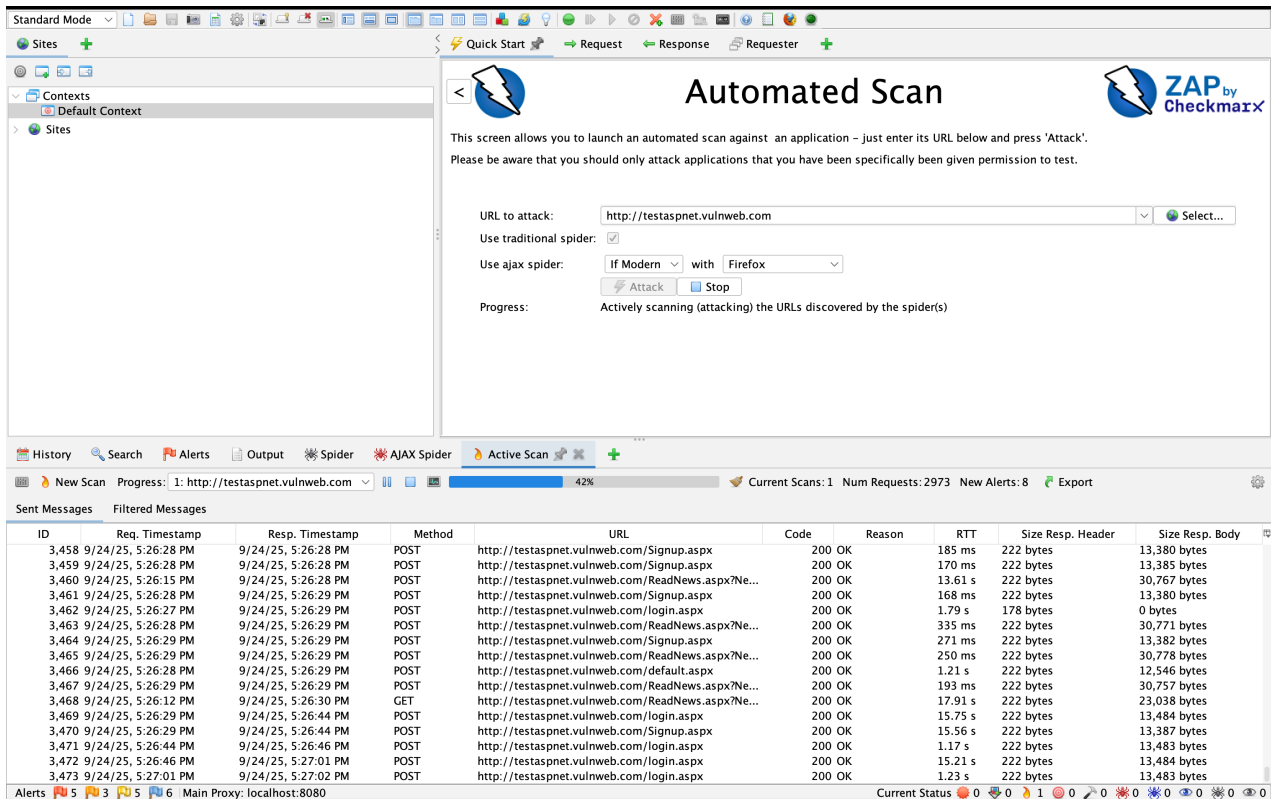
1 Вводная часть

Изначально был установлен OWASP ZAP и запущен. Выбран режим Quick Scan.

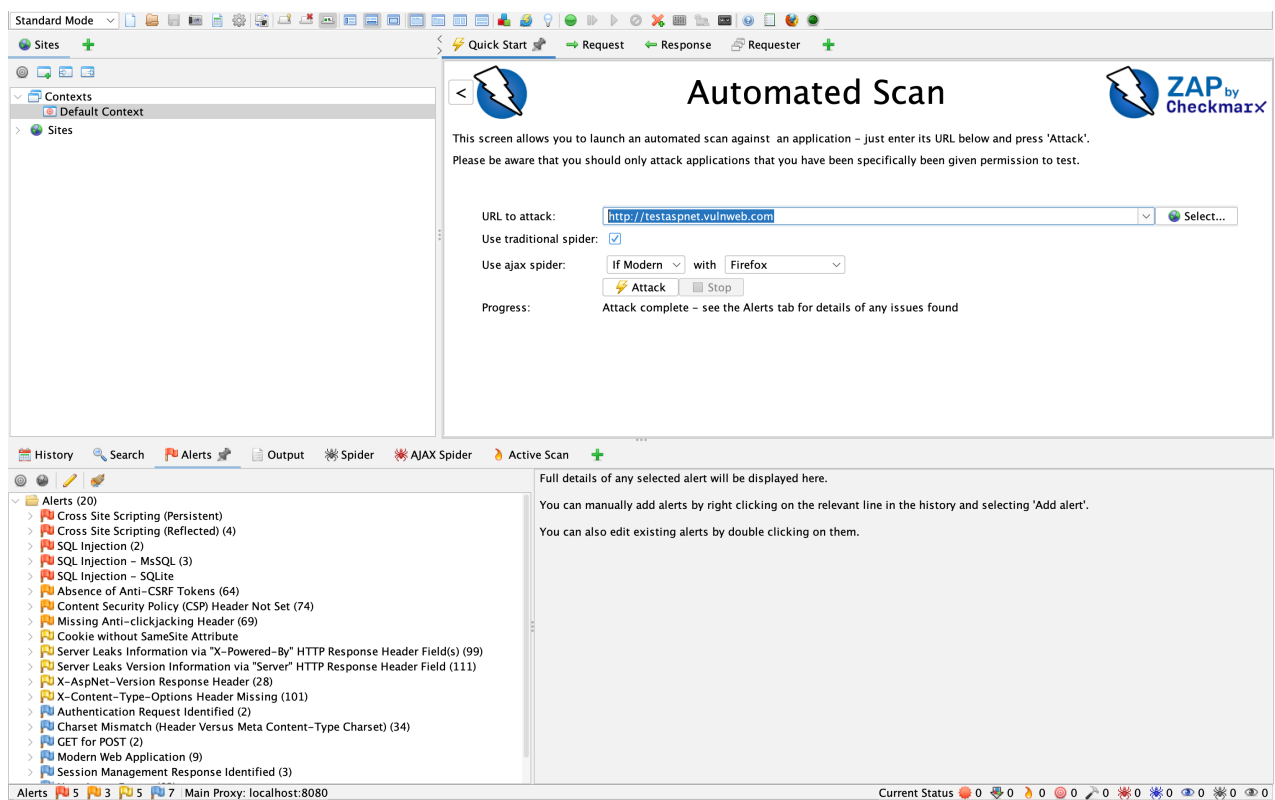
Для тестирования был выбран сайт `http://testaspnet.vulnweb.com`. Данный сайт не является коммерческим, а является тестовым приложением для тестирования безопасности. Не будет этично тестировать реальные сайты на уязвимости и баловаться с нагрузкой.

2 Ход работы

Изначально я перешёл на тестовый сайт во встроенном браузере ZAP. Выбрал режим Automatic Scan + поставил галочку, чтобы использовать традиционный Spider. Далее запустил атаку.




В течении сканирования выявлено 20 Alerts.



По окончании сканирования я сгенерировал отчёт и рассмотрел раздел Alerts.

3 XSS

Cross Site Scripting (Persistent)
URL: http://testaspnet.vulnweb.com/Comments.aspx?id=2
Risk:  High
Confidence: Medium
Parameter: tbComment
Attack: </div><script>alert(1);</script><div>
Evidence:
CWE ID: 79
WASC ID: 8
Source: Active (40014 – Cross Site Scripting (Persistent))
Input Vector: Form Query

Description:
Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to

Other Info:
Source URL: http://testaspnet.vulnweb.com/Comments.aspx?id=2

Solution:
Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Reference:
<https://owasp.org/www-community/attacks/xss/>
<https://cwe.mitre.org/data/definitions/79.html>

Alert Tags:

Key	Value
CWE-79	https://cwe.mitre.org/data/definitions/79.html
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/
POLICY_DEV_FULL	
POLICY_QA_STD	

Сканирование выявило критическую уязвимость: Persistent Cross Site Scripting (XSS) на странице комментариев по адресу <http://testaspnet.vulnweb.com/Comments.aspx?id=2>. Это позволяет злоумышленнику сохранять вредоносный скрипт, который будет выполняться для всех пользователей, просматривающих страницу.


Атака: внедрение кода через комментарий:

```
</div><script>alert(1);</script><div>
```

Уязвимость подтверждена как персистентная: скрипт сохраняется на сервере и срабатывает для любого пользователя. Уязвимости CWE-79 (OWASP TOP 10 — Injection).

Cross Site Scripting (Reflected)

URL: <http://testaspnet.vulnweb.com/ReadNews.aspx?NewsAd=javascript%3Aalert%281%29%3B&id=2>

Risk:  High

Confidence: Medium

Parameter: NewsAd

Attack: javascript:alert(1);

Evidence: javascript:alert(1);

CWE ID: 79

WASC ID: 8

Source: Active (40012 – Cross Site Scripting (Reflected))

Input Vector: URL Query String

Description:

Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to

Other Info:

Solution:

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Reference:

<https://owasp.org/www-community/attacks/xss/>
<https://cwe.mitre.org/data/definitions/79.html>


Alert Tags:

Key	Value
POLICY_QA_FULL	
WSTG-v42-INPV-01	https://owasp.org/www-project-web-security-testing-guide/...
OWASP_2017_A07	https://owasp.org/www-project-top-ten/2017/A7_2017-Cros...
POLICY_DEV_CICD	

Cross Site Scripting (Reflected) — отражённое межсайтовое выполнение скриптов (XSS).

Данная XSS-уязвимость позволяет злоумышленнику внедрить и выполнить JavaScript-код на сайте через параметр URL, если жертва перейдёт по специально подготовленной ссылке. Скрипт выполняется немедленно после перехода и может привести к краже сессии, подмене контента и выполнению любых действий от лица пользователя. Уязвимости CWE-79.

4 SQL Injection

SQL Injection
URL: http://testaspnet.vulnweb.com/Comments.aspx?id=4-2
Risk:  High
Confidence: Medium
Parameter: id
Attack: 4-2
Evidence:
CWE ID: 89
WASC ID: 19
Source: Active (40018 – SQL Injection)
Input Vector: URL Query String

Description:
SQL injection may be possible.


Other Info:
The original page results were successfully replicated using the expression [4-2] as the parameter value
The parameter value being modified was stripped from the HTML output for the purposes of the comparison.

Solution:
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'


Reference:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Alert Tags:

Key	Value
POLICY_QA_STD	
POLICY_QA_FULL	
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injec...
POLICY_DEV_CICD	

Current Status 

Суть уязвимости заключается в том, что веб-приложение не корректно обрабатывает входящий параметр id (id=4-2) в URL и позволяет злоумышленнику внедрять пользовательский ввод напрямую в SQL-запрос, что может привести к исполнению произвольного SQL-кода на сервере базы данных. Уязвимости CWE-89.

URL: <http://testaspnet.vulnweb.com/ReadNews.aspx?NewsAd=ads/def.html&id=2>
 Risk:  High
 Confidence: Medium
 Parameter: id
 Attack: 2 WAITFOR DELAY '0:0:15' --
 Evidence:
 CWE ID: 89
 WASC ID: 19
 Source: Active (40027 – SQL Injection – MsSQL)
 Input Vector: URL Query String

Description:
SQL injection may be possible.

Other Info:
The query time is controllable using parameter value [2 WAITFOR DELAY '0:0:15' --], which caused the request to take [15,350] milliseconds, when the original unmodified query with value [2] took [0] milliseconds.

Solution:
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Alert Tags:

Key	Value
POLICY_SEQUENCE	
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/
CWE-89	https://cwe.mitre.org/data/definitions/89.html
WSTG-v42-INPV-05	https://owasp.org/www-project-web-security-testing-guide/...
POLICY_DEV_FULL	

Такой же тип уязвимости SQL Injection, подтверждённая использованием конструкции:

`id=2 WAITFOR DELAY '0:0:15' --`

Значение параметра id, переданного через строку запроса, не фильтруется должным образом. Злоумышленник смог внедрить SQL-команду WAITFOR DELAY, что привело к искусственной задержке обработки запроса на сервере (на 15 секунд). Уязвимости CWE-89.

5 Absence of Anti-CSRF Tokens

```
<!-- InstanceEndEditable -->
<!--end content -->

<div id="navBar">
  <div id="search">
    <form action="search.php?test=query" method="post">
      <label>search art</label>
      <input name="searchFor" type="text" size="10">
      <input name="goButton" type="submit" value="go">
    </form>
  </div>
</div>
```

Spider Active Scan +

Absence of Anti-CSRF Tokens

URL: http://testphp.vulnweb.com/artists.php

Risk: Medium

Confidence: Low

Parameter:

Attack:

Evidence: <form action="search.php?test=query" method="post">

CWE ID: 352

WASC ID: 9

Source: Passive (10202 - Absence of Anti-CSRF Tokens)

Input Vector:

Description:

No Anti-CSRF tokens were found in a HTML submission form.
A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable

Other Info:

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token, _csrfToken] was found in the following HTML form: [Form 1: "goButton" "searchFor"].

Solution:

Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
<https://cwe.mitre.org/data/definitions/352.html>

Alert Tags:

Key	Value
OWASP_2021_A01	https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Выявлена уязвимость Absence of Anti-CSRF Tokens в HTML-форме. Это означает, что при отправке формы на сервер не используется уникальный токен, предотвращающий межсайтовые подделки запросов (CSRF). Злоумышленник может заставить пользователя выполнить нежелательное действие на сайте от его имени, подделав POST-запрос, так как форма не содержит анти-CSRF токена. При отсутствии такого токена сервер не может отличить легитимный запрос пользователя от сгенерированного злоумышленником, что позволяет атаки типа CSRF. Уязвимости CWE-352.

6 Content Security Policy (CSP) Header Not Set

Content Security Policy (CSP) Header Not Set
URL: http://testphp.vulnweb.com/sitemap.xml
Risk:  Medium
Confidence: High
Parameter:
Attack:
Evidence:
CWE ID: 693
WASC ID: 15
Source: Passive (10038 – Content Security Policy (CSP) Header Not Set)
Alert Reference: 10038-1
Input Vector:
Description:
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of
Other Info:
Solution:
Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.
Reference:
https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
<https://www.w3.org/TR/CSP/>
Alert Tags:

Key	Value
-----	-------

CSP — это дополнительный уровень безопасности веб-приложений, который позволяет ограничить источники исполнения скриптов, стилей и других ресурсов на сайте. При отсутствии этого заголовка сайт становится более уязвимым к атакам типа XSS (межсайтовый скриптинг) и инъекциям данных. Уязвимости CWE-693.

Результаты

Освоил базовые навыки динамического тестирования безопасности (DAST) на примере тестового приложения <http://testaspnet.vulnweb.com>.