Тема 5. Методи захисту від Cross-Site Scripting

Cross-site scripting (XSS) - це тип атак, при яких у вивід веб-додатків інжектується деякий JS-код. Уразливість виникає через відсутність фільтрації, коли дані, введені користувачам, використовуються для формування HTTP-відповіді.

За класифікацією OWASP TOP-10 2017 XSS-атака выдноситья до класу A3.

Для впровадження зловмисного коду на сторінку використовується текст «не по правилам»:

1) Найменування тега відділяється від атрибутів тільки \r\t\n\s символами.
2) HTML Entities, представлені в 10/16-ій формі повинні бути приведені до свого звичайного виду
3) Заборона на використання DOM-подій.
4) Заборона на включення в атрибут inline javascript або vbscript.

Атака відбувається не на сервер, а на користувача (відвідувача сайту). Найчастіше XSS-атака використовується для кражі авторизаційних cookies.

Для того щоб виявити XSS, необхідно перевірити на її наявність всі потенційно вразливі елементи сайту: поля вводу тексту, який далі буде відображатись на сайті.

Існує декілька видів класифікації XSS

1) По вектору
   - Reflected (Відбиті, непостійні)

     Атака спрацьовує коли користувач переходить по спеціально підготовленому посиланню.
   - Stored (Збережені, постійні)

2) По способу впливу існує два типи XSS-вразливості:

1. Активна
2. Пасивна.

Пасивна XSS

Пасивні XSS атаки потребують від користувача (жертви) безпосередньої участі. Наприклад, форма пошуку на форумі.

Користувача заманюють на підставний сайт різноманітними посиланнями.

Пасивна XSS може застосовувати як GET-параметри, так і POST-параметри запиту.

Активна XSS

html/js код зберігаєтья в базу і виконуєтья при кожному його виводі із бази (наприклад, пост на форумі, дані профілю і т. д.)

Активні XSS атаки потребують від користувача (жертви)

XSS через DOM

XSS-атаки через DOM можливі завдяки недостатній обробці на рівні JavaScript таких об'єктів DOM, як document.URL, document.location, document.referrer і деяких інших. Принциповою відмінністю є те, що дані взагалі не вбудовуються в HTML-код.

Наприклад, вразливий код.

<script>

var pos=document.URL.indexOf("name=")+5;

document.write(document.URL.substring(pos,document.URL.length));
</script>
Атака:
http://site/page.htm?name=<script>alert(document.cookie)</script>

Більшість рішень по запобіганню XSS використовують жорстку фільтрацію потенційно небезпечних конструкцій.

Виявлення XSS-уразливостей

Owasp Xelenium - XSS Scanner

Базується на фреймворку Selenium для автоматизації дій користувача в браузері.

XSS Me – доповнення Firefox

Тестує форми і видає результат у вигляді таблиці «XSS Heuristic Test Results»

	;	\	/	<	>	“	‘	=
searchform::s								
searchform::unnamed field								
unamed form::log								
unamed form::pwd								
unamed form::submit								
unamed form::redirect_to								

Способи захисту

Екранування вхідних даних

Функція htmlspecialchars()

Перетворює спеціальні символи в HTML сутності.

- '&' (амперсанд) перетворюється в '&'
- '"' (подвійні лапки) перетворюється в '"' when ENT_NOQUOTES is not set.
- ''' (одиночні лапки) перетворюється в ''' только в режиме ENT_QUOTES.
- '<' (знак "менше ніж") перетворюється в '<'
- '>' (знак "більше ніж") перетворюється в '>'

$name = htmlspecialchars($_POST['name'], ENT_QUOTES);

Функція htmlentities()

Перетворює всі символи у відповідні HTML сутності(для тих символів, для яких HTML сутності існують)

$name = htmlentities($_POST['name'], ENT_QUOTES, "UTF-8");

Функція strip_tags()

Видаляє HTML і PHP теги із строки.

$name = strip_tags($_POST['name']);

Функція filter_var()

Фільтрує змінну за вказаним фільтром

- FILTER_SANITIZE_STRING – видаляє тег / спеціальні символи із строки
- FILTER_SANITIZE_ENCODED – Видаляє/зашифровує спеціальні символи
- `FILTER_SANITIZE_URL` – Видаляє всі недопустимі символи із URL.

Заголовок X-XSS-Protection

- X-XSS-Protection: 0
- X-XSS-Protection: 1
- X-XSS-Protection: 1; mode=block
- X-XSS-Protection: 1; report=<reporting-uri>

Наприклад:
PHP:

header("X-XSS-Protection: 1; mode=block");

Apache (.htaccess):

```
<IfModule mod_headers.c>
  Header set X-XSS-Protection "1; mode=block"
</IfModule>
```

Встановлення кодування на кожній сторінці

```
<?php
header("Content-Type: text/html; charset=utf-8");
?>
<!DOCTYPE html>
<html>
 <head>
   <title>Charset</title>
   <meta charset="utf-8">
 </head>
```

У файлі .htaccess веб-сервера Apache

Розробники не повинні використовувати кодування CESU-8, UTF-7, BOCU-1 та кодування SCSU. В кодуванні UTF-7 можна обійти фільтрацію таких символів як ‘<’ и ‘"’.

Використання флагу HttpOnly

Цей флаг робить клієнтські куки недоступними через мови сценаріїв, такі як JavaScript. Активація цієї функції може здійснюватися в декількох місцях:

- У файлі php.ini

session.cookie_httponly = True

- В скрипті за допомогою функції session_set_cookie_params()

void session_set_cookie_params (int $lifetime [, string $path [, string $domain [, bool $secure = false [, bool $httponly = true]]]])

- У веб-додатку за допомогою функції setcookie()

bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path [, string $domain [, bool $secure = false [, bool $httponly = true]]]]]])

Заголовок Content Security Policy (CSP)

Одним із головних принципів безпеки браузерів і вебу взагалі є Same Origin Policy – дослівно «політика єдиного джерела». Її суть полягає у перевірці трьох компонентів із яких складається Origin: протокол, хост і порт.

Content Security Policy (CSP, політика захисту контенту) – це механізм забезпечення безпеки, за допомогою якого можна захищатися від атак із впровадженням контенту, наприклад, міжсайтового скриптингу (XSS, cross site

scripting). CSP описує безпечні джерела завантаження ресурсів, встановлює правила використання вбудованих стилей, скриптів, а також динамічної оцінки JavaScript – наприклад, за допомогою eval. Завантаження з ресурсів, які не входять до «білого» списку блокується.

Для використання політики сторінка повинна містити заголовок `Content-Security-Policy` із однією чи більше директивами, які представляють «білі списки». Підтримуються наступні директиви:

- default-src – перелічуються дозволені джерела за замовчуванням для інших директив. Якщо директива не вказана у заголовку, то політика застосовується згідно списку default-src.
- script-src – визначає безпечні джерела JavaScript.
- object-src – визначає безпечні джерела плагінів, таких як <object>, <embed> або <applet>.
- style-src – визначає безпечні джерела стилів
- img-src – визначає безпечні джерела зображень
- media-src – визначає безпечні джерела аудіо або відео, таких як HTML5 <audio>, <video>
- frame-src – визначає безпечні джерела для завантаження фреймів.
- font-src – визначає безпечні джерела шрифтів
- connect-src – застосовується до XMLHttpRequest (AJAX), WebSocket або EventSource

Прописування заголовка

```
# Apache config
Header set Content-Security-Policy "default-src 'self';"
# PHP example
header("Content-Security-Policy: default-src 'self'");
# Node.js example
request.setHeader("Content-Security-Policy", "default-src 'self'");
```

Використання бібліотек для кодування вхідних даних

HTML Purifier

У відповідності до конфігурації бібліотека очищує будь-який html код від усіх зловмисних, невалідних, заборонених(конфігурацією) частин коду, в тому числі окремих атрибутів.

ESAPI (The OWASP Enterprise Security API)

AntiSamy API (OWASP AntiSamy Project)

XSS-HTML-Filter

Open Sourced HTML filtering utility for Java.

Технологія XSS Auditor в браузерах

XSS_AUDITOR вбудований в деякі браузери, наприклад, Google Chrome.

ERR_BLOCKED_BY_XSS_AUDITOR

Internet Explorer видає повідомлення «Internet Explorer изменил эту страницу для предотвращения запуска межсайтовых сценариев».

Opera, Chrome, Firefox замінює спеціальні символи в шістнадцятирічний формат.