УДК: 004.49.8:004.056.53:004.738.1

АНАЛІЗ ТА МЕТОДИ ПРОТИДІЇ ВРАЗЛИВОСТЯМ XSS У СУЧАСНИХ ВЕБДОДАТКАХ

Комок В.С. **–** гр. БІК-1-22, komok1vova22@gmail.com
Калашник В.Ю. – к.т.н, ас., kalashnyk.vy@knutd.edu.ua
Київський національний університет технологій та дизайну

Мета роботи є дослідження основних типів XSS-атак, їх впливу на безпеку сучасних вебдодатків, аналіз ефективності різних методів запобігання цим вразливостям, а також розробка рекомендацій щодо їх застосування.

XSS (Cross-Site Scripting) атаки залишаються однією з найбільш поширених та небезпечних вразливостей вебдодатків, що дозволяють зловмисникам впроваджувати шкідливий код на сторінки вебсайту, що переглядаються іншими користувачами. За даними OWASP (Open Worldwide Application Security Project), XSS входить до трійки найбільш критичних вразливостей вебдодатків, що може призвести до крадіжки даних, зміни вмісту сторінки, перенаправлення користувачів на шкідливі сайти та інших негативних наслідків. Особливо вразливими до XSS-атак є вебдодатки, які обробляють велику кількість користувацького контенту, такі як соціальні мережі, форуми та блоги.

У роботі було проведено аналіз основних типів XSS-атак, таких як збережені (stored), відображені (reflected) та DOM-based XSS. Для аналізу використовувалися інструменти OWASP ZAP (Zed Attack Proxy) та Burp Suite, які дозволили виявити вразливості в тестових вебдодатках на локальному хості і були проведені в режимі manual attacks. Також було досліджено ефективність різних методів запобігання цим атакам:

- Екранування та фільтрація вхідних даних: аналіз ефективності бібліотек HTMLPurifier та DOMPurify.
- Робота з функціями валідації filter_var()
- Використання Content Security Policy (CSP): тестування різних директив CSP, таких як default-src, script-src та style-src.
- Контекстне екранування вихідних даних: аналіз ефективності використання функцій PHP htmlspecialchars() та JavaScript textContent.
- Використання HTTP-only cookie: дослідження впливу на безпеку сесій.

Аналіз показав, що найбільш ефективним методом запобігання XSS-атакам є комплексний підхід, що включає екранування та фільтрацію даних на

сервері, контекстне екранування даних на клієнтській сторні, а також використання CSP для обмеження можливостей виконання шкідливого коду. Наприклад, при тестуванні вебдодатку з використанням OWASP ZAP було виявлено, що без використання CSP успішно виконувалися відображені XSS-атаки з використанням JavaScript. Проте, після застосування CSP з директивами default-src 'self'; script-src 'self'; атаки були заблоковані. Важливо уникати використання директиви 'unsafe-inline', оскільки вона зменшує безпеку сайту, також посилення очищення даних потрібно робити через контекстне екранування, приклад серверної частини PHP: htmlspecialchars($_POST['name'], ENT_QUOTES, 'UTF-8');

Також було виявлено, що використання контекстного екранування даних на клієнтській сторні, за допомогою бібліотеки DOMPurify, значно знижує ризик успішної XSS-атаки, навіть у випадках, коли дані не були належним чином екрановані на сервері. Використання бібліотеки DOMPurify в JavaScript: const clean = DOMPurify.sanitize(' Hello ');

Висновки. В результаті проведеного дослідження було підтверджено ефективність різних методів фільтрації та екранування даних, а також важливість використання Content Security Policy (CSP) для запобігання XSS-атакам. Було показано, що комплексний підхід до захисту вебдодатків, що включає використання кількох методів захисту, є найбільш ефективним. Рекомендується інтегрувати ці методи захисту в процес розробки вебдодатків на ранніх етапах, а також регулярно проводити тестування безпеки за допомогою автоматизованих інструментів. Перспективними напрямками досліджень є розробка нових методів автоматизованого аналізу безпеки та використання машинного навчання для виявлення XSS-атак.

Література

1. OWASP Cross-site Scripting (XSS) Prevention Cheat Sheet: [Електронне посилання] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
2. Content Security Policy Level 3: [Електронне посилання] https://www.w3.org/TR/CSP3/
3. DOMPurify: Fast, pure DOM-based XSS sanitizer: [Електронне посилання] https://github.com/cure53/DOMPurify
4. Excess XSS [Електронне посилання] https://excess-xss.com/
5. Testing for Reflected Cross Site Scripting [Електронне посилання] https://owasp.org/
6. Testing for Stored Cross Site Scripting [Електронне посилання] https://owasp.org/