УДК 681.04 DOI: 10.26565/2519-2310-2021-2-01

ОЦІНКА ЕФЕКТИВНОСТІ СКАНЕРІВ БЕЗПЕКИ ВЕБ-ДОДАТКІВ

Олексій Прищепа, Дмитро Іваненко

Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
pryshchepa2020kb51@student.karazin.ua, i8o.dima@gmail.com

Рецензент: В'ячеслав Калашников, д. ф.-м.н., проф., Технологічний університет Монтеррея, 64849 Монтеррей, Нуево-Леон, Мексика
kalash@itesm.mx

Надійшло: Жовтень 2021.

Анотація: *Рівень захищеності веб-додатків з кожним роком постійно зростає, але нові рейтинги найбільш поширених загроз безпеки, свідчать проте, що проблема забезпечення їх захищеності є дуже актуальною, та постійно змінною. Тому, вкрай актуально розуміти важливість використання автоматичних сканерів безпеки веб-додатків та вміти об'єктивно оцінювати їх реальну ефективність. У роботі розглянуто процес тестування веб-додатків на наявність вразливостей (та приклади їх виявлення), з використанням безкоштовних веб-сканерів (з відкритим кодом) методом «чорного ящику». Тобто, в даному випадку, сканери взаємодіють з додатками так само, як і типовий користувач через веб-інтерфейс, за допомогою протоколу HTTP. Головною метою проведених тестувань є порівняння декількох сканерів з відкритим кодом, та визначення їх ефективності. Підкреслено, що оцінити всі показники сканерів не є можливим, внаслідок існування багатьох факторів. - Тому, в межах даної роботи, усі судження та висновки були зроблені тільки на основі аналізу отриманих звітів кожного тестового сканеру. В статі приведені відомості, щодо окремих параметрів та кількості знайдених вразливостей. Отримані результати тестувань свідчать, про те, що практика використання тільки одного сканеру э мало ефективною, тому при тестуванні потрібно використовувати одразу декілька різних рішень. Це надасть змогу отримати більш об'єктивні результати, в частині детектування, як вже відомих загроз безпеки, так і знаходження нових вразливостей (з їх додаванням до звіту). Робота буде корисна для тих, хто цікавиться питаннями оцінки стану безпеки сучасних веб-додатків.*

Ключові слова: *вразливості веб-додатків; сканери вразливостей; ефективність сканерів; тестування безпеки веб-додатків, пентестинг.*

1 Вступ

Проблема небезпечного програмного забезпечення (ПЗ) є, найважливішою технічною проблемою нашого часу, яка з кожним роком не перестає бути актуальною. Різке зростання числа веб-додатків, за останні декілька років, що забезпечують роботу бізнесу, комунікацію через соціальні мережі, розваги та сервіси, пошуку інформації і т.д. Лише посилило вимоги до створення надійного методу до написання й захисту веб-додатків та даних у них.

Зрозуміло, що неможливо створити безпечний веб-додаток, не проводячи, при розробці та на стадії впровадження, тестування безпеки. Тестування є частиною більш широкого підходу до створення безпечної системи. Багато організацій, які займаються розробкою програмного забезпечення не включають, чи включають не повністю, тестування безпеки до свого стандартизованого процесу розробки програмного забезпечення.

Само по собі тестування безпеки не є особливо гарним показником того, наскільки захищений веб-додаток, тому що існує безліч способів, якими зловмисник може зламати додаток, і просто неможливо перевірити їх всі, але тестування безпеки може знайти прості у реалізації, та популярні вразливості. [1-2]

2 Основна частина

Одна з цілей тестування безпеки полягає в тому, щоб перевірити, що засоби управління безпекою працюють належним чином. Це задокументовано за допомогою вимог безпеки, які описують функціональність контролю безпеки. На високому рівні це означає підтвердження

конфіденційності, цілісності і доступності даних, а також послуг. Інша мета полягає в тому, щоб перевірити, що засоби управління безпекою реалізовані практично без вразливостей.

Тестування рівня безпеки, як в ручному режимі, так і автоматично, проходить у відповідності з наступними етапами [2]:

- Розвідка – програма чи людина намагається дізнатися якомога більше інформації про встановлену систему. Це включає в себе спроби визначити яке програмне забезпечення використовується та якої версії, які точки входу існують, пошук прихованого контенту, відомих вразливостей та інших ознак слабкості.
- Атака – на цьому етапі тестувальник намагається використовувати відомі або передбачувані вразливості, щоб довести їх існування.
- Звіт – цей документ є результатом роботи тестувальника або програми, до якого, як правило, включається перелік вразливостей, які можливо експлуатувати *(в деяких випадках включає потенційні вразливості)*, ступінь їх ступінь їх небезпеки, вірогідні наслідки і можливі варіанти вирішення цієї загрози та іншу додаткову інформацію.

Кінцевою метою тестування на безпечність – є пошук вразливостей, для подальшого їх усунення, а також для перевірки та підтвердження, що вразливості були виправлені.

Якщо говорити про порівняння сканерів вразливостей, то можна розглянути наступні завдання [3]:

1) порівняння функціональних можливостей сканерів;
2) прийняття чогось за еталон тестування та перевірка сканерів з метою визначення ефективності їх роботи з різними видами веб-додатків.

Перше завдання *(порівняння функціональних можливостей сканерів)* можливо вирішити прийнявши за основу порівняння контрольний список, наприклад створений на основі комбінації критеріїв представлених в проектах *«OWASP top-10»* або *«Web Application Security Scanner Evaluation Criteria»* та інших, можливостей ідеального сканера і оцінити інструменти на підтримку цих можливостей. В результаті отримаємо таблицю відповідності за даними критеріями. Порівняння може бути проведено, як теоретично, так і практично.

Друге завдання, поки ще, не має остаточного рішення.

Під ефективністю роботи сканерів вразливостей, будемо розуміти сукупність з двох компонентів: – повноти відповіді та частоти помилкових спрацьовувань. Повнота визначається, як відношення кількості виявлених вразливостей до загальної кількості вразливостей у додатку. Частота помилкових спрацьовувань визначається як відношення кількості неправильно знайдених вразливостей до загальної кількості повідомлених вразливостей. Теоретично, «ідеальний» інструмент повинен демонструвати повноту рівну одиниці, а частоту помилкових спрацьовувань як нуль [2].

Слід зазначити, що даний метод має два основних недоліки:

- зазначені показники залежать від тестової вибірки;
- визначення ефективності є марним з практичної точки зору.

Перший недолік показує, що показниками ефективності при порівнянні інструментів між собою можна маніпулювати, включаючи або виключаючи з тестової вибірки для визначення еталону ті чи інші веб-додатки або вразливості в них.

Другим недоліком є те, що сканери відрізняються один від одного, і коли, наприклад, один виявляє, в тестовій вибірці, всі вразливості до атак класу *XSS*, але не помічає жодної вразливості до атак інших класів, тоді як другий, навпаки, виявляє всі вразливості до атак класу *SQL*-ін'єкції, і «не бачить» жодної вразливості до атак інших класів. В такому разі ефективність, яка може бути задана у висновку - це кількісні характеристики тестової вибірки по класах вразливостей до атак *XSS* і *SQL*- ін'єкції, відповідно.

Також важливим фактором, при порівнянні ефективності веб-сканерів, є оцінка ефективності їх пошукових машин – «краулерів». Так як у автоматичних засобів сканування першим етапом аналізу є збір даних про веб-додаток та визначення точок входу, полів для вве-

дення даних, то неточні результати (на цьому етапі) істотно впливають на можливість знайти вразливості при тестуванні знайдених місць входу.

Отже переходячи з класифікації сканерів до вразливостей слід підкреслити, що сканери безпеки веб-додатків виявляють вразливості у таких основних категоріях як [2-3]:

- Вразливості на етапі кодування. Найкраще виявляються, серед вразливостей, пов'язані з некоректною обробкою вхідних і вихідних даних, у випадках коли може виникнути логічна помилка та вразливість.
- Вразливості на етапі впровадження та налаштування програми. До цих вразливостей відносяться некоректні налаштування оточення веб-додатку: веб-серверу, сервера додатків, *SSL/TLS*, фреймворків, сторонніх компонентів, увімкнутий режим *DEBUG* і т.д.
- Вразливості на етапі експлуатації серверу та ПЗ на ньому. До цих вразливостей, наприклад, відносяться: – використання застарілого ПЗ; занадто прості паролі; зберігання конфіденційних даних та архівних копій на веб-сервері в загальному доступі, та в не шифрованому вигляді; наявність у доступі службових модулів та компонентів і т.і.

Завдання пошуку вразливостей 2-го і 3-го типів автоматизується досить непогано При цьому, завдання пошуку вразливостей етапу розробки набагато успішніше вирішується при застосуванні сканерів саме в ручному режимі. Однак, слід зауважити, що в межах даної роботи, особливості аналізу веб-додатків с залученням ручного методу не розглядається, так як метою статті є аналіз автоматизованих сканерів безпеки.

В автоматичному режимі роботи сканеру, від тестувальника вимагається тільки правильно налаштувати параметри його запуску: додати URL, логін / пароль користувача, обрати тип сканування, глибину пошуку, кількість одночасних сканувань та інші можливості. Типовими представниками даного класу є наступні інструменти: *OWASP ZAP; Nikto; Wapiti* та ін.

3 Тестування

В мережі Інтернет, на запит *«Безкоштовні сканери веб-додатків»*, отримаємо велику кількість прикладів реалізації сканерів. Деякі з них здатні вирішувати вузькоспеціалізовані задачі, як, наприклад, програми перебору паролів чи пошуку відкритих файлів і директорій (*т.з.* «*краулери*»), та ін. Але, серед всього цього розмаїття, є «повноцінні» системи, які здатні виконувати сканування більшою кількістю методів. Розглянемо деякі з них нижче [4-6].

Так, у дослідному тестуванні використовувалися наступні відомі безкоштовні сканери:

- *OWASP ZAP* – це безкоштовний інструмент з відкритим кодом для тестування на проникнення, який підтримується під егідою *OWASP*. *ZAP* розроблений спеціально для тестування веб-додатків і є досить гнучким, та здатним до розширеня. В тестуванні використовувався *ZAP* сканер в версії 2.10.0.
- *Wapiti* – сканер вразливостей веб-додатків, який дозволяє перевіряти безпеку веб-сайтів або веб-програм. Виконує сканування методом «чорного ящика», тобто скануючи веб-сторінки та шукаючи сценарії і форми, де він може вводити дані. До експерименту залучався *Wapiti* в версії 3.0.5.
- *Nikto* - це сканер веб-додатків з відкритим кодом, який виконує комплексні тести веб-додатків для кількох елементів, включаючи понад 6700 потенційно небезпечних файлів/програм. До тестів залучалась версія *Nikto* сканеру 2.1.6.

Тестуванню піддавалися три веб-додатку [7-9]: - *DVWA, Juice Shop* та *WebGoat.*

DVWA - це веб-програма *PHP/MySQL*, яка є надзвичайно вразливою. Метою *DVWA* є відпрацювання однієї з найпоширеніших веб-вразливостей, з різними складними рівнями, та зрозумілим інтерфейсом. Версія додатку на момент експериментів - 1.10.

Juice-Shop – сучасний небезпечний веб-додаток, котрий «написаний» на JavaScript. Додаток містить величезну кількість завдань різного рівня складності. Його з успіхом можна використовувати для демонстрацій, та для тренуваннь. *Juice Shop* охоплює вразливості

OWASP Top-10, разом з багатьма іншими уразливостями, які зустрічаються в реальних додатках. Версія додатку на момент експериментів – v.12.9.3.

WebGoat – це, навмисно, небезпечний додаток, який дозволяє розробникам перевіряти вразливості, які зазвичай зустрічаються в додатках на базі *Java*, котрі використовують загальні та/або популярні компоненти з відкритим кодом. Тестова версія - 8.2.2.

Завантаження кожного з дослідних веб-додатків, виконується у своєму контейнері:

docker run -d --name juice --net locnet --restart always -p 3000:3000 bkimminich/juice-shop &&
docker run -d --name dvwa --net locnet --restart always -p 8000:80 sagikazarmark/dvwa &&
docker run -d --name goat --net locnet --restart always -p 8080:8080 webgoat/goatandwolf && docker ps

На рис. 1 представлений скріншейвер *(Skr)*, якій демонструє, що веб-додаткі вже запущені і працюють.

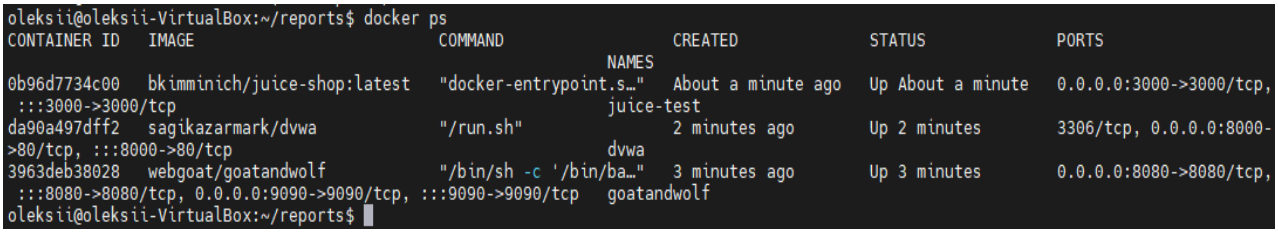


Рис. 1 – Scr працюючих веб-додатків

Визначимо їх адреси (рис. 2) командою «*docker inspect»,* для подальшого вказування їх у веб-сканерах.

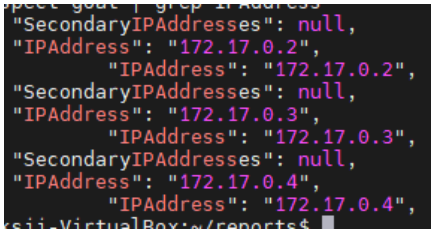


Рис. 2 – Визначення адрес працюючих веб-додатків

Враховуючи те, що обсяг статті доволі обмежений, виключимо розгляд сутності проведення перших двох етапів (*розвідування і атаки*), та зосередимось саме на розгляді найбільш важливої, для користувача частині – *Звіті*. Він є головною частиною, бо повинен зрозуміло та чітко надати відомості користувачу про знайдені у веб-додатку вразливості. При цьому, зовсім неважливо, як саме проводяться перші два етапу (розвідка і атаки), коли звіт буде не зрозумілим, та не містить потрібної суті. - Тому аналіз отриманих звітів є головним пунктом оцінки ефективності проведеного тестування.

Як вже було зазначено вище, пошук вразливостей проводився в 3-х тестових додатках: Juice-Shop, DVWA та WebGoat, які містять певні вразливості:

- *DVWA*: *XSS, SQLi, Insecure data, CSRF, Command injection, Misconfiguration.*
- *Juice-Shop: XSS, SQLi, Security Misconfiguration (SM), Sensitive Data Exposure (SDE), Vulnerable Components (VC), Cryptographic Issues (CI), Miscellaneous, Security through Obscurity (STO), Unvalidated Redirects (UR);*
- *WebGoat: XSS, SQLi, Sensitive Data Exposure, XXE, Vulnerable Components, Misconfiguration.*

Зазначимо пункти по яким порівнюються дані з відомостями звіту організації OWASP, «OWASP Top 10 2017 року» [1]. Варто зауважити, що є і нова версія, але для аналізу краще підходить стара, так як всі загрози, які були в ній, все ще залишаються актуальними.

OWASP Top 10, які можливо виявити:

- ін’єкції;
- міжсайтове виконання коду (*XSS*);

- розголошення конфіденційних даних (*Sensitive Data Exposure*);
- зовнішні XML сутності (*XXE*);
- невірне налаштування параметрів безпеки (*Security Misconfiguration*);
- використання компонентів з відомими вразливостями (*Vulnerable Components*).

Надамо стислий опис знайдених вразливостей для кожного веб-додатку.

OWASP ZAP:

- у *Juice-Shop* сканером OWASP ZAP було знайдено чотири невстановлених хедера: *Content Security Policy (CSP), X-Frame-Options, Deprecated Feature Policy, X-Content-Type-Options*; які призводять доступ *XML* файлі – а це зразу *XXE*. Інші загрози через неправильне налаштування серверу у кількості двох: *Bypassing 403, Cross-Domain Misconfiguration*. Виявлені загрози, які можуть призвести до розкриття конфіденційних даних у кількості п'ятьох: *Backup File Disclosure, Session ID in URL Rewrite, Source Code Disclosure – Java and PHP, Web Cache Deception*. Окрім розголошення даних *Source Code Disclosure – Java and PHP* призводить до розголошення версії продукту, призводячи пошук експлоїта набагато простіше. Також було знайдено одну *SQL* ін'єкцію.
- у *DVWA* було знайдено також невстановлені хедери у кількості три, з доступом до sitemap.xml. Виявлена *Application Error Disclosure* вразливість, яка при виникненні помилки, може призвести до розголошенню конфіденційної інформації. І відкритий доступ до директорій (*Directory Browsing, Directory Browsing - Apache 2*), що також може привести до розголошення даних.
- у *WebGoat* було знайдено один невстановлений хедер (*CSP*), вразливість високого рівня *Anti-CSRF* та невстановлений для нього токен.

Отже, у підсумку детекту для 3-х веб-додатків, маємо:

- *ін'єкцій – 1;*
- *міжсайтового виконання коду (XSS) -0;*
- *зовнішніх XML сутностей (XXE) – 2;*
- *загроз що сприяють розголошенні конфіденційних даних (SDE) – 7;*
- *параметрів безпеки з неправильним налаштуванням (SM) – 10;*
- *використання компонентів з вже відомими вразливостями (VC) – 2.*

По сканеру Nikto отримані наступні дані:

- *Juice-Shop* – сканер видав 167 потенційно небезпечні файли, але тільки один є реальним;
- *DVWA* – знайдено три хедери, та три відбитки ПЗ із застарілими версіями. А також, одну *CVE*, з отриманням усієї файлової структури;
- *WebGoat* – знайдено 1 невстановлений хедер, для сесійних *cookie*.

Отже, було знайдено:

- *ін'єкцій – 0;*
- *XSS – 0;*
- *XXE – 0;*
- *SDE – 2;*
- *SM – 4;*
- *VC – 0.*

Для сканера Wapiti отримані такі дані:

- *Juice-Shop* – детектовано велику кількість доступних посилань на файли *Backup*, які є помилковим викидом. Однак, при цьому є інша категорія - це потенційно небезпечні файли, які можуть призвести до розкриття конфіденційної інформації, та три невстановлених хедера і чотири відбитки версій використовуваних програм та фреймворків, що може призвести до можливості експлуатації експлойту.

- *DVWA* – знайдено п’ять потенційно небезпечних файлів та п’ять невстановлених хедера і три відбитки ПЗ, які розкривають версії використовуваних продуктів.
- *WebGoat* - знайдено чотири невстановлених хедера, два з яких відповідають за cookie, та п’ять відбитки версій використовуваних програм і фреймворків, що може призвести до експлуатації експлойту. Виявлена вразливість, типу *Internal Server Error*, яка може призвести до розголошенню конфіденційної інформації.

Таким чином, в цілому, було знайдено:

- ін’єкцій – *0;*
- *XSS – 0;*
- *XXE – 0;*
- *SDE – 5+;*
- *SM – 13;*
- *VC – 11.*

Так як жоден з тестових сканерів не знайшов вразливості типу *XSS*, то вилучимо її та побудуємо результуючу таблицю, що містить перелік відповідних загроз вразливостей.

Таблиця 1 – Зведений перелік загроз

Сканери	**Типи вразливостей**					
	SQLi	***XXE***	***SDE***	***SM***	***VC***	*Додаткові*
OWASP ZAP	1	2	7	11	2	1 - *CSRF*
Nikto	0	0	0	2	2	
Wapiti	0	0	5+	13	11	

Таким чином, як випливає з даних таблиці, за результатами сканування було знайдено велику кількість вразливостей у налаштуваннях сервера та визначено велику кількість потенційно небезпечних файлів.

4 Використані команди налаштувань та форми звітів

В загальному випадку, головною ідеєю використовуваних налаштувань веб-сканерів повинно бути отримання якомога більш інформативних даних їх звітів *(т.з. лог-файлів)*. При цьому, для більш детального огляду можливо скористатися наступним веб-ресурсом [10].

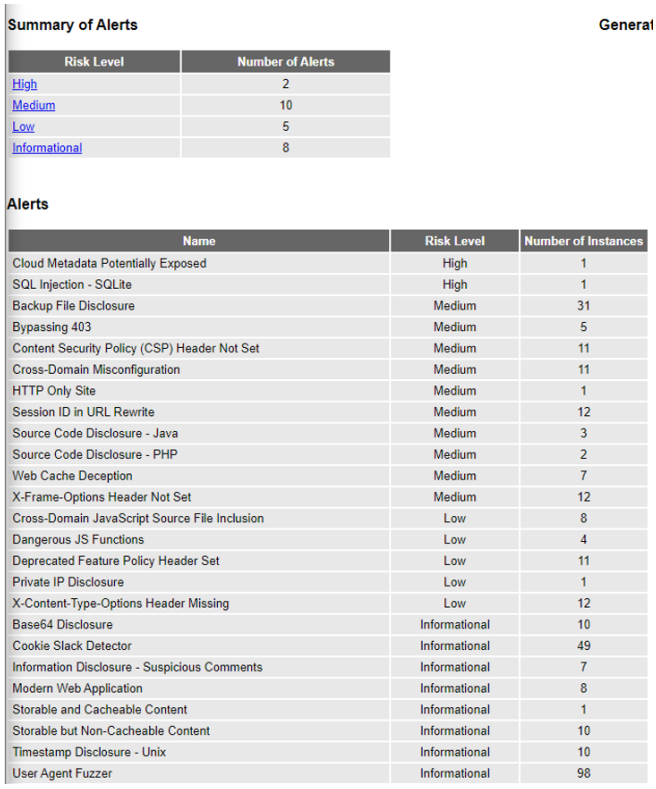
Для отримання сукупності відповідних звітів від сканеру *OWASP ZAP*, потрібно виконати наступні команди:

```
docker run --rm -d --name owasp_scanner_j -v $(pwd):/zap/wrk/:rw -t owasp/zap2docker-stable zap-full-scan.py -t http://172.17.0.2:3000/ -g gen.conf -m 10 -T 60 -I -j -r report_owasp_j.html &&
docker run --rm -d --name owasp_scanner_d -v $(pwd):/zap/wrk/:rw -t owasp/zap2docker-stable zap-full-scan.py -t http://172.17.0.3:80/vulnerabilities/ -g gen.conf -m 10 -T 60 -I -j -r report_owasp_d.html &&
docker run --rm -d --name owasp_scanner_g -v $(pwd):/zap/wrk/:rw -t owasp/zap2docker-stable zap-full-scan.py -t http://172.17.0.4:8080/WebGoat/ -g gen.conf -m 10 -T 60 -I -j -r report_owasp_g.html && docker ps
```

Нижче, на рис. 3, представлені вразливості, які були знайдені сканером ***ZAP*** (відповідно, для: - *Juice shop, DVWA, WebGoat*).

Наступний звіт від сканеру ***Nikto***, для котрого потрібно виконати наступні команди:

```
docker run --rm -d --name nikto_scanner_j -v $(pwd):/tmp olexii21/nikto_scanner -url http://172.17.0.2:3000/ -Plugins "@@ALL" -id "test@com.com:password" -Format html -output tmp/nikto_report_j.html &&
docker run --rm -d --name nikto_scanner_d -v $(pwd):/tmp olexii21/nikto_scanner -url http://172.17.0.3:80/vulnerabilities/ -Plugins "@@ALL" -id "admin:password" -Format html -output tmp/nikto_report_d.html &&
docker run --rm -d --name nikto_scanner_g -v $(pwd):/tmp olexii21/nikto_scanner -url http://172.17.0.4:8080/WebGoat/ -Plugins "@@ALL" -id "admin:password" -Format html -output tmp/nikto_report_g.html && docker ps
```



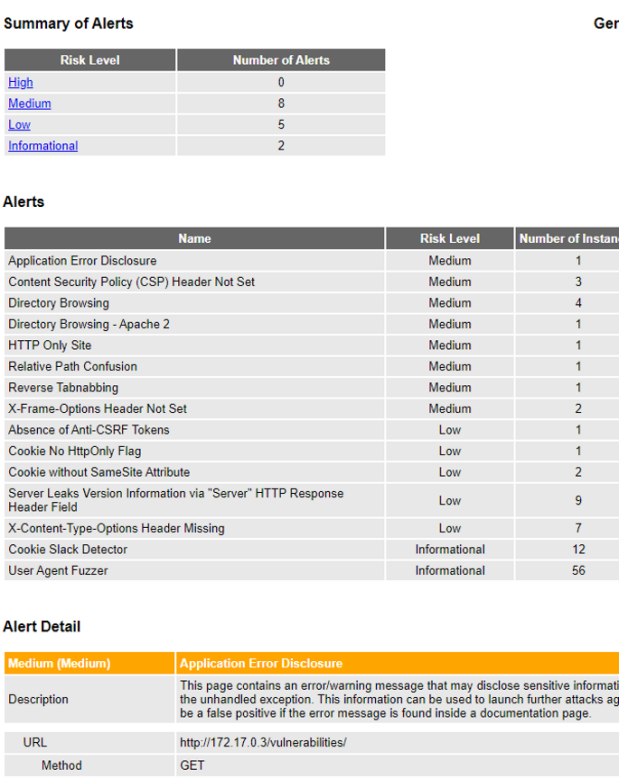
Summary of Alerts

Risk Level	Number of Alerts
High	2
Medium	10
Low	5
Informational	8

Alerts

Name	Risk Level	Number of Instances
Cloud Metadata Potentially Exposed	High	1
SQL Injection - SQLite	High	1
Backup File Disclosure	Medium	31
Bypassing 403	Medium	5
Content Security Policy (CSP) Header Not Set	Medium	11
Cross-Domain Misconfiguration	Medium	11
HTTP Only Site	Medium	1
Session ID in URL Rewrite	Medium	12
Source Code Disclosure - Java	Medium	3
Source Code Disclosure - PHP	Medium	2
Web Cache Deception	Medium	7
X-Frame-Options Header Not Set	Medium	12
Cross-Domain JavaScript Source File Inclusion	Low	8
Dangerous JS Functions	Low	4
Deprecated Feature Policy Header Set	Low	11
Private IP Disclosure	Low	1
X-Content-Type-Options Header Missing	Low	12
Base64 Disclosure	Informational	10
Cookie Slack Detector	Informational	49
Information Disclosure - Suspicious Comments	Informational	7
Modern Web Application	Informational	8
Storable and Cacheable Content	Informational	1
Storable but Non-Cacheable Content	Informational	10
Timestamp Disclosure - Unix	Informational	10
User Agent Fuzzer	Informational	98

а) Juice shop



General Summary of Alerts Ger

Risk Level	Number of Alerts
High	0
Medium	8
Low	5
Informational	2

Alerts

Name	Risk Level	Number of Instan
Application Error Disclosure	Medium	1
Content Security Policy (CSP) Header Not Set	Medium	3
Directory Browsing	Medium	4
Directory Browsing - Apache 2	Medium	1
HTTP Only Site	Medium	1
Relative Path Confusion	Medium	1
Reverse Tabnabbing	Medium	1
X-Frame-Options Header Not Set	Medium	2
Absence of Anti-CSRF Tokens	Low	1
Cookie No HttpOnly Flag	Low	1
Cookie without SameSite Attribute	Low	2
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	9
X-Content-Type-Options Header Missing	Low	7
Cookie Slack Detector	Informational	12
User Agent Fuzzer	Informational	56

Alert Detail

Medium (Medium)	Application Error Disclosure
Description	This page contains an error/warning message that may disclose sensitive informati the unhandled exception. This information can be used to launch further attacks ag be a false positive if the error message is found inside a documentation page.
URL	http://172.17.0.3/vulnerabilities/
Method	GET

б) DVWA

ZAP Scanning Report

Summary of Alerts **Generated on Fri, 1 Oct 2021 14:44:08**

Risk Level	Number of Alerts
High	1
Medium	2
Low	4
Informational	1

Alerts

Name	Risk Level	Number of Instances
Anti-CSRF Tokens Check	High	1
Content Security Policy (CSP) Header Not Set	Medium	1
HTTP Only Site	Medium	1
Absence of Anti-CSRF Tokens	Low	1
Cookie No HttpOnly Flag	Low	1
Cookie Slack Detector	Low	3
Cookie without SameSite Attribute	Low	1
User Agent Fuzzer	Informational	14

Alert Detail

High (Medium)	Anti-CSRF Tokens Check
Description	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 URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.
URL	http://3.66.163.55:8080/WebGoat/login
Method	GET

в) WebGoat

Рис. 3 – Звіт для *ZAP*

В звітах сканеру ***Nikto*** (див. рис. 4) не має підсумкової таблиці, з усіма знайденими вразливостями, тому відображено тільки кінець з відповідною статистикою.

URI	/wp-content/plugins/nextgen-gallery/products/photocrati_nextgen/modules/nextgen_addgaller
HTTP Method	POST
Description	/wp-content/plugins/nextgen-gallery/products/photocrati_nextgen/modules/nextgen_addgaller nextgen-gallery-2-0-0/
Test Links	http://172.17.0.2:3000/wp-content/plugins/nextgen-gallery/products/photocrati_nextgen/mod http://172.17.0.2:3000/wp-content/plugins/nextgen-gallery/products/photocrati_nextgen/mod
References	
URI	/wordpress/wp-content/plugins/nextgen-gallery/products/photocrati_nextgen/modules/nextger
HTTP Method	POST
Description	/wordpress/wp-content/plugins/nextgen-gallery/products/photocrati_nextgen/modules/nextger traversal-in-nextgen-gallery-2-0-0/
Test Links	http://172.17.0.2:3000/wordpress/wp-content/plugins/nextgen-gallery/products/photocrati_ne http://172.17.0.2:3000/wordpress/wp-content/plugins/nextgen-gallery/products/photocrati_ne
References	

Host Summary	
Start Time	2021-10-01 12:05:00
End Time	2021-10-01 12:07:04
Elapsed Time	124 seconds
Statistics	7423 requests, 2 errors, 167 findings

Scan Summary	
Software Details	Nikto 2.1.6
CLI Options	-url http://172.17.0.2:3000/ -Plugins @@ALL -Format html -output tmp/nikto_report_j.html
Hosts Tested	1
Start Time	Fri Oct 1 12:05:00 2021
End Time	Fri Oct 1 12:07:04 2021
Elapsed Time	124 seconds

а) Juice shop

URI	/vulnerabilities///
HTTP Method	GET
Description	/vulnerabilities/// directory listing when /'s are requested.
Test Links	http://172.17.0.3:80/vulnerabilities/// http://172.17.0.3:80/vulnerabilities///
References	OSVDB-3288
URI	/vulnerabilities/fi/
HTTP Method	GET
Description	Cookie PHPSESSID created without the httponly flag
Test Links	http://172.17.0.3:80/vulnerabilities/fi/ http://172.17.0.3:80/vulnerabilities/fi/
References	

Host Summary	
Start Time	2021-10-01 13:45:28
End Time	2021-10-01 13:45:46
Elapsed Time	18 seconds
Statistics	8123 requests, 0 errors, 17 findings

Scan Summary	
Software Details	Nikto 2.1.6
CLI Options	-url http://172.17.0.3:80/vulnerabilities/ -Plugins @@ALL -Format html -output tmp/nikto_report_d.html
Hosts Tested	1
Start Time	Fri Oct 1 13:45:27 2021
End Time	Fri Oct 1 13:45:46 2021
Elapsed Time	19 seconds

б) DVWA

URI	/WebGoat/
HTTP Method	GET
Description	Cookie JSESSIONID created without the httponly flag
Test Links	http://3.66.163.55:8080/WebGoat/ http://3.66.163.55:8080/WebGoat/
References	
URI	/WebGoat/
HTTP Method	GET
Description	The web server may reveal its internal or real IP in the Location header via a requ
Test Links	http://3.66.163.55:8080/WebGoat/ http://3.66.163.55:8080/WebGoat/
References	OSVDB-630

Host Summary	
Start Time	2021-09-30 06:15:26
End Time	2021-09-30 06:25:45
Elapsed Time	619 seconds
Statistics	8125 requests, 0 errors, 2 findings

Scan Summary	
Software Details	Nikto 2.1.6
CLI Options	-url https://3.66.163.55:8080/WebGoat/ -Plugins @@ALL -id test2021:**** -Form
Hosts Tested	1
Start Time	Thu Sep 30 06:15:26 2021
End Time	Thu Sep 30 06:25:45 2021
Elapsed Time	619 seconds

в) WebGoat

Рис. 4 – Звіт для *Nikto*

Наступний звіт від сканеру ***Wapiti***, для котрого потрібно виконати наступні команди:

docker run --rm -d --name wapiti_scanner_j -v $(pwd):/tmp olexii21/wapiti_scanner wapiti -u http://172.17.0.2:3000/ --scope folder -m "all" -l 2 -d 5 -f html -o tmp/wapiti_report_j &&

docker run --rm -d --name wapiti_scanner_d -v $(pwd):/tmp olexii21/wapiti_scanner wapiti -u http://172.17.0.3:80/vulnerabilities/ --scope folder -m "all" -a "admin%password" -l 2 -d 5 -f html -o tmp/wapiti_report_d &&

docker run --rm -d --name wapiti_scanner_g -v $(pwd):/tmp olexii21/wapiti_scanner wapiti -u http://172.17.0.4:8080/WebGoat/ --scope folder -m "all" -a "admin21%password" -l 2 -d 5 -f html -o tmp/wapiti_report_g && docker ps

Нижче, на рис. 5, представлені вразливості, які були знайдені сканером ***Wapiti*** (відповідно, для: - *Juice shop*, *DVWA* та *WebGoat*).

5 Висновки

1. Як свідчать результати тестів, ефективність розглянутих сканерів, доволі низька, але попри цього сформовані ними звіти, в більшій мірі допоможуть дізнатися, чи правильно налаштований сервер, та видно чи ні директорії і файли в них, які можуть бути потенційно

Wapiti vulnerability report
Target: http://172.17.0.2:3000/
Date of the scan: Fri, 01 Oct 2021 13:44:54 +0000. Scope of the scan: folder

Summary

Category	Number of vulnerabilities found
Backup file	2372
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	1
Cross Site Request Forgery	0
Potentially dangerous file	141
Command execution	0
Path Traversal	0
Fingerprint web application framework	1
Fingerprint web server	0
Htaccess Bypass	0
HTTP Secure Headers	2
HttpOnly Flag cookie	0
Open Redirect	0
Secure Flag cookie	0
SQL Injection	0
Server Side Request Forgery	0
Subdomain takeover	0
Blind SQL Injection	0
Cross Site Scripting	0
XML External Entity	0
Internal Server Error	0
Resource consumption	0
Fingerprint web technology	3
HTTP Methods	0

а) Juice shop

Wapiti vulnerability report
Target: http://172.17.0.3:80/vulnerabilities/
Date of the scan: Fri, 01 Oct 2021 13:42:40 +0000. Scope of the scan: folder

Summary

Category	Number of vulnerabilities found
Backup file	0
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	1
Cross Site Request Forgery	0
Potentially dangerous file	5
Command execution	0
Path Traversal	0
Fingerprint web application framework	0
Fingerprint web server	1
Htaccess Bypass	0
HTTP Secure Headers	4
HttpOnly Flag cookie	0
Open Redirect	0
Secure Flag cookie	0
SQL Injection	0
Server Side Request Forgery	0
Subdomain takeover	0
Blind SQL Injection	0
Cross Site Scripting	0
XML External Entity	0
Internal Server Error	0
Resource consumption	0
Fingerprint web technology	2
HTTP Methods	0

б) DVWA

Wapiti vulnerability report
Target: http://172.17.0.4:8080/WebGoat/
Date of the scan: Fri, 01 Oct 2021 13:43:47 +0000. Scope of the scan: folder

Summary

Category	Number of vulnerabilities found
Backup file	0
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	1
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0
Fingerprint web application framework	0
Fingerprint web server	0
Htaccess Bypass	0
HTTP Secure Headers	1
HttpOnly Flag cookie	1
Open Redirect	0
Secure Flag cookie	1
SQL Injection	0
Server Side Request Forgery	0
Subdomain takeover	0
Blind SQL Injection	0
Cross Site Scripting	0
XML External Entity	0
Internal Server Error	4
Resource consumption	0
Fingerprint web technology	5
HTTP Methods	0

в) WebGoat

Рис. 5 – Звіт для *Wapiti*

небезпечними, з боку конфіденційності. При цьому, вразливості типу *SQLi*, *XSS* та інші, де треба вводити різні послідовності для перевірки, залишаються занадто «важкими» для безкоштовних автоматизованих сканерів безпеки. - Т.ч., для пошуку таких загроз краще використовувати спеціалізовані сканери для кожного типу вразливостей.

2. Функціональне порівняння сканерів, в цілому, зробити нескладно. - Маючи список функціональних критеріїв, можна отримати оцінку відповідності цим критеріям для кожного інструменту зі списку, на вибір.

3. З усіх сканері, які були підвергнути дослідному тестуванню, саме *OWASP ZAP*, об'єктивно, є найпотужнішим. Насамперед, це зумовлено тим, що це є великий та відкритий проект, який має досить представницьке ком'юніті розробників, котрі постійно вдосконюють сканер, та є прихильниками вільного розповсюдження відповідного ПЗ.

4. Сканери безпеки веб-додатків не забезпечують гарантованого захисту та виявлення усіх можливих проблем безпеки. Причина очевидна, універсального рішення не існує, перш за все, в наслідок постійної та швидкої еволюції небезпечного програмного забезпечення. В наслідок цього, ПЗ для оцінки безпеки веб-додатків, слід розглядати не більш, як засіб здійснення оперативного початкового пошуку шаблонних проблем, в багаторівневому механізмі аналізу проблем безпеки сучасних веб-додатків.

Посилання

[1] OWASP Top 10 – 2017. (2017). Вилучено з https://owasp.org/www-project-top-ten/
[2] OWASP. Руководство по тестированию веб-безопасности, (2020).
[3] Петухов Андрей. Как выбрать сканер уязвимостей веб-приложений? (2015). Вилучено з https://www.securitylab.ru/blog/personal/andrepetukhov/143697.php
[4] ZAP - Full Scan. (2021). Вилучено з https://www.zaproxy.org/docs/docker/full-scan/
[5] The web-application vulnerability scanner. (2021). Вилучено з https://wapiti.sourceforge.io/
[6] Nikto: A Practical Website Vulnerability Scanner. (2021). Вилучено з https://securitytrails.com/blog/nikto-website-vulnerability-scanner
[7] Damn Vulnerable Web Application Docker container (2018). Вилучено з https://github.com/opsxcq/docker-vulnerable-dvwa
[8] OWASP Juice Shop. (2021). Вилучено з https://github.com/juice-shop/juice-shop
[9] WebGoat 8: A deliberately insecure Web Application. (2021). Вилучено з https://github.com/WebGoat/WebGoat
[10] Scan Reports. (2021). Вилучено з https://github.com/G4rr/reports

Reviewer: Vyacheslav Kalashnikov, Doctor of Sciences (Physics and Mathematics), Full Prof., Department of Systems and Industrial Engineering, Campus Monterrey, Monterrey, Mexico.
E-mail: kalash@itesm.mx

Received on October 2021.

Authors:
Dmytro Ivanenko, Ph.D., Senior Lecturer, Department of Security of Information Systems and Technologies, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine
E-mail: i8o.dima@gmail.com

Oleksii Pryshchepa, CSD Student, Department of Security of Information Systems and Technologies, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: pryshchepa2020kb51@student.karazin.ua

Evaluation of the efficiency of web-application safety scanners.

Abstract. The level of security of web applications is constantly growing every year, but new ratings of the most common security threats indicate that the problem of ensuring their security is very relevant and constantly changing. Therefore, it is essential to understand the importance of using automatic security scans of web applications and objectively assess their real effectiveness. The paper considers the process of testing web applications for vulnerabilities (and examples of their detection), using free web crawlers (with open-source) by the "black box" method. In this case, scanners interact with applications in the same way as a typical user through a web interface, through the HTTP protocol. The main purpose of the testing is to compare several open-source scanners and determine their effectiveness. It is underlined that it is impossible to evaluate all the indicators of scanners due to the existence of many factors. - Therefore, in the framework of this work, all judgments and conclusions were made only based on an analysis of the received reports of each test scanner. This article provides information about the individual parameters and the number of vulnerabilities found. The testing results indicate that the practice of using only one scanner is not effective, so you need to use

several different solutions at once when testing. This will allow you to get more objective results in terms of detecting both already known security threats and finding new vulnerabilities (with their addition to the final report). The work will be useful to those interested in assessing the security state of modern web applications.

Keywords*:* Web application vulnerabilities; vulnerability scanners; efficient scanners; web application security testing; pentesting.

Рецензент: Вячеслав Калашников, д.ф.-м.н., проф., Технологический университет Монтеррея, Монтеррей, Мексика.
E-mail: kalash@itesm.mx

Поступила: Октябрь 2021.

Авторы:
Дмитрий Иваненко, к.т.н., доцент кафедры Безопасности информационных систем и технологий, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: i8o.dima@gmail.com
Алексей Прищепа, студент факультета компьютерных наук, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: pryshchepa2020kb51@student.karazin.ua

Оценивание эффективности сканеров веб приложений.

Аннотация. Уровень защищенности веб-приложений с каждым годом постоянно растет, однако новые рейтинги наиболее распространенных угроз безопасности, свидетельствуют о том, что проблема обеспечения их защищенности является очень актуальной и постоянно изменяющейся. Поэтому крайне актуально понимать важность использования автоматических сканеров безопасности веб-приложений и уметь объективно оценивать их реальную эффективность. В работе рассмотрен процесс тестирования веб-приложений на наличие уязвимостей (и примеры их обнаружения), с использованием бесплатных веб-сканеров (с открытым кодом) методом «черного ящика». То есть, в данном случае, сканеры взаимодействуют с приложениями так же, как и типичный пользователь через веб-интерфейс, посредством протокола HTTP. Главной целью проведенных тестирований является сравнение нескольких сканеров с открытым кодом и определение их эффективности. Подчеркнуто, что оценить все показатели сканеров невозможно, вследствие существования многих факторов. - Поэтому, в рамках данной работы, все суждения и выводы были сделаны только на основе анализа полученных отчетов каждого тестового сканера. В статье приведены сведения относительно отдельных параметров и количества найденных уязвимостей. Полученные результаты тестирований свидетельствуют о том, что практика использования только одного сканера является мало эффективной, поэтому при тестировании нужно использовать сразу несколько различных решений. Это позволит получить более объективные результаты, в части детектирования, как уже известных угроз безопасности, так и нахождение новых уязвимостей (с их добавлением в конечный отчет). Работа будет полезна тем, кто интересуется вопросами оценки состояния безопасности современных веб-приложений.

Ключевые слова: уязвимости веб приложений; сканеры уязвимостей; эффективность сканеров; тестирование безопасности веб приложений; пентестинг.