JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 176–180
Received: 23 April 2022
Accepted: 5 May 2022

Comparative analysis of the effectiveness of OWASP ZAP, Burp Suite, Nikto and Skipfish in testing the security of web applications

Analiza porównawcza skuteczności narzędzi OWASP ZAP, Burp Suite, Nikto i Skipfish w testowaniu bezpieczeństwa aplikacji internetowych

Aleksandra Bartos, Aleksandra Kondraciuk*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
Application security is one of the key aspects necessary for its proper functioning. Security verification consists primarily in conducting regular penetration tests and checking the vulnerability of the application to various types of attacks. The recommended solution is to use tools dedicated to detecting security holes in applications. Choosing the right tool from among those available on the market can be difficult. This article presents a comparative analysis of the effectiveness of popular application security testing tools in terms of the number of detected vulnerabilities. The analysis was based on the obtained results of scanning two Internet applications containing a number of security vulnerabilities, used to learn ethical hacking.

Keywords: application security; penetration tests; testing tools

Streszczenie
Bezpieczeństwo aplikacji jest jednym z kluczowych aspektów, niezbędnych do jej prawidłowego funkcjonowania. Weryfikacja bezpieczeństwa polega przede wszystkim na przeprowadzaniu regularnych testów penetracyjnych i sprawdzaniu podatności aplikacji na różnego rodzaju ataki. Zalecanym rozwiązaniem jest skorzystanie z narzędzi dedykowanych do wykrywania luk bezpieczeństwa w aplikacjach. Wybór odpowiedniego narzędzia spośród dostępnych na rynku może okazać się trudny. W niniejszym artykule dokonano analizy porównawczej skuteczności popularnych narzędzi testowania bezpieczeństwa aplikacji pod kątem liczby wykrywanych zagrożeń. Analizę oparto o otrzymane wyniki skanowania dwóch aplikacji internetowych, zawierających szereg luk bezpieczeństwa, służących do nauki etycznego hackingu.

Słowa kluczowe: bezpieczeństwo aplikacji; testy penetracyjne; narzędzia do testowania

*Corresponding author
Email address: aleksandra.kondraciuk@pollub.edu.pl (A. Kondraciuk)

1. Wstęp

Testowanie aplikacji internetowych jest ważnym etapem w trakcie ich tworzenia jak i utrzymywania. Rodzaj testów jest zazwyczaj dobierany indywidualnie do potrzeb projektu. Jednymi z najbardziej popularnych typów są testy manualne i automatyczne. Jednak samo działanie aplikacji nie jest jedynym priorytetowym aspektem. Dlatego warto poświęcić więcej uwagi testom bezpieczeństwa.

Wraz z ciągłym rozwojem technologii rośnie liczba incydentów bezpieczeństwa. Stanowi to poważny problem, ponieważ może spowodować utratę wrażliwych danych z aplikacji i ich upowszechnienie. Użytkownik po tego typu zdarzeniu może całkowicie stracić zaufanie do aplikacji i przestać jej używać.

Testy penetracyjne [1] umożliwiają kontrolowane ataki na aplikację internetową, w celu wykrycia jej potencjalnych luk. Tego typu testy muszą być wykonywane cyklicznie z dużą dbałością o dokładność analizy, co może stać się monotonne. Dlatego warto zwrócić uwagę na dostępne rozwiązania do skanowania bezpieczeństwa aplikacji. Ważny jest odpowiedni wybór narzędzia, ponieważ każde z nich, mimo podobnych funkcjonalności oferuje inny zakres wykrywanych zagrożeń czy szybkości wykonywania skanu.

Celem artykułu jest przeprowadzanie testów penetracyjnych za pomocą narzędzi OWASP ZAP, Burp Suite, Nikto i Skipfish oraz ocena ich skuteczności w wykrywaniu zagrożeń bezpieczeństwa aplikacji.

Wybrane do badań narzędzia OWASP ZAP oraz Burp Suite są jednymi z najbardziej popularnych rozwiązań w dziedzinie testów bezpieczeństwa aplikacji internetowych [2]. Oferują zaawansowany interfejs użytkownika z rozbudowanymi funkcjonalnościami, które można dopasować do wymagań użytkownika i skanowanej aplikacji. Oba narzędzia pozwalają na wykrycie najważniejszych luk bezpieczeństwa. Do narzędzi reprezentujących nieco prostszą obsługę jak i mniejszy zakres funkcjonalności należą Nikto [3] oraz Skipfish [4]. Są to rozwiązania, których obsługa opiera się na komendach wpisywanych z poziomu linii poleceń danego systemu.

OWASP ZAP oferuje szeroką gamę funkcjonalności oraz jest powszechnie określany jako jeden z flagowych projektów organizacji OWASP, działającej na rzecz poprawy bezpieczeństwa oprogramowania [5], co pozwala na postawienie następującej tezy badawczej:

OWASP ZAP osiąga najlepsze wyniki spośród badanych narzędzi.

2. Przegląd literatury

Dokonanie wyboru narzędzi i aplikacji internetowych do przeprowadzenia badań wymagało zapoznania się z pozycjami literaturowymi opisującymi kwestie związane z bezpieczeństwem aplikacji internetowych oraz sposobami wykrywania zagrożeń. Zasady działania oraz scenariusze zastosowań testów penetracyjnych zostały dokładnie przedstawione w artykule [1]. Przeprowadzanie tego rodzaju testów wymaga pełnych uprawnień do zarządzania badaną aplikacją ze względu na możliwość wprowadzenia w niej poważnych uszkodzeń. Dlatego też konieczne było lokalne uruchomienie specjalnie do tego przeznaczonych aplikacji. Dostępne tego typu rozwiązania i ich dokumentacja została przedstawiona w pozycjach [6] oraz [7]. Opracowanie wyników skanowania wymaga wiedzy na temat badanego narzędzia oraz umiejętności porównywania otrzymanych wartości, dlatego też pomocna okazała się treść artykułu [3] zawierającego analizę wyników testów przeprowadzonych m.in. za pomocą narzędzi OWASP ZAP i Nikto. Autorzy artykułu dokonali porównania ww. narzędzi poprzez określanie czy dane narzędzie wykryło wszystkie z wyznaczonych 13 luk bezpieczeństwa. Otrzymane w ten sposób wyniki nie pozwoliły na wskazanie jednoznacznie skuteczniejszego narzędzia.

3. Narzędzia

Narzędzia, których skuteczność zostanie zbadana są popularnymi, darmowymi rozwiązaniami w dziedzinie testowania bezpieczeństwa aplikacji internetowych [2]. OWASP ZAP oraz Burp Suite to przykłady narzędzi z graficznym interfejsem użytkownika, które oferują zaawansowane ustawienia trybu skanowania aplikacji, natomiast narzędzia Nikto oraz Skipfish, uruchamiane są z poziomu linii poleceń systemu i nie oferują możliwości wyboru trybu ataku.

3.1. OWASP ZAP

OWASP ZAP (ang. Open-Source Web Application Security Scanner Zed Attack Proxy) to jeden z flagowych projektów organizacji OWASP, będący prostym w użyciu, darmowym skanerem źródłowym, służącym do wyszukiwania luk bezpieczeństwa w aplikacjach internetowych [5]. Wykryte luki narzędzie prezentuje wraz z opisem, liczbą i miejscem wystąpienia oraz propozycją rozwiązania. Dodatkowo generuje szczegółowy raport z przeprowadzonych testów.

3.2. Burp Suite

Burp Suite to produkt firmy PortSwigger służący do przeprowadzania zaawansowanych testów bezpieczeństwa aplikacji. Dostępne są trzy wydania narzędzia: Community, Enterprise i Professional. Rozbudowany skaner bezpieczeństwa pozwala na dopasowanie konfiguracji przeprowadzanych testów przez użytkownika. Dla każdej znalezionej luki zawarta jest informacja m.in. o liczbie oraz miejscu występowania, typie problemu, poziomie niebezpieczeństwa [8]. Po zakończonej weryfikacji użytkownik ma możliwość wygenerowania raportu z przeprowadzonego skanu.

3.3. Nikto

Narzędzie Nikto jest darmowym oprogramowaniem, uruchamianym z poziomu wiersza poleceń, służącym do automatycznego testowania podatności serwerów internetowych [3]. Narzędzie sprawdza występowanie w aplikacji szkodliwych plików, czy problemów z wersjami. Bada także możliwe luki spowodowane błędną konfiguracją zabezpieczeń, prowadzącą do nieautoryzowanego dostępu lub modyfikacji poufnych informacji. Nikto jest zaimplementowany w języku Perl, co umożliwia rozszerzanie narzędzia o dodatkowe funkcjonalności za pomocą wtyczek.

3.4. Skipfish

Skipfish jest narzędziem typu Open-Source do skanowania bezpieczeństwa aplikacji internetowych oraz serwerów. Uruchamiany jest za pomocą wiersza poleceń. Łatwość obsługi i szybkość działania sprawia, że jest przystępnym narzędziem nawet dla początkującego użytkownika. Umożliwia znalezienie luk bezpieczeństwa, tj. możliwość wstrzyknięcia komend systemowych, zapytań SQL, żądań HTTP. Po skończonej weryfikacji generowany jest raport z otrzymanymi wynikami, w którym znalezione błędy kategoryzowane są według stopnia ryzyka [4].

4. Badane aplikacje

Do weryfikacji skuteczności wytypowanych narzędzi zostały wybrane dwie aplikacje internetowe bWAPP [6] oraz Mutillidae [7]. Są one darmowymi projektami stworzonymi na potrzeby nauki testów bezpieczeństwa. Umożliwiają przeprowadzenie szerokiego zakresu testów pasywnych jak i aktywnych [1]. Tego typu testy mogą spowodować realne uszkodzenia w aplikacji, dlatego nie mogłyby zostać przeprowadzone na publicznie dostępnych stronach.

4.1. bWAPP

bWAPP (ang. a buggy Web APPlication) jest darmową aplikacją internetową zawierającą wiele luk bezpieczeństwa [9]. Możliwe jest jej pobranie do użytku lokalnego, jak i w wersji obrazu dysku wirtualnego do instalacji na maszynie wirtualnej. Rozwiązanie jest skierowane do początkującej jak i zaawansowanej grupy odbiorców zajmujących się przeprowadzaniem testów penetracyjnych.

4.2. Mutillidae

OWASP Mutillidae II to darmowa, otwarta, celowo podatna na ataki aplikacja internetowa, będąca dobrym środowiskiem badawczym dla osób wykonujących testy penetracyjne. Mutillidae można zainstalować w systemach Linux i Windows za pomocą LAMP, WAMP i XAMMP. Zagrożenia bezpieczeństwa występujące w aplikacji pochodzą z listy zagrożeń OWASP TOP 10, czyli standardowego dokumentu informacyjnego, prze-

znaczonego dla programistów i osób wykonujących testy bezpieczeństwa aplikacji internetowych. Lista reprezentuje szeroki zakres najważniejszych zagrożeń bezpieczeństwa aplikacji internetowych [4].

5. Kryteria badawcze i klasyfikacja zagrożeń

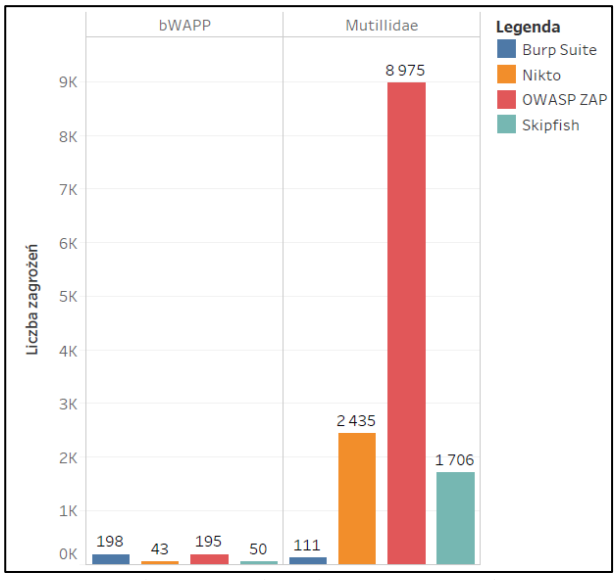
Lista powszechnych zagrożeń bezpieczeństwa aplikacji internetowych stanowi spis kryteriów badawczych, pod kątem których sprawdzano skuteczność badanych narzędzi. Są to luki bezpieczeństwa takie jak podatność na ataki XSS, SQL Injection, braki odpowiednich nagłówków, dostęp do wrażliwych danych, czy elementów struktury aplikacji. Klasyfikacja wykrytego zagrożenia pozwala na łatwy przekaz priorytetu znalezionej luki. Umożliwia to zaplanowanie odpowiednich akcji w celu eliminacji potencjalnych zagrożeń bezpieczeństwa opierając się na skali możliwych incydentów. Zagrożenia z reguły dzielą się na cztery kategorie: wysokie, średnie, niskie i informacyjne.

6. Metoda badań i środowisko badawcze

Eksperyment badawczy rozpoczęto od przygotowania środowisk badawczych oraz zaplanowania kolejności i sposobu przeprowadzania testów. Każde z narzędzi wykonało skan na dwóch uruchomionych lokalnie aplikacjach. Otrzymane wyniki zaprezentowano w tabelach, a następnie przedstawiono w formie wykresów i na ich podstawie dokonano analizy skuteczności badanych narzędzi.

7. Analiza wyników

Otrzymane wyniki badań zostały zestawione w formie wykresu oraz tabel. Wykres zaprezentowany na rysunku 1 przedstawia sumy wystąpień wszystkich wykrytych luk bezpieczeństwa przez poszczególne narzędzia w dwóch aplikacjach internetowych.



Rysunek 1: Wykres sumy wykrytych zagrożeń przez wybrane narzędzia podczas aktywnych skanów aplikacji bWAPP i Mutillidae.

Z wykresu wynika, że najlepiej pod względem liczby wykrytych zagrożeń wypadło narzędzie OWASP ZAP, ponieważ w aplikacji Mutillidae wykryło prawie 9 000 luk bezpieczeństwa, natomiast w aplikacji bWAPP 195, czyli jedynie 3 zagrożenia mniej niż narzędzie Burp Suite.

Podczas analizy wyników konieczne było jednak zweryfikowanie nie tylko sumy wykrytych zagrożeń, ale także ich rodzaju. W tym celu zestawiono wspólne zagrożenia bezpieczeństwa pod względem poziomu oraz liczby wystąpień.

Do przykładowych zagrożeń wykrytych przez każde z badanych narzędzi należą:

- **SQL Injection** - metoda ataku polegająca na wstrzyknięciu złośliwego zapytania do bazy danych, umożliwiająca manipulację jej zawartością.
- **XSS** (ang. Cross-Site Scripting) - atak polegający na wstrzykiwaniu skryptów, umożliwiających np. przejmowanie sesji użytkowników lub na przekierowywanie ich na złośliwe witryny.
- **Path Traversal** - obchodzenie ścieżki to technika ataku pozwalająca na uzyskanie dostępu do zawartości plików, katalogów i poleceń znajdujących się poza głównym katalogiem aplikacji poprzez manipulację adresem URL.
- **Directory Browsing** - możliwość podglądu zawartości katalogów. Pomaga to atakującemu w dostępie do ukrytych skryptów, plików źródłowych i poufnych informacji.
- **Absence of Anti-CSRF Tokens** - podatność aplikacji umożliwiająca zmuszenie użytkownika do wysłania żądania HTTP do miejsca docelowego bez jego wiedzy i przeprowadzenie ataku jako ofiara.
- **Cookie without Flags** - brak ustawienia odpowiednich flag bezpieczeństwa umożliwia manipulację zawartością pliku, przejęcie sesji lub przekierowanie ofiary do złośliwej aplikacji.
- **X-Content-Type-Options Header Missing** - nieodpowiednie ustawienie nagłówka X-Content-Type-Options może doprowadzić do nieprawidłowej interpretacji typu zawartości odpowiedzi z aplikacji.
- **Strict-Transport-Security Header** - brak nagłówka Stric-Transport-Security wymuszającego dostęp do aplikacji tylko za pośrednictwem protokołu HTTPS może powodować podatność witryny na ataki typu man-in-the-middle.

Podczas testów każde z narzędzi wykryło luki bezpieczeństwa, które nie są wspólnymi kryteriami skanowania pozostałych narzędzi. Ich analiza była konieczna do dokonania rzetelnej oceny skuteczności narzędzi.

Kluczowe zagrożenia wykryte przez OWASP ZAP:

- Missing Anti-clicjacking Header - odpowiedź nie obejmuje odpowiednich nagłówków zapewniających ochronę przed atakami typu „ClickJacking”.
- CSP: Wildcard Directive - dyrektywy tj. script-src, style-src, connects-src, form-action zezwalają na źródła wieloznaczne, nie są zdefiniowane lub są zdefiniowane zbyt szeroko.
- RFI (ang. Remote File Inclusion) - technika ataku używana do wykorzystywania mechanizmów „dynamicznego dołączania plików” w aplikacjach internetowych.

Kluczowe zagrożenia wykryte przez Burp Suite:

- XPath injection - metoda ataku polegająca na wstrzyknięciu prostych zapytań XPath (ang. XML Path Language) do dokumentów HTML i odczyt wrażliwych danych.
- LDAP injection - atak polegający na wstrzyknięciu danych przy pomocu protokołu LDAP (ang. Lightweight Directory Access Protocol) w celu uzyskania dostępu do poufnych danych.
- Server-side template injection - metoda ataku polegająca na wstrzyknięciu szablonu po stronie serwera w celu manipulowania działaniem aplikacji.

Kluczowe zagrożenia wykryte przez Nikto

- Component is in a non-updated version - przestarzałe komponenty używane przez aplikację są podatne na różnego rodzaju ataki.
- Allowed phpinfo() to be run, which gives detailed system information - funkcja phpinfo() wyświetla wrażliwe informacje m.in. o wersji PHP, serwerze, ścieżkach, lokalnych opcjach konfiguracyjnych, nagłówkach HTTP oraz licencji.
- HTTP TRACE method is active, suggesting the host is vulnerable to XST - atak XST (ang. Cross-Site Tracing) obejmuje użycie (XSS) oraz metod TRACE lub TRACK HTTP.

Kluczowe zagrożenia wykryte przez Skipfish

- Shell injection vector to metoda ataku polegająca na wstrzyknięciu dowolnych poleceń systemu operacyjnego za pomocą podatnych aplikacji.
- Incorrect caching directives - nieprawidłowo ustawione dyrektywy nagłówka HTTP Cache-Control. Odpowiedzialne są za kontrolowanie buforowania w aplikacjach i współdzielenia pamięci podręcznej.
- HTML form with no apparent XSRF protection - brak ochrony przed atakami typu XSRF (ang. Cross-site request forgery) w fotmularzach dostępnych w aplikacji. Podatność ma na celu wykonanie przez ofiarę niepożądanych akcji, które mogą doprowadzić do kradzieży danych.

W tabelach 2 i 3 przedstawiono wspólnie wykryte zagrożenia przez poszczególne narzędzia w aplikacjach Mutillidae oraz bWAPP. Znak „-” w komórkach oznacza, że dane zagrożenie nie jest wykrywane przez narzędzie. Natomiast wartość „0” oznacza, że narzędzie skanuje aplikację pod tym kątem, ale zagrożenie nie zostało wykryte.

Tabela 2: Liczba wystąpień poszczególnych zagrożeń wykrytych podczas skanowania aplikacji Mutillidae

Zagrożenie	Aplikacja Mutillidae							
	OWASP ZAP		Burp Suite		Nikto		Skipfish	
	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień
SQL Injection	wysoki	6	wysoki	8	wysoki	1	wysoki	96
Client-side SQL injection (DOM-based)	wysoki	13	wysoki	0	-	-	-	-
Cross-site scripting (DOM-based)	wysoki	3	wysoki	2	wysoki	0	wysoki	30
Cross Site Scripting (Persistent)	wysoki	3	wysoki	0	-	-	-	-
Cross Site Scripting (Reflected)	wysoki	100	wysoki	30	-	-	-	-
Path Traversal	wysoki	35	wysoki	3	-	-	-	-
Remote OS Command Injection	wysoki	3	wysoki	0	-	-	-	-
Vulnerable JS Library	średni	4	średni	0	-	-	-	-
Directory Browsing	średni	66	średni	0	średni	0	średni	398
Client-side HTTP parameter pollution	średni	76	niski	4	-	-	-	-
Absence of Anti-CSRF Tokens	niski	1570	niski	0	-	-	-	-
Cookie no HttpOnly Flag	niski	227	niski	1	niski	2	-	-
Cookie Without Secure Flag	niski	214	niski	0	niski	2	-	-
Cookie without SameSite Attribute	niski	5	niski	0	-	-	-	-
Loosely Scoped Cookie	niski	0	niski	0	-	-	-	-
Incomplete/No Cache-control Header Set	niski	1295	info	1	-	-	-	-
X-Content-Type-Options Header Missing	niski	2120	niski	0	-	-	-	-
X-Frame-Options Header Not Set	niski	0	info	2	niski	1	-	-
Strict-Transport-Security Header	niski	0	niski	1	niski	1	-	-
Retrieved x-powered-by header	niski	0	-	-	niski	1	-	-
Open redirection (DOM-based)	niski	0	niski	2	-	-	-	-
Cross-domain script include	niski	0	info	0	-	-	-	-
Secure Pages Iclude Mixed Content	niski	0	info	0	-	-	-	-
Charset Mismatch	info	139	info	0	-	-	niski	209
Private IP addresses disclosed	niski	30	info	2	-	-	-	-
File upload	info	0	info	1	-	-	-	-
Email addresses disclosed	info	0	info	1	-	-	-	-

Tabela 3: Liczba wystąpień poszczególnych zagrożeń wykrytych podczas skanowania aplikacji bWAPP

Zagrożenie	Aplikacja bWAPP							
	OWASP ZAP		Burp Suite		Nikto		Skipfish	
	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień	Poziom zagrożenia	Liczba wystąpień
SQL Injection	wysoki	0	wysoki	6	wysoki	0	wysoki	0
Client-side SQL injection (DOM-based)	wysoki	0	wysoki	0	-	-	-	-
Cross-site scripting (DOM-based)	wysoki	0	wysoki	0	wysoki	1	wysoki	0

Cross Site Scripting (Persistent)	wysoki	0	wysoki	5	-	-	-	-
Cross Site Scripting (Reflected)	wysoki	0	wysoki	14	-	-	-	-
Path Traversal	wysoki	0	wysoki	0	-	-	-	-
Remote OS Command Injection	wysoki	0	wysoki	4	-	-	-	-
Vulnerable JS Library	średni	1	średni	1	-	-	-	-
Directory Browsing	średni	4	średni	0	średni	7	średni	16
Client-side HTTP parameter pollution	średni	7	niski	0	-	-	-	-
Absence of Anti-CSRF Tokens	niski	5	niski	9	-	-	-	-
Cookie no HttpOnly Flag	niski	5	niski	2	niski	7	-	-
Cookie Without Secure Flag	niski	0	niski	0	niski	7	-	-
Cookie without SameSite Attribute	niski	5	niski	0	-	-	-	-
Loosely Scoped Cookie	niski	0	niski	0	-	-	-	-
Incomplete/No Cache-control Header Set	niski	0	info	1	-	-	-	-
X-Content-Type-Options Header Missing	niski	86	niski	0	-	-	-	-
X-Frame-Options Header Not Set	niski	0	info	2	niski	1	-	-
Strict-Transport-Security Header	niski	0	niski	1	niski	1	-	-
Retrieved x-powered-by header	niski	0	-	-	niski	1	-	-
Open redirection (DOM-based)	niski	0	niski	0	-	-	-	-
Cross-domain script include	niski	0	info	0	-	-	-	-
Secure Pages Iclude Mixed Content	niski	0	info	0	-	-	-	-
Charset Mismatch	info	0	info	0	-	-	niski	4
Private IP addresses disclosed	niski	0	info	0	-	-	-	-
File upload	info	0	info	0	-	-	-	-
Email addresses disclosed	info	0	info	0	-	-	-	-

8. Wnioski

Badania wytypowanych narzędzi pozwoliły na ocenę ich skuteczności. Określono ją na podstawie liczby wykrytych luk bezpieczeństwa, ich rodzaju, liczby wystąpień oraz poziomu zagrożenia jakie stanowią.

Potwierdzona została teza, że najskuteczniejszym narzędziem spośród grupy analizowanych, okazało się narzędzie OWASP ZAP, ze względu na dużą liczbę wykrytych zagrożeń wysokiego i średniego poziomu w każdej z aplikacji.

Nieco gorsze, ale znaczące wyniki osiągnęło narzędzie Burp Suite, ponieważ wykryło znaczną liczbę zagrożeń wysokiego poziomu w obu aplikacjach.

Narzędzia Nikto oraz Skipfish wykryły znacząco mniej zagrożeń niż OWASP ZAP i Burp Suite.

Poza wspólnie wykrytymi zagrożeniami narzędzia OWASP ZAP oraz Burp Suite wyłoniły także inne istotne podatności aplikacji. Luki wykryte przez Nikto i Skipfish nie stanowią przykładu poważnych zagrożeń.

Porównując otrzymane wyniki do wniosków z artykułu [3] można ocenić wybrany sposób badań skuteczności narzędzi za właściwy. Dzięki skupieniu się na pełnym zakresie otrzymanych wyników, a nie na ograniczonej liczbie kryteriów możliwe było wytypowanie najskuteczniejszego z wybranych narzędzi.

Podsumowując przeprowadzone badania narzędzie OWASP ZAP wydaje się być najlepszym wyborem, jednak należy pamiętać, że wybór narzędzia powinien być podyktowany potrzebami skanowanej aplikacji, jej wielkością i zachodzącymi procesami. Najlepszym rozwiązaniem zapewniającym bezpieczeństwo aplikacji jest jej systematyczne skanowanie za pomocą kilku narzędzi w odpowiedniej kolejności, a nie poleganie na wynikach tylko jednego z nich.

Literatura

[1] D.D. Bertoglio, A.F. Zorzo, Overview and open issues on penetration test, Journal of the Brazilian Computer Society 23(2) (2017) 1-2.

[2] Spis narzędzi służących do skanowania bezpieczeństwa polecanych przez OWASP, https://owasp.org/www-community/Vulnerability_Scanning_Tools, [02.2022].

[3] R. Devi, M. Kumar, Testing for Security Weakness of Web Applications using Ethical Hacking, 2020 4th International Conference on Trends in Electronics and Informatics 354 (2020) 358-360.

[4] D. Sagar, S. Kukreja, J. Brahma, S. Tyagi, P. Jain, Studying Open Source Vulnerability Scanners For Vulnerabilities In Web Applications, IIOABJ 9(2) (2018) 43-49.

[5] B. Mburano, W. Si, Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark, 2018 26th International Conference on Systems Engineering (2018) 1-2.

[6] Dokumentacja i kod źródłowy aplikacji bWAPP, https://sourceforge.net/projects/bwapp/files/bWAPP/, [03.2022].

[7] Dokumentacja i kod źródłowy aplikacji Mutillidae, https://github.com/webpwnized/mutillidae, [03.2022].

[8] M. El, E. McMahon, S. Samtani, M. Patton, H. Chen, Benchmarking vulnerability scanners: An experiment on SCADA devices and scientific instruments, IEEE International Conference on Intelligence and Security Informatics (ISI) (2017) 83-85.

[9] S. Tyagi, K. Kumar, Evaluation of Static Web Vulnerability Analysis Tools, 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (2018) 1-3.