JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 16 (2020) 261–268
Received: 20 June 2020
Accepted: 30 June 2020

Analysis of security CMS platforms by vulnerability scanners

Badanie bezpieczeństwa wybranych platform CMS za pomocą skanerów podatności

Patryk Zamościński*, Grzegorz Kozieł

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

Abstract
Subject of security the most popular CMS platforms has been undertaken in the following thesis. There were introduced fundamental informations about subject CMS platforms and vulnerability scanners utilised to research. For research purposes Wordpress and Joomla websites were created and investigated for security by vulnerability scanners OWASP ZAP, Vega, Detectify and Skipfish. Results were grouped by some criteria: vulnerabilities by category and vulnerabilities by threat level. Obtained results were examined in two ways: analysis of residual results, for each website scanning and analysis of aggregated results from all scanners. After that, conclusions about CMS platforms security have been drawn.

Keywords: CMS; security; vulnerability scanner

Streszczenie
W niniejszej pracy został podjęty temat bezpieczeństwa najbardziej popularnych platform CMS. Przedstawiono podstawowe informacje o badanych platformach CMS oraz o skanerach podatności użytych w badaniach. W celach badawczych zostały utworzone witryny Wordpress i Joomla, które następnie przebadano pod kątem bezpieczeństwa za pomocą skanerów OWASP ZAP, Vega, Detectify oraz Skipfish. Wyniki zostały pogrupowane według kryteriów: podatności w poszczególnych kategoriach oraz podatności według poziomu zagrożenia. Wyniki były rozpatrywane na dwa sposoby: analiza wyników szczątkowych, oraz analiza zbiorcza zsumowanych wyników ze wszystkich skanerów. Na koniec zostały wyciągnięte wnioski w odniesieniu do obu platform.

Słowa kluczowe: CMS; bezpieczeństwo; skaner podatności

*Corresponding author
Email address: patryk.zamoscinski@pollub.edu.pl (P. Zamościński), g.koziel@pollub.pl (G. Kozieł)

1. Wstęp

W ostatnich dekadach nastąpił gwałtowny rozwój Internetu. Co się z tym wiąże, duża część życia społecznego została przeniesiona do sieci. W dzisiejszym świecie bez wychodzenia z domu można zrobić wiele rzeczy, poczynając od rozrywki i realizacji swoich pasji (fora internetowe skupiające entuzjastów, blogi), poprzez robienie zakupów a kończąc na załatwianiu spraw urzędowych czy pracy zdalnej.

Zagadnienia związane z bezpieczeństwem w sieci mają kluczowe znaczenie w dzisiejszym świecie. Dane, które podajemy podczas rejestracji w różnego rodzaju stronach internetowych przedstawiają dużą wartość dla cyberprzestępców. Może się wydawać, że jedyne miejsca, gdzie cyberbezpieczeństwo jest szczególnie ważne to aplikacje bankowe i poczta e-mail. Jednak każda strona, która posiada jakieś informacje o swoich użytkownikach jest potencjalnym celem atakującego.

W dzisiejszych czasach systemy CMS są bardzo popularne ze względu na to, że nie wymagają od użytkownika umiejętności programistycznych przy tworzeniu strony WWW. Statystyki pokazują, że aż 46% aplikacji internetowych posiada podatności, które mają krytyczne znaczenie dla bezpieczeństwa [1]. Dlatego autor tej pracy postanowił sobie za cel zbadanie dwóch najpopularniejszych platform CMS (WordPress i Joomla) dostępnych na rynku, opisanie wykrytych luk w zabezpieczeniach oraz wyłonienie bezpieczniejszej platformy.

2. Przegląd literatury

W pracy [2] autorzy porównali pod kątem bezpieczeństwa platformy Drupal i Joomla. Proces badawczy składał się z 4 etapów: instalacja i konfiguracja witryn, przeprowadzenie prostych, zautomatyzowanych skanów, analiza plików źródłowych oraz wykonanie ukierunkowanych ataków. Wyniki przedstawiono w postaci tabeli zawierającej kategorie bezpieczeństwa z wyszczególnionymi kryteriami, oraz oceną tych kryteriów.

Szersze podejście do badania bezpieczeństwa współczesnych stron internetowych można odnaleźć w pracy [1]. Autorzy użyli skanera Acunetix Online do przebadania losowo wybranych 10 000 stron internetowych. Z przeprowadzonych badań wysnuto wniosek, że w 35% stron internetowych wykryto co najmniej jedną podatność o wysokim stopniu zagrożenia.

W trakcie przeszukiwania zbioru pozycji literaturowych odnaleziono prace traktujące o bezpieczeństwie systemów typu CMS [3, 4, 5, 6]. Okazało się, że praca Comparative Analysis Of Web Security In Open Source Content Management System [3] porusza tą samą tematykę co niniejsza praca, z tym że autorzy porównywali jeszcze platformę Drupal.

Po przeprowadzeniu badań literaturowych stwierdzono, że co prawda poruszony temat był badany, jednakże nie przeprowadzono dogłębnych badań, które naświetliłyby luki w zabezpieczeniach badanych systemów CMS i dałyby wyczerpującą odpowiedź na pytania o bezpieczeństwo tych systemów.

3. Metodyka badawcza

Badania zostały przeprowadzone w następujący sposób: na serwerze lokalnym zostały utworzone dwie witryny, jedna stworzona za pomocą platformy WordPress a druga za pomocą Joomla. Na potrzeby badania skanerem Detectify należało stworzyć witryny na zewnętrznym hostingu, identyczne, jak te wykorzystywane w pozostałych badaniach. Obie witryny zawierają podstawowe elementy platform CMS takie jak: podstrony, posty, formularze logowania, pole „szukaj", kategorie oraz tagi. Następnie tak przygotowane witryny zostały poddane skanowaniu pod kątem bezpieczeństwa za pomocą wybranych skanerów bezpieczeństwa.

Wyniki zostały pogrupowane według kryteriów: liczba podatności w poszczególnych kategoriach oraz liczba podatności dla każdego poziomu zagrożenia. Otrzymane wyniki były rozpatrywane na dwa sposoby: analiza wyników szczątkowych, po każdym skanowaniu witryny, oraz w podsumowaniu analiza zbiorcza zsumowanych wyników otrzymanych ze wszystkich skanerów.

4. Badane platformy CMS

4.1. WordPress

Jak pokazują badania, WordPress jest aktualnie najbardziej popularnym systemem CMS na świecie z udziałami w rynku rzędu 63,3% [6]. Aż 36,4% wszystkich stron internetowych w jakiś sposób używa WordPress [6]. Jeśli chodzi o platformy CMS, to WordPress jest niekwestionowanym liderem na rynku. udostępnia szeroki wachlarz możliwości tworzenia stron internetowych, od zwykłych witryn, aplikacji internetowych, po rozbudowane serwisy, a nawet sklepy internetowe.

Swoją popularność WordPress zawdzięcza głównie bardzo dużej i aktywnie działającej społeczności. Z jednej strony społeczność pomaga początkującym użytkownikom służąc poradami i wychwytuje różnego rodzaju błędy platformy. Z drugiej strony społeczność to twórcy, którzy są odpowiedzialni za tworzenie rozszerzeń do WordPress, takich jak wtyczki lub motywy. To właśnie wtyczki są największą zaletą tego CMS. Ogromna (największa spośród wszystkich platform CMS) liczba rozszerzeń pozwala na modyfikację każdego aspektu witryny internetowej [4]. Liczba tylko bezpłatnych wtyczek szacowana jest na ponad 10000 i cały czas tworzone są nowe.

WordPress ma bardzo dużo różnego rodzaju funkcjonalności, dodatkowo rozszerzanych przez wtyczki, jednak warto przedstawić najważniejsze z nich [7,8]:

- możliwość definiowania własnych taksonomii, czyli kategorii i tagów,
- ochrona przed spamem,
- zarządzanie użytkownikami,
- możliwość zabezpieczenia hasłem konkretnego postu lub strony,
- łatwy import zasobów,
- inteligentne formatowanie tekstu,
- zarządzanie przepływem pracy,
- dostęp do panelu administracyjnego z aplikacji mobilnej,
- wyszukiwarka,
- przyjazne dla użytkownika adresy URL,
- przetłumaczony na ponad 180 języków,
- interfejs XML-RPC umożliwiający wymianę danych z innymi systemami.

4.2. Joomla

Drugim co do popularności systemem CMS jest Joomla. Zgodnie z raportem firmy w3techs 2.4% wszystkich współczesnych stron internetowych używa tej platformy, natomiast udziały w rynku tego CMS są na poziomie 4.1% [6].

Na popularność tej platformy składa się kilka czynników, jednym z nich jest mnogość zastosowań. Joomla na początku był używany głównie do tworzenia witryn publicystycznych, takich jak blogi czy portale informacyjne, jednak wraz z rozwojem platformy zostały dodane funkcjonalności umożliwiające tworzenie rozbudowanych portali, stron internetowych, korporacyjnych sieci intranet lub extranet a także serwisów eCommerce. Poza tym istnieje możliwość używani przez deweloperów pełnoprawnego szkieletu projektowego Joomla Framework, co jeszcze poszerza zakres możliwości wykorzystania tej platformy. Joomla cechuje się elastycznością, liczne rozszerzenia i szablony pozwalają dostosować witrynę do określonych wymagań.

O wysokiej pozycji Joomla na tle innych platform CMS decyduje duża liczba udostępnianych funkcjonalności [7,9]:

- zarządzanie użytkownikami i ich uprawnieniami,
- wyszukiwarka,
- elastyczność w tworzeniu różnego rodzaju witryn,
- duża liczba rozszerzeń (ponad 8000),
- zarządzanie menu,
- zarządzanie kategoriami,
- zarządzanie poszczególnymi modułami witryny,
- zarządzanie ustawieniami pamięci podręcznej cache,
- zarządzanie szablonami,
- zarządzanie banerami i reklamami,
- zaawansowany edytor tekstowy,
- rozszerzona dokumentacja dla developerów,
- tłumaczenie na 70 języków.

5. Wykorzystane skanery podatności

5.1. OWASP ZAP

OWASP Zed Attack Proxy (OWASP ZAP) jest najczęściej używanym przez testerów skanerem podatności. Ze względu na prostotę w instalacji i użytkowaniu ZAP może być używany zarówno przez początkujących, jak i zaawansowanych testerów. OWASP ZAP ma postać aplikacji desktopowej dostępnej na platformach

Windows, Linux oraz macOS. Jest to oprogramowanie typu open source rozwijane w przez organizację OWASP.

OWASP ZAP posiada funkcjonalności przydatne dla każdego profesjonalnego testera bezpieczeństwa:

- lokalne proxy - pozwala przechwytywać i modyfikować przesyłane żądania,
- pająk – narzędzie służące do przeszukiwania witryny i znajdowania powiązań,
- pająk AJAX – do przeszukiwania witryny wykorzystuje Crawjax, aby odkryć wszystkie powiązania stron [10],
- automatyczne skanowanie – w wersji pasywnej (analiza wysłanych i odebranych żądań) oraz aktywnej (preparowanie i wysyłanie żądań do witryny),
- fuzzer – umożliwia wysyłanie do celu dużej ilości niepoprawnych i niespodziewanych danych. Użytkownik wybiera jakie dane zostaną wysłane, można też definiować własne zestawy danych [10].

5.2. Vega

Vega jest to projekt open source firmy Subgraph zajmującej się cyberbezpieczeństwem. Vega jest aplikacją desktopową, posiada GUI stworzone w języku Java oraz działa na systemach Windows, Linux oraz macOS [11].

Vega posiada funkcjonalności pozwalające na kompleksowe przebadanie danej witryny:

- rozszerzalność – Vega udostępnia API do tworzenia własnych scenariuszy ataku [11],
- zaawansowany automatyczny skaner,
- proxy dedykowane do obserwacji i ingerencji w przesyłane żądania [11],
- skaner proxy – umożliwia uruchomienie modułów ataku podczas gdy użytkownik przegląda stronę za pomocą proxy. Umożliwia to wykonywanie połowicznie automatycznych testów bezpieczeństwa, aby zapewnić możliwie jak największe pokrycie kodu [11].

5.3. Detectify

Detectify został założony w 2013 roku. Jest to projekt komercyjny, udostępniający 14-dniowy okres próbny. Detectify jako jedyny skaner z zestawienia jest opartą na SaaS (Software as a Service) aplikacją internetową [12]. Według twórców ten skaner testuje strony internetowe pod kątem obecności ponad 1500 podatności, w tym tych z zestawienia OWASP Top 10.

Jako jedyny z używanych skanerów Detectify stosuje weryfikację, czy domena należy do użytkownika. Aby potwierdzić własność witryny, użytkownik musi pobrać plik tekstowy i wysłać go na serwer.

Detectify udostępnia swoim użytkownikom wiele ciekawych funkcjonalności, które pomagają monitorować stan bezpieczeństwa strony internetowej:

- skaner sprawdzający strony na obecność ponad 1500 podatności [12],
- aktualizowane testy bezpieczeństwa tworzone przez etycznych hakerów [12],
- monitorowanie zasobów – pozwala na monitorowanie witryny pod kątem bezpieczeństwa i - jeśli zostanie wykryta podejrzana działalność - użytkownik jest o tym powiadamiany,
- skanowanie cykliczne,
- rozbudowane statystyki.

5.4. Skipfish

Skipfish to skaner podatności open source stworzony przez Michała Zalewskiego a udostępniany przez Google. Skipfish nie ma interfejsu użytkownika, jest przeznaczony do pracy w środowisku Linux, Windows oraz macOS, jako aplikacja konsolowa [13]. Twórca przyznaje, że Skipfish nie spełnia wielu wymagań stawianych przed skanerami podatności, jednak to narzędzie rozwiązuje pewne niedogodności spotykane w innych skanerach i może być pomocne w naprawdę wielu przypadkach [13].

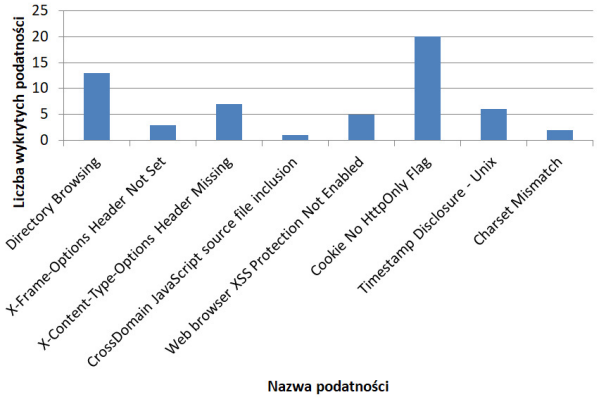
Mimo że Skipfish nie jest w pełni kompleksowym narzędziem to jednak zawiera wiele funkcjonalności, które wyróżniają ten skaner na tle innych:

- wysoka wydajność – wysyłanie ponad 500 żądań na sekundę do celów dostępnych przez Internet, ponad 2000 do celów dostępnych przez LAN/MAN oraz ponad 7000 do celów dostępnych lokalnie sprawiają, że Skipfish jest bardzo wydajnym narzędziem [13],
- multipleksowanie, działanie asynchroniczne oraz model przetwarzania danych sprawiają, że Skipfish zużywa relatywnie mało zasobów [13],
- łatwość w użyciu,
- dobrze zaprojektowane testy bezpieczeństwa [13].

6. Wyniki badań

6.1. Wyniki audytu OWASP ZAP

Na rysunku 6.1 przedstawiono podatności wykryte podczas skanowania witryny WordPress wraz z liczbą adresów URL zawierających daną podatność.

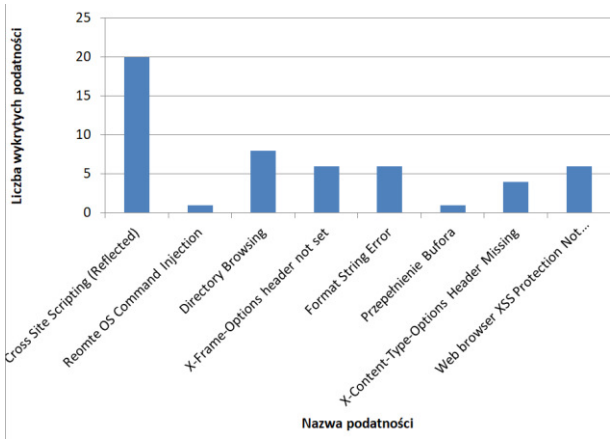


Rysunek 1: Podatności wykryte podczas skanowania witryny WordPress za pomocą skanera OWASP ZAP

Najliczniej występującą podatnością wykrytą podczas skanowania jest Cookie No HttpOnly Flag. Ciasteczka tej strony mogą zostać przejęte i przeniesione na inną stronę. Stwarza to możliwość przejęcia sesji użytkownika.

Najgroźniejszą podatnością wykrytą podczas skanowania jest Directory Browsing. Atakujący może dowolnie przeglądać listę katalogów na serwerze, co umożliwia poznanie struktury aplikacji. Można w taki sposób uzyskać dostęp do ukrytych skryptów, kopii zapasowych, które pomogą zaplanować dalsze etapy ataku.

Na rys. 2 przedstawiono wyniki skanowania witryny Joomla za pomocą skanera OWASP ZAP.

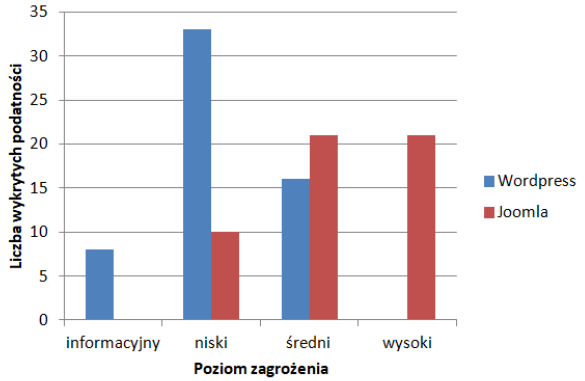


Rysunek 2: Podatności wykryte podczas skanowania witryny Joomla za pomocą skanera OWASP ZAP

W tym przypadku najgroźniejszą oraz najczęściej występującą podatnością jest Cross Site Scripting (Reflected).

Kolejnym słabym punktem witryny jest wrażliwość na atak Remote OS Command Injection. Atak ten umożliwia nieautoryzowane wykonanie poleceń systemowych poprzez brak filtrowania danych wejściowych używanych do tworzenia poleceń systemowych. Wstrzykiwanie poleceń systemowych jest również możliwe w przypadku niepoprawnego wywoływania programów zewnętrznych. Podatność została odnaleziona w podstronie *about*.

Na rys. 3 przedstawiono liczbę wykrytych podatności według poziomu zagrożenia.



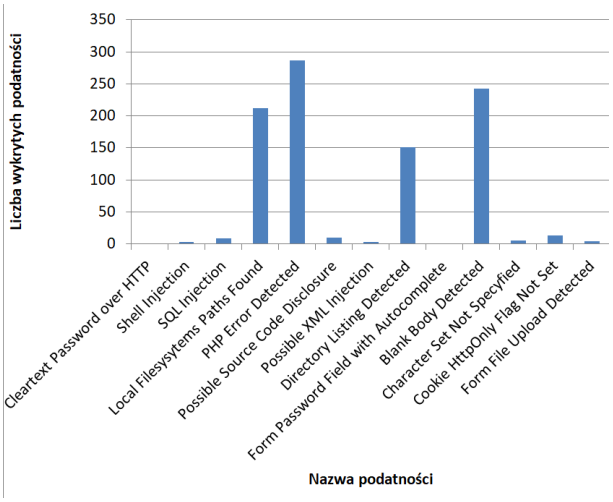
Rysunek 3: Liczba podatności według poziomu zagrożenia wykryta podczas skanowania programem OWASP ZAP

Mimo że witryna WordPress miała wykrytych więcej podatności, to jednak większość z nich stanowią te o niskim stopniu zagrożenia. Podczas skanowania nie została wykryta żadna podatność w wysokim stopniu wpływająca na bezpieczeństwo witryny.

W przypadku witryny Joomla sytuacja jest odwrotna. Znaczna większość wykrytych podatności ma przypisany poziom średni lub wysoki. Podatności o niskim stopniu zagrożenia zostało wykrytych niewiele, a tych o najniższym poziomie nie znaleziono wcale. Oznacza to, że mimo tego, że witryna WordPress miała więcej luk bezpieczeństwa, to jednak witryna Joomla jest relatywnie gorzej zabezpieczona.

6.2. Wyniki audytu Vega

Poniżej na rysunku 4 przedstawiono podatności wykryte w witrynie WordPress.



Rysunek 4: Podatności wykryte podczas skanowania witryny WordPress za pomocą skanera Vega

Najczęściej występującą podatnością wykrytą podczas skanowania jest PHP Error Detected. Gdy podczas otwierania strony zostanie wykryty błąd, zamiast docelowej strony zostaje wyświetlona wygenerowana automatycznie strona, która zawiera szczegółowe informacje dotyczące występującego błędu. Atakujący poprzez odpowiednią preparację żądań i analizę wywołanych błędów PHP może poznać schemat działania aplikacji.

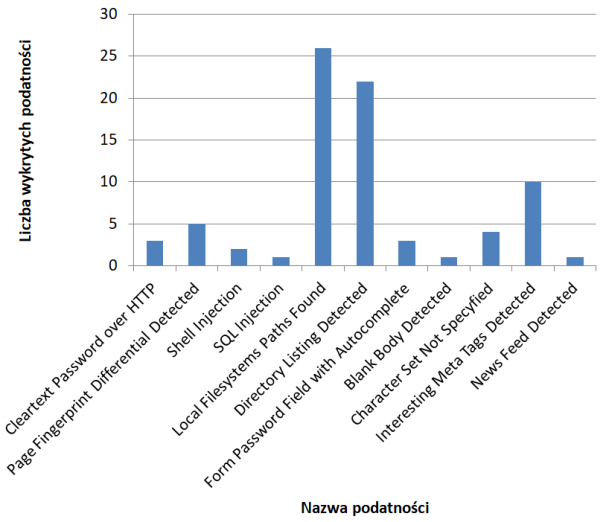
Najgroźniejszymi wykrytymi podatnościami są SQL Injection oraz Shell Injection.

Po przeskanowaniu witryny WordPress przeprowadzono audyt dla witryny Joomla. Otrzymane wyniki zostały przedstawione na rysunku 5.

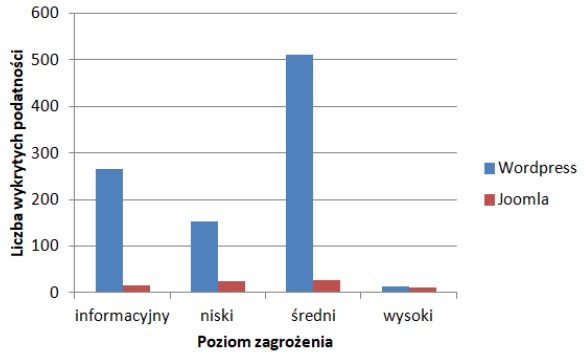
Najliczniej występujące to: Local Filesystems Paths Found oraz Directory Listing Detected.

Pierwsza z wymienionych podatności polega na wyświetlaniu ścieżki bezwzględnej pliku, najczęściej w odpowiedzi na błąd. Jest to dosyć poważna podatność, umożliwiająca atakującemu pozyskanie informacji o środowisku serwerowym. Znajomość układu plików na serwerze może zwiększyć szansę powodzenia ataku typu „blind attack”.

W kolejnym etapie badań liczbę wykrytych podatności podzielono według stopnia niebezpieczeństwa nadanego przez skaner. Wyniki przedstawiono na rysunku 6.



Rysunek 5: Podatności wykryte podczas skanowania witryny Joomla za pomocą skanera Vega



Rysunek 6: Liczba podatności według poziomu zagrożenia wykryta podczas skanowania programem Vega

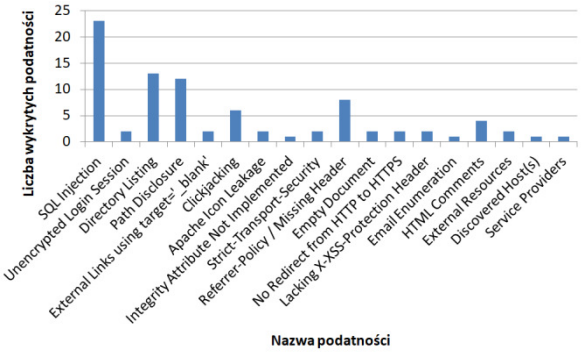
Widoczna jest duża różnica pomiędzy wynikami dla WordPress i Joomla. Podczas skanowania witryny WordPress znaleziono kilka typów podatności, które zawierają znaczącą większość wszystkich wykrytych podatności. Niemal wszystkie z tych podatności odnoszą się do plików zasobów. Spośród tych 4 typów podatności, 3 zostały również odkryte podczas skanowania witryny Joomla, jednak tam liczba wykrytych podatności dla każdego typu wynosiła ok. 20. Prawdą jest, że WordPress zawiera więcej zasobów niż Joomla, jednak główną przyczyną tak dużej rozbieżności w liczbie wykrytych podatności jest gorsze zabezpieczenie zasobów witryny WordPress. W każdej kategorii więcej podatności zostało odnalezionych w witrynie WordPress.

6.3. Wyniki audytu Detectify

Na rysunku 7 przedstawiono podatności wykryte podczas skanowania witryny WordPress wraz z liczbą adresów URL zawierających daną podatność.

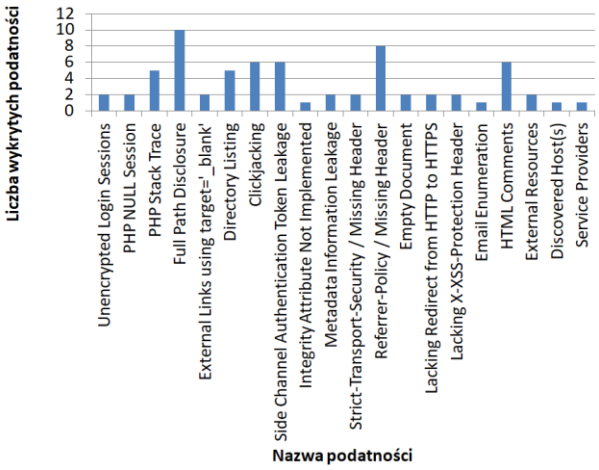
Najczęściej występującą i najgroźniejszą podatnością wykrytą podczas skanowania jest SQL Injection.

Bardzo niepokojącą podatnością znalezioną podczas przeprowadzania badań jest Unencrypted Login Session. Oznacza to, że jeżeli atakującemu uda się przechwycić ruch internetowy, może on odczytać dane logowania, ponieważ są przesyłane w postaci tekstu jawnego. Ta podatność jest tym groźniejsza, że została znaleziona w formularzu logowania do panelu administracyjnego. W witrynie WordPress został zaimplementowany protokół HTTPS jednak jest on nieużywany, jak wskazuje podatność Lacking redirect from HTTP to HTTPS



Rysunek 7: Podatności wykryte podczas skanowania witryny WordPress za pomocą skanera Detectify

Następnie badaniu została poddana witryna Joomla. Wyniki zostały przedstawione na rysunku 8.



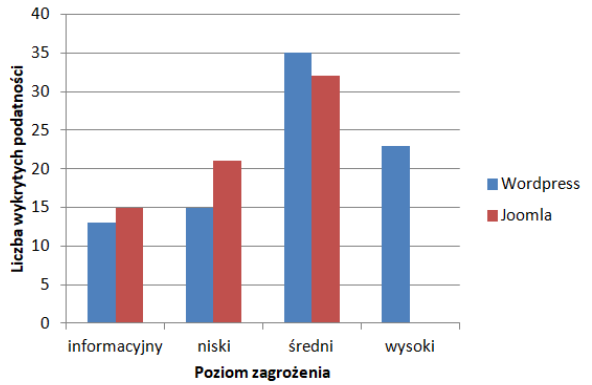
Rysunek 8: Podatności wykryte podczas skanowania witryny Joomla za pomocą skanera Detectify

Najczęściej wykrywaną podatnością podczas skanowania okazało się Full Path Disclosure. Oznacza to, że istnieje możliwość podejrzenia pełnej ścieżki niektórych plików na serwerze. Może to pomóc atakującemu zorientować się w strukturze katalogów i plików, może stanowić również bazę do innych ataków, takich jak Local File Inclusion, zwłaszcza w połączeniu z również wykrytą podatnością PHP Null Session.

Kolejną, często występującą podatnością jest Side Channel Authentication Token Leakage. W niektórych starszych przeglądarkach możliwe jest zbieranie informacji o bieżących działaniach kryptograficznych poprzez wykorzystanie elementów strony internetowej jako bocznych kanałów. Następnie zdobyte informacje można wykorzystać podczas inżynierii wstecznej do odwrócenia tokenów bezpieczeństwa. Tym sposo-

bem możliwe jest wyodrębnienie danych użytkownika takich jak login lub adres e-mail.

Kolejnym krokiem było przedstawienie znalezionych podatności dla obu witryn z poziomem zagrożenia przypisanym przez skaner, tak jak to pokazano na rys. 9.

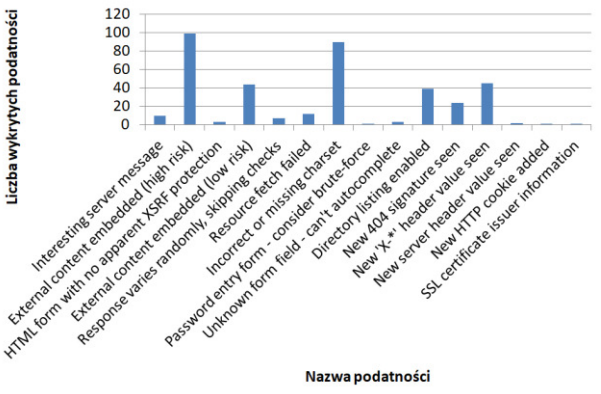


Rysunek 9: Liczba podatności według poziomu zagrożenia wykryta podczas skanowania programem Detectify

Porównując otrzymane wyniki można zauważyć, że liczba wykrytych podatności poziomu informacyjnego, niskiego i średniego dla obu witryn jest zbliżona. Widać przewagę WordPress w zagrożeniach poziomu informacyjnego i niskiego objawiającą się mniejszą liczbą wykrytych podatności, natomiast biorąc pod uwagę zagrożenia o średnim poziomie zagrożenia przewagę ma Joomla. Jednakże w witrynie Joomla nie wykryto żadnych podatności o wysokim stopniu zagrożenia, w przeciwieństwie do WordPress. W WordPress wykryto 23 podatności o wysokim stopniu niebezpieczeństwa dla witryny, są to podatności typu SQL Injection. Z powyższego porównania można wyciągnąć wniosek, że w badaniu skanerem Detectify witryna WordPress okazała się znacznie gorzej zabezpieczona niż witryna Joomla.

6.4. Wyniki audytu Skipfish

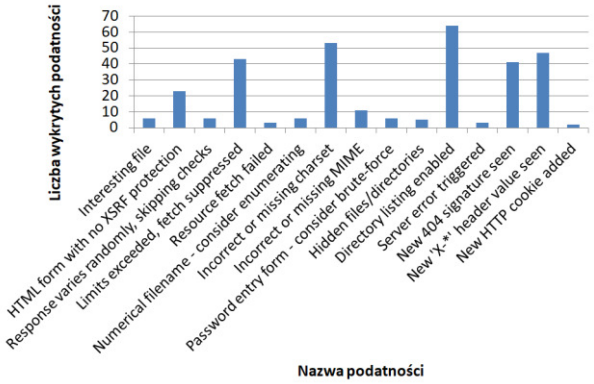
Na rys. 10 przedstawiono podatności wykryte w witrynie WordPress, pogrupowane według kategorii podatności.



Rysunek 10: Podatności wykryte podczas skanowania witryny WordPress za pomocą skanera Skipfish

Spośród wszystkich wykrytych podatności najczęściej występowała podatność External content embedded on a page (high risk), wykryto również inny wariant tej podatności (low risk). Obie podatności oznaczają, że strona pobiera dane z zewnętrznego źródła bez odpowiedniego filtrowania i walidacji. Brak poprawnego sprawdzania danych wejściowych może otworzyć drogę do przeprowadzenia najbardziej niebezpiecznych ataków takich jak Cross-Site Scripting i SQL Injection.

Następnie badaniu została poddana witryna Joomla. Rezultaty przedstawiono na rysunku 11.

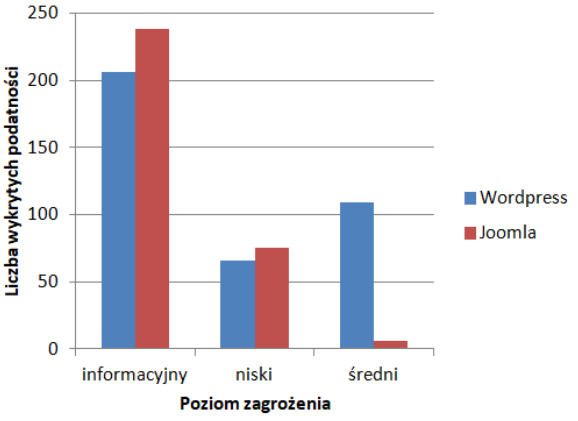


Rysunek 11: Podatności wykryte podczas skanowania witryny Joomla za pomocą skanera Skipfish

W przypadku tej witryny najczęściej występującą podatnością było Directory listing enabled.

Podatność, która występowała niemal równie często podczas skanowania to Incorrect or missing charset. Brak zestawu znaków powoduje, że przeglądarki próbują same odnaleźć pasujący zestaw znaków. W tym momencie atakujący dostaje możliwość stworzenia i wysłania pewnej treści na stronę www. W ostateczności brak zestawu znaków może umożliwić atak typu Cross-Site Scripting. Podczas skanowania znaleziono również formularze, które nie miały zaimplementowanej ochrony przeciwko atakowi typu XSRF (Cross-Site Request Forgery).

Na rysunku 12 przedstawiono liczbę znalezionych podatności podzielonych według poziomu zagrożenia.



Rysunek 12: Liczba podatności według poziomu zagrożenia wykryta podczas skanowania programem Skipfish

Podczas skanowania narzędziem Skipfish w badanych witrynach nie wykryto podatności o wysokim

stopniu zagrożenia. W zagrożeniach poziomu informacyjnego i niskiego widać, że wyniki są podobne, w obu przypadkach więcej podatności wykryto w witrynie Joomla. Natomiast dla zagrożeń poziomu średniego, czyli najgroźniejszych wykrytych podatności, witryna WordPress prezentuje się o wiele gorzej od Joomla. W pierwszej kolejności w WordPress wykryto aż 109 podatności poziomu średniego, podczas gdy w Joomla liczba ta wynosi 6. Poza tym w witrynie Joomla wykryto tylko jedną kategorię zagrożeń poziomu średniego, to jest kategorię Interesting file. Natomiast w WordPress wykryto dwie takie kategorie (Interesting server messager oraz External content embedded). Biorąc to wszystko pod uwagę, w tym porównaniu WordPress wygląda na witrynę gorzej zabezpieczoną.

6.5. Wyniki zbiorcze

Tabela 1: Suma podatności znalezionych przez skanery podczas badań witryn WordPress i Joomla

	WordPress	Joomla
Liczba wykrytych podatności	1444	508

Po przeanalizowaniu danych z tab. 1 widać znaczną przewagę liczby podatności znalezionych w WordPress, w tej witrynie znaleziono ok. 3 razy więcej podatności niż w Joomla. W znacznym stopniu odpowiedzialne za ten wynik są cztery kategorie podatności, gdzie w ramach każdej znaleziono ponad 200 podatności. W sumie te cztery kategorie zawierają połowę wszystkich wykrytych podatności w witrynie WordPress. Ten CMS udostępnia więcej funkcjonalności niż Joomla, co przekłada się na bardziej rozbudowane drzewo katalogów i więcej plików źródłowych, z których każdy musi być zabezpieczony. Drugim powodem wykrycia tak dużej liczby podatności jest niewątpliwie słabe zabezpieczenie tych plików w witrynie WordPress. Po dokładniejszym przeanalizowaniu wyników okazało się, że w wybranych katalogach obu witryn o podobnej liczbie zasobów, w których wykryto daną podatność, w witrynie WordPress większość zasobów danego katalogu zawierało podatność podczas gdy w witrynie Joomla było tylko kilka wykrytych przypadków.

Tabela 2: Podatności odkryte podczas badań pogrupowane według poziomu zagrożenia

	Witryna	
Poziom zagrożenia	WordPress	Joomla
wysoki	38	34
średni	837	160
niski	97	109
informacyjny	472	205

Po zsumowaniu podatności według poziomów zagrożenia (tab. 2) okazało się, że w każdym poziomie, oprócz niskiego, Joomla okazuje się bardziej bezpiecznym systemem CMS. WordPress przegrywa w najważniejszych kategoriach, czyli w poziomie wysokim i średnim. Różnica wyników dla poziomu wysokiego nie jest znaczna, nie rozstrzyga o tym, która witryna jest lepiej zabezpieczona, jednak jeśli wziąć pod uwagę rozkład podatności w poszczególnych kategoriach, okazuje się, że w WordPress wykryto bardzo dużo podatności typu SQL injection. Ta podatność została uznana w raporcie OWASP ZAP Top 10 [12] za najgroźniejszą podatność wykrywaną w aplikacjach internetowych. W podatnościach poziomu średniego widać znaczącą przewagę witryny Joomla. Na tym poziomie zagrożenia przeważają podatności odnoszące się do plików źródłowych, a w tej kategorii, jak już wspomniano wcześniej, WordPress pod względem bezpieczeństwa przegrywa z Joomla.

7. Wnioski

Podczas analizy okazało się, że wyniki szczegółowe nie wskazują jednoznacznie na to, która witryna jest lepiej zabezpieczona.

Po zagłębieniu się w wyniki szczegółowe widoczne jest, że w większości porównań witryna Joomla wypada lepiej, jednak były też takie dane, które wskazywały na WordPress jako witrynę lepiej zabezpieczoną.

Jednak po zsumowaniu wyników i rozpatrywaniu ich jako całość można wyciągnąć tylko jeden wniosek: to Joomla jest lepiej zabezpieczoną witryną. Joomla wygrywa z WordPress w każdej z kategorii: podatności w poszczególnych kategoriach oraz wykryte podatności według poziomu zagrożenia. Jedną z przyczyn takiego stanu rzeczy jest niewątpliwie to, że WordPress jest bardziej rozbudowaną platformą, wprowadza więcej funkcjonalności niż Joomla, co przekłada się na więcej zasobów, które należy zabezpieczyć. Jednak często wykrywano kilka podatności, które odnosiły się do tego samego miejsca, co wskazuje na niższy poziom bezpieczeństwa witryny WordPress.

Podsumowując, przeprowadzone badania wskazują na Joomla jako lepiej zabezpieczoną platformę CMS. Jednak obie witryny mają poważne luki w zabezpieczeniach. Z tego powodu żadna z badanych witryn w podstawowej konfiguracji nie powinna przechowywać wrażliwych danych.

Literatura

[1] Acunetix Web Application Vulnerability Report 2019, https://cdn2.hubspot.net/hubfs/4595665/Acunetix_web_application_vulnerability_report_2019.pdf, [04.05.2020].

[2] M. Meike, J. Sametinger, A. Wiesauer, Security in Open Source Web Content Management Systems, IEEE Security and Privacy Magazine, 2009.

[3] S.K. Patel, V.R Rathod, S. Parikh, Comparative Analysis Of Web Security In Open Source Content Management System, ISSP, 2013.

[4] S.K. Patel, V.R Rathod, S. Parikh, Joomla. Drupal and WordPress - A Statistical Comparison of Open Source CMS, IEEE, 2011.

[5] A. Sagala, E. Manurung, Testing and Comparing Result Scanning Using Web Vulnerability Scanner, American Scientific Publishers, 2015.

[6] Usage statistics of content management systems, https://w3techs.com/technologies/overview/content_management, [07.04.2020].

[7] C. Pepper., M. Tietz, D. Weeks, Open Source Development and Application Security Survey Analysis, Securosis, 2014.

[8] WordPress, https://WordPress.org, [07.04.2020].

[9] Joomla!, https://www.joomla.org/, [07.04.2020].

[10] OWASP ZAP, https://www.zaproxy.org/, [21.05.2020].

[11] Vega, https://subgraph.com/vega/, [21.05.2020].

[12] Detectify, https://detectify.com/, [21.05.2020].

[13] SkipfishDoc, https://code.google.com/archive/p/skipfish/wikis/SkipfishDoc.wiki, [21.05.2020].