Eksploatacja i bezpieczeństwo systemów

dr inż. Mirosław Mazurek

Zakład Systemów Złożonych
Bud. F, pok. 305, tel. 17 865 11 04

OWASP

OWASP Open Web Application Security Project oznacza społeczność online, która tworzy artykuły, metodologie, dokumentację, narzędzia i technologie z zakresu bezpieczeństwa aplikacji internetowych.

The Top 10 OWASP 2020

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access control
- Security misconfigurations
- Cross Site Scripting (XSS)
- Insecure Deserialization
- Using Components with known vulnerabilities
- Insufficient logging and monitoring

- Wstrzyknięcie
- Zepsute uwierzytelnianie
- Narażenie wrażliwych danych
- Jednostki zewnętrzne XML (XXE)
- Uszkodzona kontrola dostępu
- Błędne konfiguracje bezpieczeństwa
- Cross Site Scripting (XSS)
- Niepewna deserializacja
- Korzystanie ze składników ze znanymi lukami w zabezpieczeniach
- Niewystarczające rejestrowanie i monitorowanie

https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2020/

Injection

Wstrzyknięcie kodu ma miejsce, gdy osoba atakująca wysyła nieprawidłowe dane do aplikacji sieci Web z zamiarem zmuszenia jej do zrobienia czegoś, do czego aplikacja nie została zaprojektowana / zaprogramowana.

Najczęstszym przykładem tej luki w zabezpieczeniach jest zapytanie SQL używające niezaufane dane.

String query = “SELECT * FROM accounts WHERE custID = ‘” + request.getParameter(“id”) + “‘”;

To zapytanie można wykorzystać, wywołując stronę internetową wykonującą go za pomocą następującego adresu URL: http://example.com/app/accountView?id= 'lub' 1 '=' 1, powodując zwrócenie wszystkich wierszy przechowywanych w tabela bazy danych.

Istotą luki w iniekcji kodu jest brak walidacji i dezynfekcji danych wykorzystywanych przez aplikację internetową, co oznacza, że ta luka może występować w prawie każdym typie technologii.

Wszystko, co przyjmuje parametry jako dane wejściowe, może potencjalnie być podatne na atak wstrzyknięcia kodu.

Injection

Zapobieganie podatności na wstrzykiwanie kodu naprawdę zależy od technologii używanej w witrynie. Na przykład, jeśli używasz WordPressa, możesz zminimalizować podatność na wstrzykiwanie kodu, ograniczając go do minimum zainstalowanych wtyczek i motywów.

Jeśli masz dostosowaną aplikację internetową i dedykowany zespół programistów, musisz upewnić się, że masz wymagania dotyczące bezpieczeństwa, których mogą przestrzegać programiści podczas projektowania i pisania oprogramowania. Pozwoli im to myśleć o bezpieczeństwie podczas cyklu życia projektu.

Injection

Zalecenia techniczne OWASP:

Zapobieganie iniekcjom SQL wymaga oddzielenia danych od poleceń i zapytań.

Preferowaną opcją jest użycie bezpiecznego interfejsu API, który całkowicie eliminuje użycie interpretera lub zapewnia sparametryzowany interfejs lub migrację w celu użycia narzędzi do mapowania obiektowo-relacyjnego (ORM). Uwaga: Nawet po sparametryzowaniu procedury składowane mogą nadal wprowadzać wstrzykiwanie SQL, jeśli PL / SQL lub T-SQL konkatenują zapytania i dane lub wykonują wrogie dane za pomocą EXECUTE IMMEDIATE lub exec ().

Użyj pozytywnej lub „białej listy” walidacji danych wejściowych po stronie serwera. Nie jest to pełna obrona, ponieważ wiele aplikacji wymaga znaków specjalnych, takich jak pola tekstowe lub interfejsy API dla aplikacji mobilnych.

Injection

W przypadku pozostałych dynamicznych zapytań należy uciec specjalne znaki, używając specjalnej składni Escape dla tego interpretera. Uwaga: struktury SQL, takiej jak nazwy tabel, nazwy kolumn itp., Nie można zmienić znaczenia, dlatego nazwy struktur dostarczone przez użytkownika są niebezpieczne. Jest to powszechny problem w oprogramowaniu do pisania raportów.

Użyj LIMIT i innych formantów SQL w zapytaniach, aby zapobiec masowemu ujawnianiu rekordów w przypadku wstrzyknięcia SQL.

Z tych zaleceń można wyodrębnić dwie rzeczy:

Oddzielenie danych od logiki aplikacji internetowej.
Wdrożyć ustawienia i / lub ograniczenia, aby ograniczyć narażenie danych w przypadku udanych ataków z zastrzykiem.

Bez odpowiednich środków wstrzyknięcie kodu stanowi poważne zagrożenie dla właścicieli witryn. Ataki te wykorzystują luki bezpieczeństwa w celu wrogiego przejęcia lub wycieku poufnych informacji.

Broken Authentication

Luka w uwierzytelnianiu może pozwolić osobie atakującej na użycie metod ręcznych i / lub automatycznych w celu uzyskania kontroli nad dowolnym kontem w systemie - lub, co gorsza, w celu uzyskania pełnej kontroli nad systemem.

Nieprawidłowe uwierzytelnianie zwykle odnosi się do problemów logicznych, które występują w mechanizmie uwierzytelniania aplikacji, takich jak złe zarządzanie sesjami podatne na wyliczanie nazw użytkowników - gdy złośliwy aktor stosuje techniki siłowe w celu odgadnięcia lub potwierdzenia prawidłowych użytkowników w systemie.

Aby zminimalizować ryzyko zepsutego uwierzytelnienia, unikaj pozostawienia strony logowania dla administratorów publicznie dostępnych dla wszystkich odwiedzających witrynę:

/ administrator w Joomla !,
/ wp-admin / on WordPress,
/index.php/admin w Magento,
/ user / login na Drupal.

Najczęstszą formą tej wady jest umożliwienie użytkownikom brutalnego wymuszenia kombinacji nazwy użytkownika i hasła względem tych stron.

Broken Authentication

Według OWASP Top 10 luki te mogą przybierać różne formy. Aplikacja internetowa zawiera usterkę zepsutego uwierzytelniania, jeśli:

- Zezwala na zautomatyzowane ataki, takie jak wypełnianie poświadczeń, w których osoba atakująca ma listę prawidłowych nazw użytkowników i haseł.
- Umożliwia brutalną siłę lub inne automatyczne ataki.
- Zezwala na domyślne, słabe lub dobrze znane hasła, takie jak „Hasło1” lub „admin / admin”.
- Wykorzystuje słabe lub nieskuteczne odzyskiwanie danych uwierzytelniających i procesy zapomnianego hasła, takie jak „odpowiedzi oparte na wiedzy”, których nie można zabezpieczyć.
- Używa zwykłego hasła, szyfrowanych lub słabo zakodowanych haseł.
- Brakuje lub jest nieskuteczne uwierzytelnianie wieloskładnikowe.
- Udostępnia identyfikatory sesji w adresie URL (np. Przepisywanie adresów URL).
- Nie obraca identyfikatorów sesji po udanym logowaniu.
- Nie unieważnia poprawnie identyfikatorów sesji. Sesje użytkownika lub tokeny uwierzytelniające (szczególnie tokeny logowania jednokrotnego) nie są poprawnie unieważniane podczas wylogowywania lub w okresie bezczynności.

Pisanie niezabezpieczonego oprogramowania powoduje większość z tych luk. Można je przypisać wielu czynnikom, takim jak brak doświadczenia programistów. Może to być również konsekwencją bardziej zinstytucjonalizowanych awarii, takich jak brak wymagań bezpieczeństwa lub organizacje, które pędziły wydania oprogramowania, innymi słowy, wybierając działające oprogramowanie zamiast bezpiecznego oprogramowania.

Broken Authentication

Aby uniknąć luk w zabezpieczeniach związanych z uszkodzonym uwierzytelnianiem, upewnij się, że programiści stosują się do najlepszych praktyk bezpieczeństwa witryny. Wspieraj ich, zapewniając dostęp do zewnętrznych audytów bezpieczeństwa i wystarczająco dużo czasu, aby poprawnie przetestować kod przed wdrożeniem do produkcji.

Zalecenia techniczne OWASP są następujące:

- Tam, gdzie to możliwe, należy wdrożyć uwierzytelnianie wieloskładnikowe, aby zapobiec automatycznemu upychaniu poświadczeń, brutalnej sile i kradzieży ataków ponownego użycia poświadczeń.
- Nie wysyłaj ani nie wdrażaj z domyślnymi poświadczeniami, szczególnie dla użytkowników administracyjnych.
- Wdrażaj słabe hasła, takie jak testowanie nowych lub zmienionych haseł na liście 10 000 najgorszych haseł.
- Dostosuj zasady długości, złożoności i rotacji haseł do wytycznych NIST 800-63 B w sekcji 5.1.1 dotyczących zapamiętanych tajemnic lub innych nowoczesnych, opartych na dowodach zasad haseł.
- Upewnij się, że rejestracja, odzyskiwanie poświadczeń i ścieżki API są zabezpieczone przed atakami polegającymi na wyliczeniu konta, używając tych samych komunikatów dla wszystkich wyników.
- Ograniczaj lub coraz częściej opóźniaj nieudane próby logowania. Rejestruj wszystkie awarie i powiadamiaj administratorów o wykryciu upychania poświadczeń, brutalnej siły lub innych ataków.
- Skorzystaj z bezpiecznego, wbudowanego menedżera sesji po stronie serwera, który po zalogowaniu generuje nowy losowy identyfikator sesji z wysoką entropią. Identyfikatory sesji nie powinny znajdować się w adresie URL. Identyfikatory powinny być również bezpiecznie przechowywane i unieważniane po wylogowaniu, bezczynności i bezwzględnych limitach czasu.

Security Misconfigurations

Ataki brutalne polegają na wypróbowaniu wielu możliwych kombinacji, ale istnieje wiele wariantów tego ataku, aby zwiększyć jego skuteczność. Oto najczęstsze:

- Niepoprawne wady
- Domyślne konfiguracje
- Nieużywane strony
- Niechronione pliki i katalogi
- Niepotrzebne usługi

Jednym z najczęstszych błędów webmastera jest zachowanie domyślnych konfiguracji CMS.

Dzisiejsze aplikacje CMS (choć łatwe w użyciu) mogą być trudne z punktu widzenia bezpieczeństwa dla użytkowników końcowych. Zdecydowanie najczęstsze ataki są całkowicie zautomatyzowane. Wiele z tych ataków polega na tym, że użytkownicy mają tylko ustawienia domyślne.

Oznacza to, że dużą liczbę ataków można złagodzić, zmieniając ustawienia domyślne podczas instalowania systemu CMS.

Istnieją ustawienia, które możesz dostosować, aby kontrolować komentarze, użytkowników i widoczność informacji o użytkownikach. Uprawnienia do plików to kolejny przykład domyślnego ustawienia, które można wzmocnić.

Security Misconfigurations

Gdzie może wystąpić błędna konfiguracja zabezpieczeń?

Błędna konfiguracja może się zdarzyć na dowolnym poziomie stosu aplikacji, w tym:

Usługi sieciowe
Platforma
serwer internetowy
Serwer aplikacji
Baza danych
Ramki
Kod niestandardowy
Fabrycznie zainstalowane maszyny wirtualne
Pojemniki
Przechowywanie

Security Misconfigurations

Przykłady scenariuszy ataku z błędną konfiguracją zabezpieczeń

Według OWASP są to niektóre przykłady scenariuszy ataku:

Scenariusz nr 1: serwer aplikacji jest dostarczany z przykładowymi aplikacjami, które nie są usuwane z serwera produkcyjnego.

Te przykładowe aplikacje mają znane wady bezpieczeństwa, które atakujący wykorzystują do złamania zabezpieczeń serwera. Jeśli jedna z tych aplikacji jest konsolą administracyjną, a konta domyślne nie zostały zmienione, atakujący loguje się przy użyciu domyślnych haseł i przejmuje kontrolę.

Scenariusz # 2: Wykaz katalogów nie jest wyłączony na serwerze. Atakujący odkrywa, że może po prostu wyświetlić listę katalogów. Znajdują i pobierają skompilowane klasy Java, które dekompilują i inżynierii wstecznej, aby wyświetlić kod. Atakujący następnie stwierdza poważną wadę kontroli dostępu w aplikacji.

Scenariusz # 3: konfiguracja serwera aplikacji pozwala na szczegółowe komunikaty o błędach, np. stosy śladów, które zostaną zwrócone użytkownikom. Potencjalnie ujawnia to poufne informacje lub podstawowe wady, takie jak wersje komponentów. Wiadomo, że są wrażliwe.

Scenariusz # 4: dostawca usług w chmurze ma domyślne uprawnienia udostępniania otwarte dla Internetu przez innych użytkowników CSP. Umożliwia to dostęp do przechowywanych wrażliwych danych w ramach pamięci w chmurze.

Security Misconfigurations

Aby zapobiec błędnym konfiguracjom zabezpieczeń:

Powtarzalny proces hartowania, który umożliwia szybkie i łatwe wdrożenie innego środowiska, które jest odpowiednio zablokowane. Środowiska programistyczne, kontroli jakości i produkcyjne powinny być skonfigurowane identycznie, z różnymi poświadczeniami używanymi w każdym środowisku. Zautomatyzuj ten proces, aby zminimalizować wysiłek wymagany do skonfigurowania nowego bezpiecznego środowiska.

Minimalna platforma bez zbędnych funkcji, komponentów, dokumentacji i próbek. Usuń lub nie instaluj nieużywanych funkcji i ram.

Zadanie przeglądania i aktualizacji konfiguracji odpowiednich dla wszystkich uwag bezpieczeństwa, aktualizacji i poprawek w ramach procesu zarządzania poprawkami. W szczególności sprawdź uprawnienia do przechowywania w chmurze.

Segmentowa architektura aplikacji, która zapewnia efektywną i bezpieczną separację między komponentami lub dzierżawcami, z segmentacją, konteneryzacją lub grupami zabezpieczeń w chmurze.

Wysyłanie dyrektyw bezpieczeństwa do klientów, np. Nagłówki bezpieczeństwa.

Zautomatyzowany proces weryfikacji skuteczności konfiguracji i ustawień we wszystkich środowiskach.

Cross-Site Scripting (XSS)

Cross Site Scripting (XSS) to szeroko rozpowszechniona luka, która wpływa na wiele aplikacji internetowych. Ataki XSS polegają na wstrzykiwaniu złośliwych skryptów po stronie klienta do strony internetowej i wykorzystywaniu strony jako metody propagacji.

Ryzyko związane z XSS polega na tym, że pozwala atakującemu na wstrzyknięcie treści do strony internetowej i modyfikację sposobu jej wyświetlania, zmuszając przeglądarkę ofiary do wykonania kodu dostarczonego przez atakującego podczas ładowania strony.

XSS jest obecny w około dwóch trzecich wszystkich aplikacji.

Ogólnie rzecz biorąc, luki w zabezpieczeniach XSS wymagają uruchomienia przez użytkownika pewnego rodzaju interakcji, albo za pomocą inżynierii społecznej, albo wizyty na określonej stronie. Jeśli usterka XSS nie zostanie załatana, może być bardzo niebezpieczna dla dowolnej witryny.

Cross-Site Scripting (XSS)

Przykłady luk w zabezpieczeniach XSS

Wyobraź sobie, że jesteś w panelu administracyjnym WordPress dodając nowy post. Jeśli używasz wtyczki z zapisaną luką XSS, która jest wykorzystywana przez hakera, może ona zmusić twoją przeglądarkę do utworzenia nowego użytkownika administracyjnego, gdy jesteś w panelu wp-admin lub może edytować post i wykonywać inne podobne działania .

Luka w zabezpieczeniach XSS daje atakującemu prawie pełną kontrolę nad najważniejszym obecnie oprogramowaniem komputerów: przeglądarkami.

W 2017 roku nasz zespół badawczy ujawnił lukę w zabezpieczeniach XSS przechowywaną w rdzeniu witryn WordPress. Zdalni napastnicy mogą wykorzystać tę lukę, aby zniszczyć losowy post na stronie WordPress i przechowywać w niej złośliwy kod JavaScript.

Cross-Site Scripting (XSS)

Reflected XSS: Aplikacja lub interfejs API zawiera nieważne i nieskalowane dane wejściowe użytkownika jako część danych wyjściowych HTML. Pomyślny atak może pozwolić atakującemu na wykonanie dowolnego kodu HTML i JavaScript w przeglądarce ofiary. Zazwyczaj użytkownik musi wchodzić w interakcje z jakimś złośliwym linkiem prowadzącym do strony kontrolowanej przez osobę atakującą, taką jak złośliwe strony z wodopojem, reklamy lub podobne.

Przechowany XSS: Aplikacja lub interfejs API przechowuje niezaszyfrowane dane wejściowe użytkownika, które są przeglądane później przez innego użytkownika lub administratora. Przechowywany XSS jest często uważany za wysokie lub krytyczne ryzyko.

DOM XSS: Frameworki JavaScript, aplikacje jednostronicowe i interfejsy API, które dynamicznie zawierają dane kontrolowane przez osobę atakującą na stronę, są wrażliwe na DOM XSS. Idealnie byłoby, gdyby aplikacja nie wysyłała danych kontrolowanych przez osobę atakującą do niebezpiecznych interfejsów API JavaScript. Typowe ataki XSS obejmują kradzież sesji, przejęcie konta, obejście MFA, zastąpienie lub unieważnienie węzła DOM (takie jak panele logowania trojana), ataki na przeglądarkę użytkownika, takie jak pobieranie złośliwego oprogramowania, rejestrowanie kluczy i inne ataki po stronie klienta.

Cross-Site Scripting (XSS)

Jak zapobiegać lukom w zabezpieczeniach XSS

Środki zapobiegawcze mające na celu zmniejszenie szans na ataki XSS powinny uwzględniać oddzielenie niezaufanych danych od aktywnej zawartości przeglądarki. Wytyczne OWASP zawierają kilka praktycznych wskazówek, jak to osiągnąć:

Używając frameworków, które automatycznie uciekają z XSS zgodnie z projektem, takich jak najnowszy Ruby on Rails, React JS. Poznaj ograniczenia ochrony XSS dla każdego frameworka i odpowiednio postępuj z przypadkami użycia, które nie zostały uwzględnione.

Ucieczka niezaufanych danych żądania HTTP na podstawie kontekstu w danych wyjściowych HTML (treść, atrybut, JavaScript, CSS lub adres URL) usunie luki w Odbiciu i Zapisanym XSS. Ściągawka OWASP do zapobiegania XSS zawiera szczegółowe informacje na temat wymaganych technik ucieczki danych.

Zastosowanie kodowania kontekstowego podczas modyfikowania dokumentu przeglądarki po stronie klienta działa przeciwko DOM XSS. Gdy nie da się tego uniknąć, do interfejsów API przeglądarki można zastosować podobne kontekstowe techniki zmiany znaczenia, jak opisano w arkuszu próbnym OWASP dotyczącym zapobiegania XSS opartym na modelu DOM.

Włączenie polityki bezpieczeństwa treści (CSP) to kompleksowa kontrola ograniczająca zagrożenie w stosunku do XSS. Jest skuteczny, jeśli nie istnieją inne luki, które pozwoliłyby na umieszczenie złośliwego kodu za pomocą plików lokalnych (np. Nadpisywanie ścieżki przejścia lub podatne biblioteki z dozwolonych sieci dostarczania treści).

Insecure Deserialization

Uwaga: W pierwszej dziesiątce OWASP zauważono, że to zagrożenie bezpieczeństwa zostało dodane przez badanie branżowe i nie opierało się na kwantyfikowalnych badaniach danych.

Każdy twórca stron internetowych musi pogodzić się z faktem, że osoby atakujące / badacze bezpieczeństwa będą próbowali bawić się wszystkim, co wchodzi w interakcję z ich aplikacjami - od adresów URL po obiekty serializowane.

W informatyce obiekt jest strukturą danych; innymi słowy sposób na uporządkowanie danych. Aby ułatwić zrozumienie niektórych kluczowych pojęć:

Proces serializacji przekształca obiekty w ciągi bajtów.
Proces deserializacji polega na konwersji ciągów bajtów na obiekty.

Przykłady niepewnych scenariuszy ataku deserializacji

Zgodnie z wytycznymi OWASP, oto kilka przykładów scenariuszy ataku:

Scenariusz nr 1: aplikacja React wywołuje zestaw mikrousług Spring Boot. Będąc programistami funkcjonalnymi, starali się upewnić, że ich kod jest niezmienny. Rozwiązaniem, które wymyślili, jest szeregowanie stanu użytkownika i przekazywanie go tam iz powrotem przy każdym żądaniu. Osoba atakująca zauważa sygnaturę obiektu Java „R00” i używa narzędzia Java Serial Killer do uzyskania zdalnego wykonania kodu na serwerze aplikacji.

Scenariusz # 2: Forum PHP używa serializacji obiektów PHP do zapisania „super” ciasteczka, zawierającego identyfikator użytkownika, rolę, skrót hasła i inne stany:

a: 4: {i: 0; i: 132; i: 1; s: 7: „Mallory”; i: 2; s: 4: „użytkownik”;

i: 3; s: 32: ”b6a8b3bea87fe0e05022f8f3c88bc960 ″;}

Osoba atakująca zmienia obiekt zserializowany, aby dać sobie uprawnienia administratora:

Insecure Deserialization

Jak zapobiegać niebezpiecznej deserializacji

Najlepszym sposobem ochrony aplikacji sieci Web przed tego rodzaju ryzykiem jest nieakceptowanie obiektów serializowanych z niezaufanych źródeł.

Jeśli nie możesz tego zrobić, zabezpieczenia OWASP zawierają więcej technicznych zaleceń, które Ty (lub Twoi programiści) możecie spróbować wdrożyć:

Wdrożenie kontroli integralności, takiej jak podpisy cyfrowe na dowolnych szeregowanych obiektach, aby zapobiec tworzeniu wrogich obiektów lub manipulowaniu danymi.

Egzekwowanie ścisłych ograniczeń typów podczas deserializacji przed utworzeniem obiektu, ponieważ kod zwykle oczekuje definiowalnego zestawu klas. Wykazano obejścia tej techniki, więc poleganie wyłącznie na niej nie jest wskazane.

Izolowanie i uruchamianie kodu, który w miarę możliwości deserializuje się w środowiskach o niskich uprawnieniach.

Rejestrowanie wyjątków i awarii deserializacji, na przykład gdy typ przychodzący nie jest typem oczekiwanym lub deserializacja generuje wyjątki.

Ograniczanie lub monitorowanie przychodzących i wychodzących połączeń sieciowych z kontenerów lub serwerów, które dokonują deserializacji.

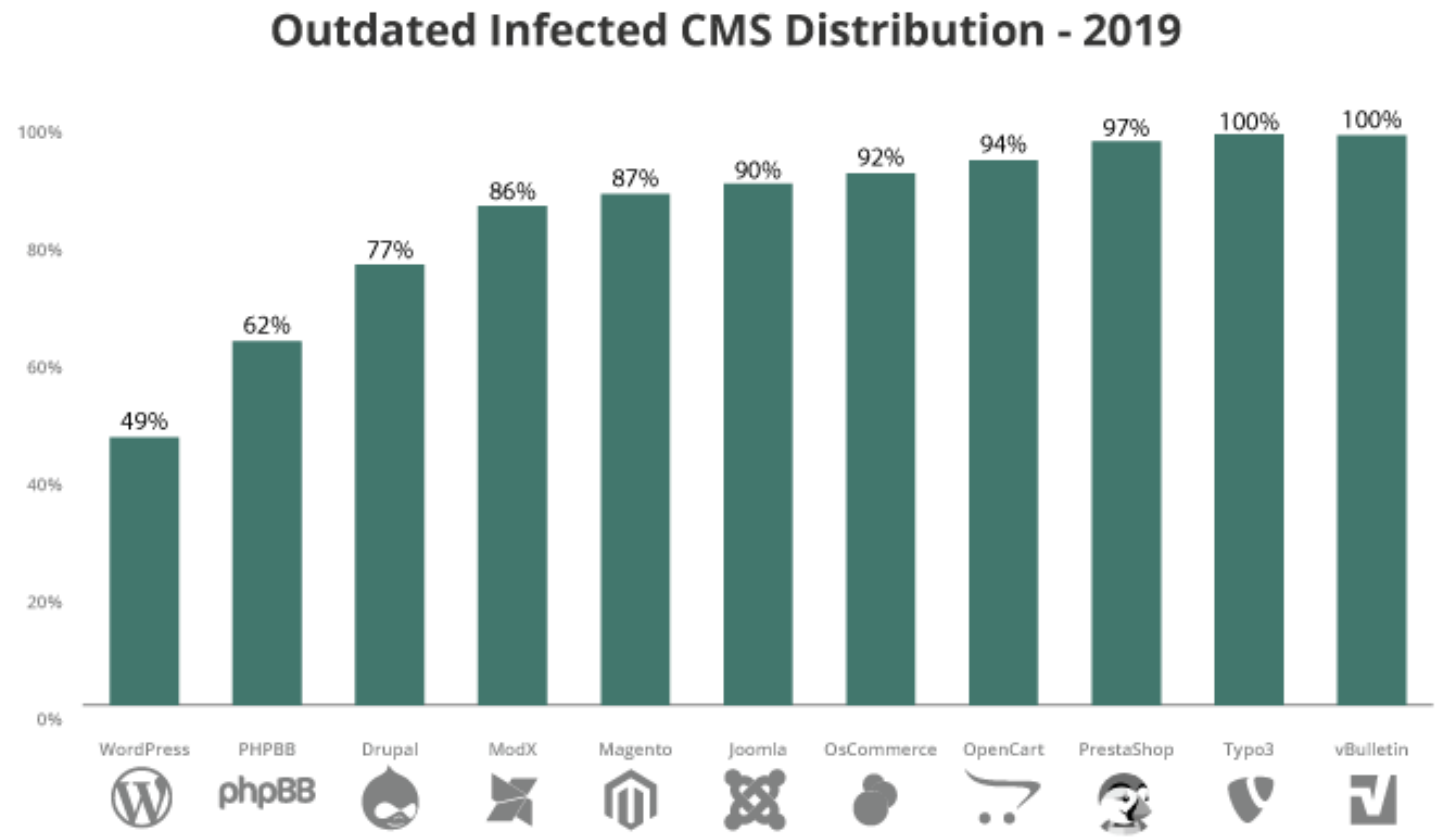
Monitorowanie deserializacji, ostrzeganie, jeśli użytkownik nieustannie deserializuje.

Using Components with Known Vulnerabilities

Obecnie nawet proste strony internetowe, takie jak blogi osobiste, mają wiele zależności.

Wszyscy możemy się zgodzić, że niezaktualizowanie każdego oprogramowania na zapleczu i froncie witryny bez wątpienia spowoduje poważne zagrożenie bezpieczeństwa wcześniej niż później.

Na przykład w 2019 r. 56% wszystkich aplikacji CMS było nieaktualnych w momencie infekcji.



Using Components with Known Vulnerabilities

Pytanie brzmi: dlaczego nie aktualizujemy naszego oprogramowania na czas? Dlaczego dzisiaj jest to tak ogromny problem?

Istnieje kilka możliwości, takich jak:

Webmasterzy / programiści nie nadążają za tempem aktualizacji; w końcu poprawna aktualizacja wymaga czasu.

Starszy kod nie będzie działał w nowszych wersjach jego zależności.

Webmasterzy obawiają się, że coś pęknie na ich stronie internetowej.

Webmasterzy nie mają specjalistycznej wiedzy, aby poprawnie zastosować aktualizację.

Może to być trochę zbyt dramatyczne, ale za każdym razem, gdy zignorujesz ostrzeżenie o aktualizacji, możesz pozwolić, aby znana luka przetrwała w twoim systemie. Zaufaj nam, cyberprzestępcy szybko badają oprogramowanie i listy zmian.

Niezależnie od przyczyny uruchamiania przestarzałego oprogramowania w Twojej aplikacji internetowej, nie możesz pozostawić go bez ochrony. Zarówno Sucuri, jak i OWASP zalecają wirtualne łatanie w przypadkach, gdy łatanie nie jest możliwe.

Wirtualne łatanie pozwala stronom internetowym, które są przestarzałe (lub mają znane luki w zabezpieczeniach), ochronę przed atakami, zapobiegając ich wykorzystaniu w locie. Zazwyczaj odbywa się to za pomocą zapory ogniowej i systemu wykrywania włamań.

Using Components with Known Vulnerabilities

Wrażliwe aplikacje

Wrażliwe aplikacje są zwykle nieaktualne, zgodnie z wytycznymi OWASP, jeśli:

Nie znasz wersji wszystkich używanych komponentów (zarówno po stronie klienta, jak i serwera). Obejmuje to komponenty, których bezpośrednio używasz, a także zagnieżdżone zależności.

Oprogramowanie jest podatne na zagrożenia, nie jest obsługiwane lub nieaktualne. Obejmuje to system operacyjny, serwer WWW / aplikacji, system zarządzania bazą danych (DBMS), aplikacje, interfejsy API i wszystkie komponenty, środowiska wykonawcze i biblioteki.

Nie znasz wersji wszystkich używanych komponentów (zarówno po stronie klienta, jak i serwera). Obejmuje to komponenty, których bezpośrednio używasz, a także zagnieżdżone zależności.

Nie naprawiasz ani nie uaktualniasz podstawowej platformy, struktur i zależności w sposób terminowy, oparty na ryzyku. Zdarza się to zwykle w środowiskach, w których łatanie jest co miesiąc lub co kwartał kontrolowane przez zmiany, co pozostawia organizacje otwarte na wiele dni lub miesięcy niepotrzebnego narażenia na naprawione luki.

Twórcy oprogramowania nie testują kompatybilności zaktualizowanych, uaktualnionych lub załatanych bibliotek.

Nie zabezpieczasz konfiguracji komponentów.

Using Components with Known Vulnerabilities

Jak unikać używania składników o znanych lukach w zabezpieczeniach

Niektóre sposoby zapobiegania korzystaniu z wrażliwych komponentów to:

Usuń wszystkie niepotrzebne zależności.
Zrób spis wszystkich swoich komponentów po stronie klienta i serwera.
Monitoruj źródła, takie jak typowe luki w zabezpieczeniach i ujawnienia (CVE) oraz National Vulnerability Database (NVD), w celu wykrycia luk w zabezpieczeniach komponentów.
Pozyskaj komponenty tylko z oficjalnych źródeł.
Pozbądź się składników, które nie są aktywnie konserwowane.
Użyj wirtualnego łatania za pomocą Zapory aplikacji WWW.

Insufficient Logging and Monitoring

Nie można nie doceniać znaczenia zabezpieczenia strony internetowej. Chociaż 100% bezpieczeństwa nie jest realistycznym celem, istnieją sposoby regularnego monitorowania witryny, abyś mógł podjąć natychmiastowe działania, gdy coś się wydarzy.

Brak skutecznego procesu rejestrowania i monitorowania może zwiększyć szkody związane z naruszeniem bezpieczeństwa witryny.

W Sucuri zdecydowanie zalecamy, aby każda strona była odpowiednio monitorowana. Jeśli potrzebujesz monitorować swój serwer, OSSEC jest bezpłatnie dostępny, aby Ci pomóc. OSSEC aktywnie monitoruje wszystkie aspekty aktywności systemu za pomocą monitorowania integralności plików, monitorowania dziennika, sprawdzania katalogu głównego i monitorowania procesu.

Insufficient Logging and Monitoring

Według OWASP są to niektóre przykłady scenariuszy ataku z powodu niewystarczającego rejestrowania i monitorowania:

Scenariusz nr 1: włamano się do oprogramowania forum projektu typu open source, prowadzonego przez mały zespół, wykorzystując wadę tego oprogramowania. Atakującym udało się usunąć wewnętrzne repozytorium kodu źródłowego zawierające następną wersję i całą zawartość forum. Chociaż źródło można odzyskać, brak monitorowania, rejestrowania lub ostrzegania doprowadził do znacznie gorszego naruszenia. W wyniku tego problemu projekt oprogramowania forum nie jest już aktywny.

Scenariusz nr 2: osoba atakująca skanuje w poszukiwaniu użytkowników o wspólnym haśle. Mogą przejąć wszystkie konta z tym hasłem. Dla wszystkich innych użytkowników ten skan pozostawia tylko jeden fałszywy login. Po kilku dniach można to powtórzyć przy użyciu innego hasła.

Scenariusz nr 3: Podobno główny amerykański detalista miał wewnętrzną piaskownicę do analizy złośliwego oprogramowania analizującą załączniki. Oprogramowanie piaskownicy wykryło potencjalnie niechciane oprogramowanie, ale nikt nie zareagował na to wykrycie. Piaskownica od dłuższego czasu wyświetlała ostrzeżenia, zanim wykryła naruszenie z powodu nieuczciwych transakcji kartowych przez zewnętrzny bank.

Insufficient Logging and Monitoring

Jak sprawnie monitorować witrynę

Prowadzenie dzienników kontroli jest niezbędne do kontrolowania wszelkich podejrzanych zmian w witrynie. Dziennik kontroli to dokument, który rejestruje zdarzenia w witrynie, dzięki czemu można wykryć anomalie i potwierdzić z osobą odpowiedzialną, że konto nie zostało naruszone.

Niezależnie od przyczyny uruchamiania przestarzałego oprogramowania w Twojej aplikacji internetowej, nie możesz pozostawić go bez ochrony. Zarówno Sucuri, jak i OWASP zalecają wirtualne łatanie w przypadkach, gdy łatanie nie jest możliwe.

Wiemy, że niektórzy użytkownicy mogą mieć trudności z ręcznym wykonywaniem dzienników kontroli. Jeśli masz witrynę WordPress, możesz skorzystać z naszej bezpłatnej wtyczki zabezpieczającej WordPress, która pomoże Ci w logach kontroli. Wtyczkę można pobrać z oficjalnego repozytorium WordPress.
Wrażliwe aplikacje

Platforma bezpieczeństwa witryny Sucuri ma kompleksowe rozwiązanie do monitorowania witryny, które obejmuje:

- Zdalny skaner
- Skaner czarnej listy stron internetowych
- Skaner po stronie serwera
- Skaner DNS
- Skaner czasu działania