OWASP

Open Web Application Security Project

Application Security Verification Standard 3.0.1

Standard weryfikowania bezpieczeństwa aplikacji

Czerwiec 2016

Podziękowania

O standardzie

Application Security Verification Standard (Standard Weryfikacji Bezpieczeństwa Aplikacji) jest listą wymagań dla zabezpieczania aplikacji lub jej testów. Wymagania te mogą być wykorzystywane przez architektów, programistów, testerów, specjalistów do spraw bezpieczeństwa. Mogą też być wykorzystywane przez użytkowników, w celu zdefiniowania jak powinna wyglądać bezpieczna aplikacja.

Licencja i prawa autorskie

Wersja 3.0, 2015

Kierownik projektu	Główny autor	Współpracownicy oraz korektorzy
Andrew van der Stock Daniel Cuthbert	Jim Manico	Abhinav Sejpal Ari Kesäniemi Boy Baukema Colin Watson Cristinel Dumitru David Ryan François-Eric Guyomarc'h Gary Robinson Glenn Ten Cate James Holland Martin Knobloch Raoul Endres Ravishankar S Riccardo Ten Cate Roberto Martelloni Ryan Dewhurst Stephen de Vries Steven van der Baan

Wersja 2.0, 2014

Kierownik projektu	Główni autorzy	Współpracownicy oraz korektorzy
Daniel Cuthbert Sahba Kazerooni	Andrew van der Stock Krishna Raja	Antonio Fontes Colin Watson Jeff Sergeant Pekka Sillanpää Archangel Cuison Dr Emin Tatli Jerome Athias Safuat Hamdy Ari Kesäniemi Etienne Stalmans Jim Manico

Kierownik projektu	Główni autorzy	Współpracownicy oraz korektorzy
		Scott Luc Boy Baukema Evan Gaustad Mait Peekma Sebastien Deleersnyder

Wersja 1.0, 2009

Kierownik projektu	Główny autor	Współpracownicy oraz korektorzy
Mike Boberski Jeff Williams Dave Wichers	Jim Manico	Andrew van der Stock Dr. Sarbari Gupta John Steven Pierre Parrend Barry Boyd Dr. Thomas Braun Ken Huang Richard Campbell Bedirhan Urgun Eoin Keary Ketan Dilipkumar Vyas Scott Matsumoto Colin Watson Gaurang Shah Liz Fong Shouvik Bardhan Dan Cornell George Lawless Mandeep Khera Stan Wisseman Dave Hausladen Jeff LoSapio Matt Presson Stephen de Vries Theodore Winograd Jeremiah Grossman Nam Nguyen Steve Coyle Dave van Stein John Martin Paul Douthit Terrie Diaz

Polskie tłumaczenie ASVS 3.0.1

Niniejsze tłumaczenie powstało staraniem wolontariuszy zgromadzonych wokół społeczności OWASP Polska.

Autorzy niniejszego tłumaczenia starali się dokonać wszelkiej staranności podczas prowadzonych prac. Specyficzny - informatyczny i często slangowy język w niektórych miejscach był niezmiernie trudny do dosłownego przekazania - z tego też powodu niekiedy zdecydowano się na pozostawienie anglicyzmów lub tłumaczeń opisowych. W przypadku sporów dotyczących zastosowanych terminów należy sięgnąć do oryginalnej wersji anglojęzycznej. Dokumentem referencyjnym był dokument z czerwca 2016 r. "Application Security Verification Standard 3.0.1".

Kierownik projektu tłumaczenie	Współpracownicy oraz korektorzy
Mateusz Olejarka	Dawid Bałut
Jakub Syta	Andrzej Dalasiński
	Maciej Grela
	Antoni Grzymała
	Michał Kapica
	Adam Lange
	Tomasz Polański
	Paweł Pietrzyński
	Adam Sokołowski
	Jakub Tomaszewski
	Bogdan Żurek

Jeżeli czytelnikom uda się w treści wykryć niespójności w tłumaczeniu, bardzo prosimy o kontakt z OWASP Polska i zaproponowanie lepszych sformułowań. Zachęcamy także to zaangażowania się w pracę międzynarodowej społeczności OWASP celem doskonalenia tego i innych projektów.

Przedmowa

Zapraszamy do zapoznania się z Application Security Verification Standard (ASVS) w wersji 3.0. Stanowi on efekt prac społeczności, która spróbowała przygotować strukturę ramową dla wymagań bezpieczeństwa i kontroli. Standard koncentruje się na normalizacji funkcjonalnych i niefunkcjonalnych wymagań bezpieczeństwa niezbędnych w trakcie projektowania, rozwoju i testowania nowoczesnych aplikacji internetowych.

ASVS v3.0 jest zwieńczeniem wysiłków społeczności, która brała również pod uwagę informacje zwrotne płynące ze środowiska. W obecnej wersji uznaliśmy za istotne, by podkreślić doświadczenia płynące z rzeczywistych przypadków użycia ASVS. Pomoże to nowym adeptom ASVS w planowaniu możliwości wykorzystania standardu, jednocześnie wspierając obecnych użytkowników w uczeniu się z doświadczeń innych.

Spodziewamy się, że nigdy nie uda się osiągnąć pełnej zgody odnośnie zawartości standardu. Analiza ryzyka jest zawsze w pewnym stopniu subiektywna, co generuje problemy podczas próby uogólnienia w jednym standardzie, który powinien odpowiadać na potrzeby wszystkich organizacji. Mamy jednak nadzieję, że najnowsze aktualizacje wprowadzone w tej wersji są krokiem we właściwym kierunku, oraz w znaczący sposób rozszerzają koncepcje wprowadzone tym ważnym standardem.

Co nowego w wersji 3.0?

W wersji 3.0 dodaliśmy kilka nowych sekcji, dotyczących konfiguracji, usług internetowych (ang. web service) oraz nowoczesnych aplikacji klienckich, by w jeszcze lepszy sposób dostosować standard do nowoczesnych aplikacji. Są one zazwyczaj aplikacjami reagującymi (ang. responsive applications) szeroko stosującymi frontend HTML5 lub klienta mobilnego, który wykorzystuje wspólny zestaw usług sieciowych RESTful przy użyciu uwierzytelniania SAML.

Usunęliśmy również zbędne duplikacje pojawiające się w ramach standardu, na przykład, w celu zapewnienia, że deweloper aplikacji mobilnych nie będzie musiał wielokrotnie testować tych samych elementów.

Przygotowaliśmy także mapowanie na CWE (Wspólny Spis Podatności - ang. Common Weakness Enumeration). Mapowanie CWE mogą być wykorzystane do identyfikacji informacji, takich jak prawdopodobieństwo i konsekwencje udanego wykorzystania podatności. Pomoże to, ogólnie rzecz ujmując, uzyskać wgląd w to, co może się nie udać, gdy mechanizmy bezpieczeństwa nie będą wykorzystywane lub poprawnie wdrożone. Jest to uzupełnione o informacje, w jaki sposób należy usuwać te słabości.

Udało się nam również dotrzeć do szerokiej społeczności i przeprowadzić sesje podczas AppSec UE 2015 oraz w trakcie ostatniej sesji roboczej podczas AppSec USA 2015, podczas których wzajemnie przeglądaliśmy standard. Dzięki temu udało się nam uzyskać ogromną ilość opinii płynących ze społeczności. W trakcie tych przeglądów, w przypadku, gdy sens treści ulegał istotny zmianom, tworzyliśmy nowy mechanizm kontrolny i usuwaliśmy poprzedni. Celowo postanowiliśmy nie wykorzystywać ponownie nieaktualnych

mechanizmów kontrolnych, ponieważ mogłoby to być źródłem nieporozumień. W załączniku A zamieściliśmy pełne mapowanie poczynionych zmian.

Reasumując, wersja 3.0 jest największą zmianą standardu w jego historii. Mamy nadzieję, że uznasz tę aktualizację standardu za użyteczną i będziesz używać go w sposób, który możemy sobie tylko wyobrazić.

Sposób użycia Application Security Verification Standard

ASVS ma dwa główne cele:

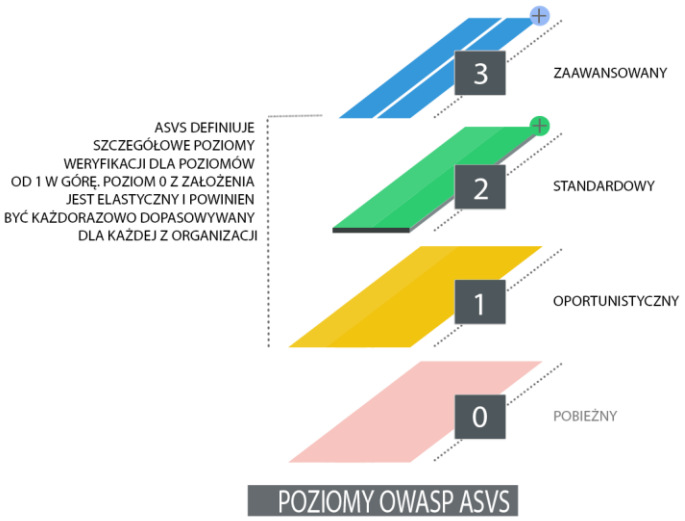
- pomoc organizacjom w rozwoju i utrzymaniu bezpiecznych aplikacji
- umożliwienie podmiotom oferującym usługi bezpieczeństwa, dostawcom narzędzi w zakresie bezpieczeństwa oraz użytkownikom dostosowania wzajemnych wymagań oraz ofert

Poziomy weryfikacji bezpieczeństwa aplikacji

Application Security Verification Standard definiuje trzy poziomy weryfikacji bezpieczeństwa, w ramach każdego z poziomów zwiększa się jej głębokość.

- ASVS Poziom 1 jest przeznaczony dla wszystkich programów.
- ASVS Poziom 2 jest przeznaczony dla aplikacji, które zawierają dane wymagające ochrony.
- ASVS Poziom 3 jest przeznaczony dla najbardziej krytycznych aplikacji - takich, które wykonują transakcje znaczniej wartości, zawierają wrażliwe dane medyczne, a także innych aplikacji, które wymagają najwyższego poziomu zaufania.

Każdy poziom ASVS zawiera listę wymagań bezpieczeństwa. Każde z tych wymagań może być mapowane na specyficzne cechy i właściwości dotyczące zabezpieczeń, które powinny zostać wbudowane w oprogramowanie przez deweloperów.



Rysunek 1 - Poziomy OWASP Application Security Verification Standard 3.0

Jak wykorzystywać standard?

Jednym z najlepszych sposobów korzystania z Application Security Verification Standard jest użycie go jak wytycznych podczas tworzenia listy kontrolnej bezpiecznego kodowania specyficznej dla aplikacji, platformy lub organizacji. Dostosowanie ASVS do konkretnych przypadków zwiększy presję na stosowanie wymogów bezpieczeństwa, które są najważniejsze dla twoich projektów jak i środowisk.

Poziom 1: Oportunistyczny

Aplikacja osiąga Poziom 1 ASVS (Oportunistyczny), jeśli odpowiednio chroni przed lukami w zabezpieczeniach aplikacji, które są łatwe do odkrycia oraz są zawarte w OWASP Top 10 i innych podobnych listach kontrolnych.

Poziom 1 jest zwykle odpowiedni dla aplikacji, dla których wystarczający jest niski poziom zaufania dla prawidłowego stosowania mechanizmów bezpieczeństwa, w przypadkach, gdy potrzebna jest szybka analiza szerokiej listy aplikacji w ramach korporacji lub też jako wsparcie podczas tworzenia listy priorytetowych wymagań bezpieczeństwa podczas szerszych inicjatyw. Zabezpieczenia Poziomu 1 mogą być zapewnione albo automatycznie za pomocą narzędzi albo w sposób ręczny, bez dostępu do kodu źródłowego. Poziom 1 uważamy za minimum wymagane dla wszystkich aplikacji.

Zagrożenia dla aplikacji będą pochodzić najprawdopodobniej ze strony atakujących, którzy wykorzystują proste i niskonakładowe techniki w celu identyfikacji łatwych do znalezienia i łatwych do wykorzystania podatności. Jest to niejako przeciwieństwem dla zdeterminowanego atakującego, który skupi wysiłek na zaatakowaniu konkretnej aplikacji. Jeśli dane przetwarzane przez aplikację mają wysoką wartość, to raczej nie powinieneś się zatrzymać na Poziomie 1.

Poziom 2: Standardowy

Aplikacja osiąga Poziom 2 ASVS (Standardowy) w przypadku, gdy właściwie chroni przed większością obecnych ryzyk dotyczących oprogramowania.

Poziom 2 zapewnia, że mechanizmy bezpieczeństwa są wdrożone, są skuteczne oraz są wykorzystywane wewnątrz aplikacji. Poziom 2 jest zazwyczaj właściwy dla aplikacji, które obsługują istotne transakcje biznesowe - włączając w to przetwarzanie informacji medycznych, informacji ,które realizują funkcje wrażliwe lub krytyczne dla biznesu, a także, które przetwarzają inne zasoby wymagające ochrony.

Zagrożeniem dla Poziomu 2 będą zazwyczaj wykształceni i zmotywowani atakujący, koncentrujący się na konkretnych celach oraz wykorzystujący narzędzia i techniki, które są wyjątkowo skuteczne podczas wykrywania oraz eksploatacji słabości istniejących w ramach aplikacji.

Poziom 3: Zaawansowany

Poziom 3 jest najwyższym poziomem weryfikacji w ramach ASVS. Jest zazwyczaj zarezerwowany dla aplikacji, które wymagają szczególnego poziomu weryfikacji

mechanizmów bezpieczeństwa, takich jak te, które można znaleźć w zastosowaniach wojskowych, ochronie zdrowia i bezpieczeństwa publicznego, ochronie infrastruktury krytycznej itp. Organizacje mogą wymagać zastosowania Poziomu 3 ASVS dla aplikacji, które realizują funkcje krytyczne, w przypadkach, gdy awaria może mieć znaczący wpływ na działalność organizacji, a nawet na jej przetrwanie. Przykładowe wskazówki dla zastosowania ASVS Poziom 3 przedstawiono poniżej. Aplikacja osiąga Poziom 3 ASVS (Zaawansowany) jeśli odpowiednio chroni przed zaawansowanymi podatnościami w zabezpieczeniach aplikacji, a także demonstruje zasady dobrego projektowania zabezpieczeń.

Aplikacja znajdująca się na Poziomie 3 ASVS wymaga bardziej dogłębnej analizy, architektury, kodowania i testowania niż na pozostałych poziomach. Bezpieczna aplikacja w znacznej mierze składa się z modułów (w celu ułatwienia np. jej elastyczności, skalowalności a przede wszystkim stosowania kolejnych warstw zabezpieczeń) a każdy moduł (wydzielony w sposób fizyczny i/lub wykorzystując połączenia sieciowe) dba o poprawność własnych wymagań bezpieczeństwa (tzw. wielowarstwowa ochrona - ang. defence in depth), które muszą być następnie właściwie udokumentowane. Wymagania obejmują zabezpieczenia zapewniające poufność (np. szyfrowanie), integralność (np. transakcje, walidacja wejście), dostępność (np. sprawna obsługa obciążeń - ang. handling load gracefully), uwierzytelnianie (w tym między systemami), niezaprzeczalność, autoryzację i audyt (logowanie).

Praktyczne zastosowania ASVS

Z różnymi zagrożeniami powiązane są różne motywacje. Niektóre branże mają unikalne zasoby informacyjne i technologiczne oraz specyficzne wymagania regulacyjne dotyczące zgodności.

Poniżej przedstawiamy wytyczne dotyczące rekomendowanych poziomów ASVS dostosowane do poszczególnych branż. Chociaż w ramach poszczególnych branż można zauważyć pewne unikalne kryteria i różnice, wspólnym zagrożeniem jest to, że oportunistyczny atakujący będzie szukał wszelkich łatwych do eksploatacji, podatnych aplikacji. Dlatego też Poziom 1 ASVS jest zalecany dla wszystkich aplikacji, niezależnie od branży. Organizacje są jednak zachęcane by bliżej przyjrzały się swoim unikalnym ryzykom biorąc pod uwagę naturę prowadzonej działalności. Na drugim końcu znajduje się Poziom 3 ASVS, który jest zarezerwowany dla tych przypadków, które mogą stanowić zagrożenie dla bezpieczeństwa osobowego lub przypadków, gdy pełne naruszenie bezpieczeństwa aplikacji może poważnie wpłynąć na funkcjonowanie organizacji.

Branża	Profil zagrożenia	Rekomendacje L1	Rekomendacje L2	Rekomendacje L3
Finanse i ubezpieczenia	Chociaż ta branża doświadcza prób ze strony oportunistycznych atakujących, to jest przede wszystkim postrzegana jako cel wysokiej wartości przez zmotywowanych atakujących a ataki są często motywowane finansowo. Zazwyczaj szukają oni danych wymagających ochrony lub informacji uwierzytelniających, które mogą być użyte do popełnienia oszustwa lub wykorzystane bezpośrednio poprzez skorzystanie z funkcjonalności przepływów pieniężnych wbudowanych w aplikacje. Techniki często zawierają kradzież danych uwierzytelniających, ataki na poziomie aplikacji i inżynierię społeczną. Niektórymi istotnymi wymaganiami dla zapewniania zgodności są Payment Card Industry Data Security Standard (PCI DSS),Gramm Leech Bliley Act oraz Sarbanes-Oxley Act (SOX).	Wszystkie aplikacje dostępne przez Internet.	Aplikacje, które zawierają dane wymagające ochrony takie jak numery kart kredytowych, dane osobowe oraz informacje, które mogą posłużyć do przesyłania ograniczonych ilości pieniędzy w ograniczony sposób. Przykładowo: (i) transfer środków pieniężnych pomiędzy rachunkami w ramach tej samej organizacji lub (ii) wolniejsze formy transferu pieniędzy (np. ACH) z ustawionymi limitami transakcyjnymi lub (iii) transfery pieniężne ze sztywno ustawionymi limitami w poszczególnych okresach czasu	Aplikacje, które zawierają duże ilości informacji wymagających ochrony lub pozwalają na szybki transfer dużych sum pieniędzy (np. przelewów) i/lub transfer dużych sum pieniędzy w formie pojedynczych transakcji lub też jako partia mniejszych transferów.

Branża	Profil zagrożenia	Rekomendacje L1	Rekomendacje L2	Rekomendacje L3
Produkcyjna, usługi profesjonalne, transportowa, technologiczna, media, infrastruktura i obronność	Branże te wydają się nie mieć wiele wspólnego, niemniej aktorzy zagrożeń, którzy najczęściej atakują organizacje w tych branżach są bardziej skłonni do wykonywania ukierunkowanych ataków wymagających więcej czasu, umiejętności i zasobów. Często wrażliwe informacje lub systemy nie są łatwe do zlokalizowania i wymagają wsparcia z wewnątrz organizacji i technik inżynierii społecznej. Ataki mogą wykorzystywać osoby z wewnątrz i zewnątrz organizacji lub też jednocześnie z obu tych grup. Ich cele mogą obejmować uzyskanie dostępu do własności intelektualnej pozwalającej na osiągnięcie strategicznej lub technologicznej przewagi. Nie można również przeoczyć atakujących zainteresowanych przerwaniem w funkcjonowania aplikacji, wpłynięciem na jej zachowanie lub zakłóceniem systemów wrażliwych. Większość atakujących szuka danych wymagających ochrony, które mogą być wykorzystane do osiągnięcia pośredniego lub bezpośredniego zysku, w tym danych osobowych lub informacji o płatnościach. Dane te mogą być następnie wykorzystane do kradzieży tożsamości, realizacji oszukańczych płatności lub innych oszustw.	Wszystkie aplikacje dostępne przez Internet.	Aplikacje zawierające informacje wewnętrzne lub też informacje na temat pracowników, które mogą być wykorzystane do przeprowadzenia ataków socjotechnicznych. Aplikacje zawierające niekrytyczną lecz istotną własność intelektualną lub tajemnice handlowe.	Aplikacje zawierające cenną własność intelektualną, tajemnice handlowe lub tajemnice rządowe (na przykład w Stanach Zjednoczonych mogą to być informacje sklasyfikowane jako tajne lub powyżej), które są kluczowe do przetrwania i sukcesu organizacji. Aplikacje sterujące wrażliwymi funkcjonalnościami (np. tranzytowe, produkcyjne, systemy kontroli) lub, które mają możliwość spowodowania zagrożenia dla bezpieczeństwa życia ludzkiego.
Ochrona zdrowia	Większość atakujących szuka danych wymagających ochrony, które mogą być wykorzystane do osiągnięcia pośredniego lub bezpośredniego zysku, w tym danych osobowych lub informacjach o płatnościach. Dane te mogą być następnie wykorzystane do kradzieży tożsamości, realizacji oszukańczych płatności lub innych oszustw. W Stanach Zjednoczonych jest to regulowane poprzez Health Insurance Portability and Accountability Act (HIPAA), reguły powiadamiania o przypadkach naruszenia prywatności lub bezpieczeństwa oraz zasadach ochrony pacjentów (http://www.hhs.gov/ocr/privacy/.	Wszystkie aplikacje dostępne przez Internet.	Aplikacje ze średnią lub niewielką ilością wrażliwy danych medycznych, danymi osobowymi oraz informacjami na temat płatności.	Aplikacje służące do sterowania sprzętem medycznym, urządzeniami lub zapisami, które mogą stanowić zagrożenie dla życia ludzkiego. Systemy płatnicze oraz systemy obsługujące terminale w punktach płatniczych (ang. Point of Sale - POS), które zawierają duże ilości danych transakcyjnych , które mogą być użyte do popełnienia oszustwa. Obejmuje to również interfejsy administracyjne dla tych aplikacji.

Branża	Profil zagrożenia	Rekomendacje L1	Rekomendacje L2	Rekomendacje L3
Handel detaliczny, spożywcza i hotelarska	Wielu atakujących w tym segmencie wykorzystuje oportunistyczne taktyki typu "rozwal i zabierz". Istnieje jednak również stałe zagrożenie atakami na konkretne aplikacje, które zawierają informacje dotyczące płatności, które są wykorzystywane do przeprowadzania transakcji finansowych lub przechowywania danych osobowych. Chociaż rzadziej niż w przypadku wyżej wymienionych zagrożeń, istnieje również możliwość przeprowadzenia bardziej zaawansowanych ataków na te branże nakierowanych na kradzież własności intelektualnej, zdobycia przewagi konkurencyjnej lub uzyskania przewagi biznesowej podczas negocjacji w ramach organizacji docelowej lub wobec jej partnera biznesowego.	Wszystkie aplikacje dostępne przez Internet.	Nadaje się do aplikacji biznesowych, katalogów produktów, wewnętrznych informacji korporacyjnych oraz aplikacji zawierających ograniczone informacje o użytkownikach (np. informacje kontaktowe). Aplikacje z małą lub umiarkowaną ilością danych dotyczących płatności lub z funkcjonalnością kasy.	Systemy płatnicze oraz systemy obsługujące terminale w punktach płatniczych (ang. Point of Sale - POS), które zawierają duże ilości danych transakcyjnych, które mogą być użyte do popełnienia oszustwa. Obejmuje to również interfejsy administracyjne dla tych aplikacji. Aplikacje z dużą ilością informacji wymagających ochrony takich jak całe numery kart kredytowych, nazwisko panieńskie matki, numery ubezpieczenia społecznego itp.

Studium Przypadku

Studium Przypadku 1: ASVS jako wytyczne dla testowania bezpieczeństwa

Na prywatnym uniwersytecie w stanie Utah, USA, kampusowy zespół Red Team używa OWASP ASVS, jako wytycznych podczas prowadzenia testów penetracyjnych aplikacji. Jest on używany przez cały proces testów penetracyjnych, od inicjujących spotkań poświęconych planowaniu i określeniu zakresu po kierowanie czynnościami testowania, a także jako sposób na nadanie jednolitej struktury dla wyników przedstawianych w końcowym raporcie dla klientów. Red Team organizuje także szkolenia dla zespołu używającego ASVS.

Kampusowy zespół Red Team wykonuje testy penetracyjne sieci oraz aplikacji dla różnych oddziałów kampusu w ramach uniwersyteckiej kompleksowej strategii bezpieczeństwa informacji. Podczas inicjujących spotkań planistycznych, klienci często niechętnie udostępniają ich aplikacje do testowania przez zespół studentów. Wprowadzenie ASVS i wyjaśnienie interesariuszom, że czynności testowania będą prowadzone według tego standardu oraz, że końcowy raport będzie zawierał informację o tym jak aplikacja jest zabezpieczona w odniesieniu do standardu, natychmiast rozwiewa wiele obaw. Następnie ASVS używany jest podczas wyznaczania zakresu w celu określenia ilości czasu i nakładów, jakie należy poświęcić na testowanie. Poprzez wykorzystanie predefiniowanych w ASVS poziomów weryfikacyjnych, Red Team wyjaśnia określane poziomem ryzyka testowanie. Pomaga to klientom, interesariuszom i zespołowi osiągnąć zgodę odnośnie do właściwego zakresu dla przedmiotowej aplikacji.

Po rozpoczęciu testowania, Red Team wykorzystuje ASVS do organizacji działań i podziału prac. Poprzez śledzenie które wymagania weryfikacyjne zostały już przetestowane, a które ciągle trwają, menedżerowie projektu w prosty sposób obserwują postęp testów. Prowadzi to do sprawnej komunikacji z klientami oraz daje menedżerom projektu zdolność do lepszego zarządzania zasobami. Z uwagi na to, iż Red Team składa się głównie ze studentów, większość członków zespołu ma wiele rozmaitych wymagań czasowych wynikających z różnych kursów. Dobrze zdefiniowane zadania, oparte na indywidualnie zweryfikowanych wymaganiach lub całych kategoriach, pomaga członkom zespołu zrozumieć dokładnie co powinno być przetestowane i pozwala im na dokładne oszacowanie długości czasu potrzebnego do realizacji zadania. Także w trakcie raportowania widać korzyści z jasnej organizacji ASVS, ponieważ członkowie zespołu mogą opisywać wyniki zanim przejdą do następnego zadania, efektywnie realizując przygotowywanie raportu jednocześnie z prowadzeniem testu penetracyjnego.

Zespół Red Team organizuje końcowy raport zgodnie z ASVS, raportując status każdego wymagania weryfikacyjnego i dostarczając dodatkowych szczegółów tam, gdzie jest to potrzebne. Daje to klientom oraz interesariuszom dobry pogląd na to jak zabezpieczone są ich aplikacje w stosunku do standardu. Jest to równocześnie bardzo wartościowe dla ich przyszłych działań, ponieważ pozwala uwidocznić poprawę bądź pogorszenie bezpieczeństwa w czasie. Ponadto, interesariusze zainteresowani jak aplikacja realizuje jedną lub wiele określonych kategorię, mogą łatwo odnaleźć tę informację, ponieważ format raportu ściśle przestrzega formatu zalecanego przez ASVS. Jasna organizacja ASVS ułatwia także szkolenie

nowych członków zespołu w zakresie pisania raport, szczególnie w porównaniu z poprzednim formatem raportu.

Ostatecznie, szkolenie zespołu Red Team poprawiło się po wdrożeniu ASVS. Poprzednio, cotygodniowe szkolenia koncentrowały się wokół tematów wybranych przez lidera lub menedżera projektu. Te z kolei wybierane były według wymagań członków zespołu i dostrzeganych potrzeb. Szkolenie oparte na takich kryteriach miało potencjał rozszerzania umiejętności członków zespołu, lecz niekoniecznie pozostawało w związku z działaniami Red Team. Innymi słowy, zespół nie poprawiał się znacząco w zakresie przeprowadzania testów penetracyjnych. Obecnie, po zaadoptowaniu ASVS, szkolenie zespołu koncentruje się na tym jak testować indywidualne wymagania weryfikacyjne. To prowadzi do znaczącej poprawy w wymiernych umiejętnościach poszczególnych członków zespołu oraz poprawy jakości raportu końcowego.

Studium Przypadku 2: ASVS jako bezpieczny SDLC

Start up zamierzający dostarczać analizy typu Big Data dla instytucji finansowych zdaje sobie sprawę z tego, że bezpieczeństwo w fazie rozwoju jest na czele listy wymagań w celu uzyskania dostępu i przetwarzania metadanych finansowych. W tym przypadku start up wybrał ASVS jako podstawę ich bezpiecznego cyklu rozwoju projektowego prowadzonego zgodnie z metodologią „agile".

Start up wykorzystuje ASVS do generacji komponentów biznesowych (ang. epic) oraz przypadków użycia dla funkcjonalnych problemów bezpieczeństwa, takich jak np. sposoby zaimplementowania funkcjonalność logowania. Start up używa ASVS w sposób inny niż większość– przegląda ASVS wybierając wymagania, które pasują do obecnej iteracji (sprint'a). Następnie dodaje je bezpośrednio do rejestru produktu (ang. backlog) dla kolejnej iteracji (sprintu), jeżeli jest to wymaganie funkcjonalne, a w przeciwnym wypadku, jako ograniczenie w stosunku do obecnych przypadków użycia. Na przykład, dodając wybór TOTP (Time-based One-time Password) dwuskładnikowego uwierzytelniania, wraz z politykami hasłowymi oraz kontrolą usług sieciowych podwajamy detekcję ataków „brute force" oraz skuteczność mechanizmu zapobiegawczego. W przyszłych iteracjach dodatkowe wymagania będą wybierane zgodnie z zasadami „dokładnie na czas" lub „tego nie będziesz potrzebował".

Developerzy używają ASVS, jako listy punktów kontrolnych w trakcie wzajemnego przeglądu kodu źródłowego, który zabezpiecza przed wprowadzeniem niebezpiecznego kodu. Jest również stosowany przy planowaniu fazy retrospekcji, aby sprawdzić developerów, którzy wprowadzili nową funkcjonalność i upewnić się, że wzięli oni pod uwagę odpowiednie wymagania ASVS. Pozwala również ustalić, czy cokolwiek mogłoby zostać poprawione lub zredukowane w przyszłych iteracjach.

W końcu, developerzy używają ASVS również w ramach zautomatyzowanych mechanizmów weryfikujących bezpieczeństwo podczas testów jednostkowych i integracyjnych w celu testowania przypadków użycia, nadużyć oraz odporności na błędne dane. Celem jest redukcja ryzyka wynikającego z kaskadowej metodyki zakładającej przeprowadzanie „testów penetracyjnych na koniec projektu", która zazwyczaj skutkuje kosztownym poprawianiem kodu (refaktoringiem) wraz z dostarczaniem kluczowych kompilacji do wersji produkcyjnej.

Ponieważ po każdej iteracji mogą być promowane nowe kompilacje, nie jest wystarczającym by polegać na jednym działaniu zapewniającym jakość. Tym samym poprzez automatyzację reżimu testowania, nawet doświadczeni pentesterzy dysponujący tygodniami czasu na przetestowanie aplikacji nie powinni odnaleźć w niej poważnych problemów.

Ocena, czy oprogramowanie osiągnęło weryfikowany poziom

OWASP nt. certyfikacji ASVS oraz stosowania oznaczeń zgodności

OWASP, jako organizacja non-profit niezależna od dostawców, nie udziela certyfikacji żadnemu z dostawców, audytorów lub systemów informatycznych.

Wszystkie tego rodzaju zapewnienia atestacyjne, wykorzystywane znaki zaufania, lub certyfikacje nie są oficjalnie weryfikowane, rejestrowane, lub certyfikowane przez OWASP. W tym świetle każda organizacja powinna być świadoma zaufania, jakim obdarza jakąkolwiek trzecią stronę lub zaufaną markę deklarującą certyfikację zgodnie z wymaganiami ASVS.

Nie powinno to powstrzymywać organizacji od oferowania tego rodzaju usług atestacyjnych, tak długo jak nie deklarują one oficjalnej certyfikacji OWASP.

Wskazówki dla organizacji certyfikujących

Standard Weryfikacji Bezpieczeństwa Aplikacji może być używany do weryfikacji aplikacji na zasadzie otwartej księgi, obejmującej otwarty i nieograniczony dostęp do kluczowych zasobów, takich jak architekci i developerzy, dokumentacji projektowej, kodu źródłowego oraz uwierzytelnionego dostępu do systemów testowych (zawierającego co najmniej jedno konto na każdą z ról), szczególnie dla poziomów weryfikacji L2 i L3.

Historycznie patrząc, testy penetracyjne i przeglądy kodu pod względem bezpieczeństwa obejmowały jedynie kwestie „wyjątków" – co oznacza, że w końcowym raporcie pojawiały się tylko porażki. Organizacja certyfikująca musi w każdym raporcie zawrzeć zakres weryfikacji (szczególnie, jeśli kluczowy komponent znajduje się poza zakresem, taki jak uwierzytelnienie SSO) oraz podsumowanie wyników weryfikacji, obejmujące testy zakończone sukcesem oraz porażką, z jasnym wskazaniem jak poradzić sobie z testami zakończonymi porażką.

Jako standard branżowy postrzegane jest utrzymywanie szczegółowej dokumentacji pracy, zrzutów ekranu lub nagrań, skryptów pozwalających na wiarygodne i powtarzalne uruchamianie exploit'u, oraz elektronicznych zapisów przebiegu testów, takich jak logi przechwytywania proxy, a także powiązanych zapisów takich jak listy czyszczenia (ang. cleanup list). Może to bowiem być bardzo użyteczne w celu udowodnienia uzyskanych wyników większości deweloperów wątpiących w nie. Niewystarczającym jest po prostu uruchomienie narzędzi i raportowanie porażek, gdyż w ten sposób nie dostarczy się wystarczających dowodów na to, że wszystkie kwestie na poziomie certyfikacji zostały przetestowane i były testowane z należytą starannością. W przypadku sporu, powinny istnieć wystarczające dowody potwierdzające, że każde wymaganie weryfikacyjne rzeczywiście zostało przetestowane.

Rola automatycznych narzędzi do wykonywania testów penetracyjnych

Zachęcamy do wykorzystywania automatycznych narzędzi penetracyjnych, w celu realizacji jak największego zakresu prac oraz sprawdzenia tak wielu parametrów jak jest to możliwe, przy pomocy maksymalnej możliwej ilości różnych form złośliwych ataków.

Przeprowadzenie pełnej weryfikacji ASVS wyłącznie przy pomocy automatycznych narzędzi testów penetracyjnych jest jednakże niemożliwe. Podczas gdy większość wymagań L1 może być realizowana przy pomocy testów automatycznych, całość wymagań nie może zamykać się wyłącznie w automatycznych testach penetracyjnych.

Proszę zauważyć, że granice oddzielające testy automatyczne od ręcznych nie są już wyraźne w związku z dojrzewaniem przemysłu bezpieczeństwa aplikacji. Narzędzia automatyczne są często manualnie dostrajane przez ekspertów, a testerzy manualni często wspierają się szeroką gamą różnorodnych narzędzi automatycznych.

Rola testów penetracyjnych

Możliwe jest przeprowadzenie manualnych testów penetracyjnych i weryfikacja wszystkich kwestii istotnych dla poziomu L1 bez żądania dostępu do kodu źródłowego, aczkolwiek nie jest to rekomendowana praktyka. L2 wymaga pewnego dostępu do developerów, dokumentacji, kodu, oraz uwierzytelnionego dostępu do systemu. Nie jest możliwa weryfikacja wszystkich wymagań poziomu L3 przy pomocy testów penetracyjnych, ponieważ większość dodatkowych kwestii wymaga przeglądu konfiguracji systemu, przeglądu po kątem złośliwego kodu, modelowania zagrożeń, a także innych artefaktów niemożliwych do sprawdzenia podczas testów penetracyjnych.

Jako szczegółowe wytyczne dla architektury bezpieczeństwa

Jednym z najbardziej powszechnych zastosowań ASVS jest wykorzystanie go przez architektów bezpieczeństwa. Dwie główne ramowe architektury bezpieczeństwa: SABSA i TOGAF nie posiadają wystarczającej ilości informacji niezbędnych do kompletnego przeglądu architektury bezpieczeństwa aplikacji. ASVS może być użyty do uzupełnienia tych braków pozwalając architektom bezpieczeństwa na dokonanie wyboru lepszych zabezpieczeń dla powszechnych problemów takich jak wzorce ochrony danych i strategie walidacji danych wejściowych.

Jako substytut dla komercyjnych list kontrolnych dotyczących bezpiecznego kodu

Wiele organizacji może skorzystać na zaadoptowaniu ASVS poprzez wybranie jednego z trzech poziomów lub poprzez rozdzielenie ASVS i wybranie tego co jest wymagane dla każdego poziomu ryzyka aplikacji, w sposób specyficzny dla określonej dziedziny. Zachęcamy do tego typu czynności tak długo jak długo utrzymywane będzie śledzenie i kontrolowanie poczynionych zmian. Przykładowo, jeżeli aplikacja jest zgodna z wymaganiem 4.1, powinno to oznaczać to samo również w rozdzielnych wariantach ewoluującego standardu.

Jako wytyczne dla testów jednostkowych i testów integracyjnych

Standard ASVS jest zaprojektowany tak, aby można go było realizować w dużym stopniu poprzez testy, z jedynym wyjątkiem dotyczącym wymagań dotyczących architektury i złośliwego kodu. Przygotowując testy jednostkowe i testy integracyjne testujących specyficzne i odpowiednie zestawy danych wejściowych (ang. fuzz) oraz przypadki nadużyć (ang. abuse) aplikacja powinny stać się samo-weryfikowalne wraz z każdą kompilacją. Na przykład, można skonstruować dodatkowe testy w ramach zestawu mechanizmów

testowania mechanizmów uwierzytelniających, testujące parametr nazwy użytkownika dla nazw powszechnie występujących, enumeracji kont, ataków typu brute force, wstrzykiwania LDAP i SQL oraz ataków XSS. Podobnie, testowanie parametru hasła powinno uwzględniać powszechnie występujące hasła, długość hasła, wstrzykiwanie bajtów zerowych (ang. null byte), usuwania parametru, ataków XSS, enumeracji kont, itd.

W ramach szkoleń w obszarze bezpiecznego programowania

Standard ASVS może być także wykorzystany do definiowania charakterystyki bezpiecznego oprogramowania. Wiele kursów dotyczących „bezpiecznego kodowania" to po prostu kursy etycznego hacking'u z niewielkim dodatkiem porad dotyczących kodowania. To nie pomaga developerom. Zamiast tego, kursy bezpiecznego programowania mogą wykorzystywać ASVS z silnym naciskiem na proaktywne mechanizmy ASVS, raczej niż zestawu 10 najważniejszych rzeczy, których nie należy wykonywać.

Projekty OWASP wykorzystujące ASVS

Security Knowledge Framework

https://www.owasp.org/index.php/OWASP_Security_Knowledge_Framework

Szkolenie deweloperów z pisania bezpiecznego kodu. SKF działa na zasadzie open-sourcowej aplikacji "Python-Flask web-application", która wykorzystuje OWASP Application Security Verification Standard w celu przeszkolenia Ciebie i Twojego zespołu ze pisania bezpiecznego kodu, w sposób rozmyślny.

OWASP Zed Attack Proxy

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

The OWASP Zed Attack Proxy (ZAP) jest łatwym do wykorzystania zintegrowanym narzędziem wspomagającym wykonywanie testów penetracyjnych, które ma na celu wykrywanie podatności w webaplikacjach. Zostało zaprojektowane tak, aby mogło być wykorzystywane przez osoby z szerokim spektrum doświadczeń w zakresie bezpieczeństwa. Tym samym jest idealne dla deweloperów i testerów funkcjonalnych, którzy dopiero rozpoczynają wykonywanie testów penetracyjnych

OWASP Cornucopia

https://www.owasp.org/index.php/OWASP_Cornucopia

OWASP Cornucopia ma postać gry karcianej, która ma na celu wsparcie zespołów programistycznych w identyfikowaniu wymagań bezpieczeństwa w projektach realizowanych w sposób zwinny (ang. agile), konwencyjny jak również zgodnie z formalnymi metodykami. Jest niezależne od wykorzystywanego języka, platform i technologii. Zestawy Cornucopia zostały wybrane bazując na strukturze OWASP Secure Coding Practices - Quick Reference Guide (SCP), ale dodatkowo brały pod uwagę OWASP Application Security Verification Standard, OWASP Testing Guide oraz David Rook’s Principles of Secure Development.

Szczegółowe wymagania dla weryfikacji

V1. Wymagania weryfikacji dla architektury, projektowania i modelowania zagrożeń

V2. Wymagania weryfikacji dla uwierzytelniania

V3. Wymagania weryfikacji dla zarządzania sesją

V4. Wymagania weryfikacji dla kontroli dostępu

V5. Wymagania weryfikacji dla obsługi złośliwych danych wejściowych

V7. Wymagania weryfikacji dla nieaktywnych mechanizmów kryptograficznych

V8. Wymagania weryfikacji dla obsługi i logowania błędów

V9. Wymagania weryfikacji dla mechanizmów ochrony danych

V10. Wymagania weryfikacji dla zabezpieczania komunikacji

V11. Wymagania weryfikacji dla konfigurowania bezpieczeństwa HTTP

V13. Wymagania weryfikacji zabezpieczeń przed złośliwym kodem

V15. Wymagania weryfikacji logiki biznesowej

V16. Wymagania weryfikacji dla plików i innych zasobów

V17. Wymagania weryfikacji dla aplikacji mobilnych

V18. Wymagania weryfikacyjne dla web serwisów (dodane w wersji 3.0)

V19. Wymagania weryfikacyjne dla procesu konfiguracji (dodane w wersji w 3.0)

V1: Wymagania weryfikacji dla architektury, projektowania i modelowania zagrożeń

Cele kontrolne

Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- dla poziomu 1, wszystkie komponenty aplikacji są zidentyfikowane i istnieje powód dla których zostały tam umieszczone;
- dla poziomu 2, architektura została zdefiniowana a kod jest zgodny z architekturą;
- dla poziomu trzeciego, istnieje zdefiniowana architektura oraz dokumentacja projektowa i jest ona wykorzystywana w sposób skuteczny.

Wymagania

#	Opis	1	2	3	Od
1.1	Zweryfikuj, czy wszystkie komponenty aplikacji są zidentyfikowane, są znane i są rzeczywiście potrzebne.	x	x	x	1.0
1.2	Zweryfikuj, czy wszystkie komponenty są zidentyfikowane takie jak biblioteki, moduły i zewnętrzne systemy, które nie są częścią aplikacji ale są wymagane do jej działania.		x	x	1.0
1.3	Zweryfikuj, czy została zdefiniowana wysoko-poziomowa architektura aplikacji.		x	x	1.0
1.4	Zweryfikuj, czy komponenty aplikacji są zdefiniowane w kontekście funkcji biznesowych oraz funkcji bezpieczeństwa, które dostarczają.			x	1.0
1.5	Zweryfikuj, czy wszystkie komponenty niebędące częścią aplikacji, choć będące przez nią wykorzystywane, są zdefiniowane pod kątem funkcjonalności - w tym funkcjonalności bezpieczeństwa, które dostarczają.			x	1.0
1.6	Zweryfikuj, czy model oceny ryzyka dla aplikacji został utworzony i czy pokrywa wszystkie ryzyka w ramach STRIDE (Podszywanie się - Spoofing, Modyfikacja danych - Tampering, Zaprzeczalność - Repudiation, Wyciek informacji - Information Disclosure i Eskalacja uprawnień - Privileges Escalation).			x	1.0

1.7	Zweryfikuj, czy wszystkie zabezpieczenia (włącznie z bibliotekami komunikującymi się z zewnętrznymi usługami bezpieczeństwa) są wdrażane w sposób scentralizowany.		x	1.0
1.8	Zweryfikuj, czy komponenty są odseparowane od siebie poprzez zdefiniowane zabezpieczenia, takie jak segmentacja sieci, reguły zapory sieciowej czy grupy dostępu w usługach chmurowych.	x	x	3.0
1.9	Zweryfikuj, czy aplikacja ma jasno zdefiniowany rozdział pomiędzy warstwą danych, warstwą kontrolera i warstwą prezentacyjną, tak aby decyzje dotyczące bezpieczeństwa były wymuszane na zaufanych systemach.	x	x	3.0
1.10	Zweryfikuj, czy w kodzie aplikacji klienckiej nie ma wrażliwych informacji z zakresu logiki biznesowej, sekretnych kluczy i innych zastrzeżonych informacji.	x	x	3.0
1.11	Zweryfikuj czy wszystkie komponenty aplikacji, biblioteki, moduły, struktura ramowa, platformy i systemy operacyjne są wolne od znanych podatności.	x	x	3.0.1

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- Threat Modeling Cheat Sheet

  https://www.owasp.org/index.php/Application_Security_Architecture_Cheat_Sheet
- Attack Surface Analysis Cheat Sheet:

  https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet

V2. Wymagania weryfikacji dla uwierzytelniania

Cele kontrolne

Uwierzytelnianie to akt ustanowienia lub potwierdzenia, że coś (ktoś) jest autentyczne i że potwierdzenia dokonane przez lub dla tej rzeczy są prawdziwe. Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- weryfikuje cyfrową tożsamość nadawcy w trakcie komunikacji;
- zapewnia, że tylko upoważnione podmioty mogą się uwierzytelnić, a dane uwierzytelniające są transportowane w sposób bezpieczny.

Wymagania

#	Opis	1	2	3	Od
2.1	Zweryfikuj, czy wszystkie strony oraz zasoby standardowo wymagają uwierzytelnienia, za wyjątkiem tych, które mają być dostępne publicznie (zasada pełnej mediacji).	x	x	x	1.0
2.2	Zweryfikuj, czy pola zawierające dane uwierzytelniające nie są uzupełniane przez aplikację. Propozycje uzupełniania realizowane przez aplikację oznacza, że są one przechowywane czystym tekstem lub w odwracalnym formacie, co jest zabronione.	x	x	x	3.0.1
2.4	Zweryfikuj, czy wszystkie mechanizmy uwierzytelniania są wymuszane po stronie serwera.	x	x	x	1.0
2.6	Zweryfikuj, czy wszystkie mechanizmy uwierzytelniania, które kończą pracę niepowodzeniem, robią to w sposób bezpieczny, aby mieć pewność, że atakujący nie może się zalogować.	x	x	x	1.0
2.7	Zweryfikuj, czy pole dla hasła zezwala lub sprzyja używaniu haseł (passphrase) i nie ma wprowadzonych ograniczeń chroniących przed korzystaniem z menedżerów haseł oraz wpisywaniem długich lub bardzo złożonych haseł.	x	x	x	3.0.1
2.8	Zweryfikuj, czy wszystkie operacje dotyczące uwierzytelniania (takie jak rejestracja, aktualizacja profilu, przypomnienie loginu, przypomnienie hasła, unieważnienie zgubionego kodu do telefonicznej obsługi klienta lub automatycznej obsługi klienta), które powodują odzyskanie dostępu do konta, posiadają przynajmniej takie same zabezpieczenia jak podstawowe mechanizmy	x	x	x	2.0

#	Opis	1	2	3	Od
	uwierzytelniania.				
2.9	Zweryfikuj, czy funkcja zmiany hasła zawiera konieczność podania hasła obecnie używanego, nowego hasła oraz konieczność ponownego wpisania nowego hasła.	x	x	x	1.0
2.12	Zweryfikuj, czy wszystkie decyzje dotyczące uwierzytelnienia mogą być logowane bez przechowywania poufnych identyfikatorów sesji lub haseł. Powinno to zawierać zapytania do odpowiednich metadanych niezbędnych do ewentualnego postępowania wyjaśniającego w zakresie bezpieczeństwa.		x	x	3.0.1
2.13	Zweryfikuj, czy hasła do kont są przechowywane wykorzystując jednokierunkową funkcję skrótu z solą i czy są wystarczająco skomplikowane by się obronić przed atakami brute force oraz odtwarzaniem hasła z jego skrótu.		x	x	3.0.1
2.16	Zweryfikuj, czy dane uwierzytelniające są przesyłane wykorzystując właściwy zaszyfrowany link oraz, czy wszystkie strony/funkcje które wymagają by użytkownik podał dane uwierzytelniające wymagają takiego szyfrowanego linku.	x	x	x	3.0
2.17	Zweryfikuj, czy mechanizmy odzyskiwania hasła nie ujawniają dotychczasowych haseł i czy nowe hasło nie jest przesyłane w formie jawnej do użytkownika.	x	x	x	2.0
2.18	Zweryfikuj, czy nie jest możliwa enumeracja wykorzystująca loginy użytkowników, funkcję resetowania hasła lub funkcjonalność przypomnienia nazwy użytkownika.	x	x	x	2.0
2.19	Zweryfikuj, czy platforma lub inne komponenty z których korzysta aplikacja nie używają domyślnych haseł (np. admin/password).	x	x	x	2.0
2.20	Zweryfikuj, czy wdrożono mechanizmy przeciwdziałające automatyzacji, aby zapobiegać testowaniu ujawnionych danych uwierzytelniających, atakom brute force i atakom blokującym konta.	x	x	x	3.0.1
2.21	Zweryfikuj, czy wszystkie dane uwierzytelniające służące do uzyskiwania dostępu do usług zewnętrznych względem aplikacji, są zaszyfrowane i przechowywane w zabezpieczonej lokalizacji (a nie w kodzie źródłowym).		x	x	2.0

#	Opis	1	2	3	Od
2.22	Zweryfikuj, czy funkcje odzyskiwania hasła i inne ścieżki odzyskiwania poświadczeń uwzględniają użycie tokenu udostępniającego jednorazowe, zmienne w czasie hasła (Time-based One-Time Password algorithm - TOTP) lub innego tokena programowego, mechanizmu push lub mechanizmu odzyskiwania offline. Wykorzystywanie losowych wartości przesyłanych pocztą elektroniczną lub SMS powinno być jedynie ostatecznością, gdyż dowiedziono ich słabości.	x	x	x	3.0.1
2.23	Zweryfikuj że status blokowania konta uwzględnia mechanizm "miękkiej" i "twardej" blokady, przy czym te statusy nie wykluczają się nawzajem. Jeżeli konto jest zablokowane w sposób "miękki" ze względu na atak brute force, nie powinno to resetować statusu "twardej" blokady.		x	x	3.0
2.24	Zweryfikuje, czy w przypadku wykorzystywania współdzielonych sekretnych pytań, nie naruszają one przepisów dotyczących prywatności i są na tyle trudne by w rzeczywistości chronić aplikację.	x	x	x	3.0.1
2.25	Zweryfikuj, czy system umożliwia zablokowanie ponownego wykorzystania określonej liczby haseł, poprzednio użytych przez użytkownika.		x	x	2.0
2.26	Zweryfikuj, czy powtórne uwierzytelnianie, dwuskładnikowe uwierzytelnianie lub podpisywanie danych transakcyjnych jest stosowane w przypadku transakcji wysokiej wartości.		x	x	3.0.1
2.27	Zweryfikuj, czy wdrożono mechanizmy blokujące użycie znanych lub słabych haseł.	x	x	x	3.0
2.28	Zweryfikuj, czy wszystkie próby uwierzytelnienia, niezależnie czy zakończone powodzeniem czy niepowodzeniem, odpowiadają w tym samym średnim czasie.			x	3.0
2.29	Zweryfikuj, czy hasła, klucze bądź inne informacje uwierzytelniające nie są zapisane w kodzie źródłowym lub repozytoriach kodu.			x	3.0
2.30	Zweryfikuj, czy aplikacja pozwala użytkownikom na uwierzytelnienie przy wykorzystaniu uprawnionego bezpiecznego mechanizmu uwierzytelniania.	x	x	x	3.0
2.31	Zweryfikuj, czy jeżeli aplikacja pozwala użytkownikom na uwierzytelnienie, mogą oni to zrobić wykorzystując dwa		x	x	3.0

#	Opis	1	2	3	Od
	czynniki lub inne silne metody uwierzytelniania albo też wykorzystując inny podobny sposób postępowania, który chroni przed ujawnieniem nazwy wraz z hasłem.				
2.32	Zweryfikuj, czy interfejs administracyjny nie jest dostępny dla stron niezaufanych.	x	x	x	3.0
2.33	Funkcja autouzupełniania w przeglądarce lub integracja z menedżerami haseł powinny być dopuszczone, chyba że są zabronione na podstawie przeprowadzonej analizy ryzyka.	x	x	x	3.0.1

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Testing for Authentication
  https://www.owasp.org/index.php/Testing_for_authentication
- Password storage cheat sheet
  https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet
- Forgot password cheat sheet
  https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet
- Choosing and Using Security Questions
  https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet

V3. Wymagania weryfikacji dla zarządzania sesją

Cele kontrolne

Jednym z głównych komponentów każdej webaplikacji jest mechanizm, przy pomocy którego nadzorowany i utrzymywany jest jej stan, w trakcie interakcji z użytkownikiem. Jest to określane jako zarządzanie sesją i jest definiowane jako zestaw narzędzi nadzorujących stanowo (state-full) interakcje pomiędzy użytkownikiem i webaplikacją. Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- sesje są unikalne dla każdego użytkownika i nie mogą zostać odgadnięte lub współdzielone;
- sesje są unieważniane, gdy tylko przestają być niezbędne oraz przerywane, gdy nie są wykorzystywane.

Wymagania

#	Opis	1	2	3	Od
3.1	Zweryfikuj, czy aplikacja nie używa zmodyfikowanego managera sesji lub, jeśli jest on używany, to czy jest odporny na wszystkie powszechne ataki na zarządzanie sesją.	x	x	x	1.0
3.2	Zweryfikuj, czy sesje są unieważniane po wylogowaniu się użytkownika.	x	x	x	1.0
3.3	Zweryfikuj, czy sesje wygasają po określonym czasie bezczynności.	x	x	x	1.0
3.4	Zweryfikuj, czy sesje wygasają po administracyjnie konfigurowalnym maksymalnym okresie, niezależnie od aktywności (bezwzględny okres ważności).			x	1.0
3.5	Zweryfikuj, czy wszystkie strony, do których dostęp wymaga uwierzytelnienia, zawierają łatwy i widoczny dostęp do funkcji wylogowania.	x	x	x	1.0
3.6	Zweryfikuj, czy identyfikatory sesji nigdy nie są ujawniane w URL, w komunikatach błędów lub logach. Należy również zweryfikować, czy aplikacja nie wspiera przepisywania w URL ciasteczek sesyjnych (ang. URL-rewriting).	x	x	x	1.0
3.7	Zweryfikuj, czy każde pomyślne uwierzytelnienie a także ponowne uwierzytelnienie, tworzy nową sesję z nowym identyfikatorem.	x	x	x	1.0

3.10	Zweryfikuj, czy tylko identyfikatory sesji wygenerowane przez platformę aplikacji są uznawane przez nią za aktywne.		x	1.0	1.0
3.11	Zweryfikuj, czy identyfikatory sesji są wystarczająco długie, losowe i unikalne w obrębie odpowiedniej aktywnej bazy sesji.		x	x	1.0
3.12	Zweryfikuj, czy identyfikatory sesji przechowywane w ciasteczkach mają ustawioną ścieżkę z wartością odpowiednio restrykcyjną dla danej aplikacji i czy tokeny sesji uwierzytelniania mają dodatkowo ustawione atrybuty "HttpOnly" i "secure".	x	x	x	3.0
3.16	Zweryfikuj, czy aplikacja ogranicza liczbę jednoczesnych aktywnych sesji.	x	x	x	3.0
3.17	Zweryfikuj, czy lista aktywnych sesji jest wyświetlana dla każdego użytkownika w profilu konta lub podobnym. Użytkownik powinien posiadać możliwość zakończenia dowolnej aktywnej sesji.	x	x	x	3.0
3.18	Zweryfikuj, czy użytkownik ma zaproponowaną opcję zakończenia wszystkich innych aktywnych sesji, po pomyślnym zakończeniu procesu zmiany hasła.	x	x	x	3.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Session Management Testing
  https://www.owasp.org/index.php/Testing_for_Session_Management
- OWASP Session Management Cheat Sheet:
  https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

V4. Wymagania weryfikacji dla kontroli dostępu

Cele kontrolne

Uwierzytelnianie jest koncepcją zapewniającą dostęp jedynie do tych zasobów, na które wyrażono zgodę. Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- osoby uzyskujące dostęp posiadają ważne dane uwierzytelniające;
- użytkownicy są powiązani z dobrze zdefiniowanymi zestawami ról i uprawnień;
- role i uprawnienia są chronione przed ponownym wykorzystaniem lub modyfikacją.

Wymagania

#	Opis	1	2	3	Od
4.1	Zweryfikuj, czy została wdrożona zasada minimalnych uprawnień - użytkownicy powinni mieć dostęp tylko do funkcji, plików, linków URL, usług oraz innych zasobów do których posiadają zezwolenie.	x	x	x	1.0
4.4	Zweryfikuj, czy dostęp do wrażliwych zapisów jest chroniony w taki sposób, aby tylko autoryzowane obiekty lub dane były dostępne dla użytkownika (dla przykładu chroń przed ingerowaniem użytkowników w parametry umożliwiające zobaczenie lub dokonywanie zmian na koncie innego użytkownika).	x	x	x	1.0
4.5	Zweryfikuj, czy listowanie katalogów jest wyłączone (chyba, że jest celowo dozwolone). Dodatkowo aplikacje nie powinny pozwalać na odkrycie lub ujawnienie plików lub metadanych katalogów takich jak Thumbs.db, .DS_Store, .git lub foldery .svn.	x	x	x	1.0
4.8	Zweryfikuj, czy mechanizmy kontroli dostępu, które kończą pracę niepowodzeniem, robią to w sposób bezpieczny.	x	x	x	1.0
4.9	Zweryfikuj, czy te same reguły kontroli dostępu występujące w warstwie prezentacji są również egzekwowane po stronie serwera.	x	x	x	1.0
4.10	Zweryfikuj, czy wszystkie atrybuty użytkowników i danych oraz informacje o zasadach dostępu wykorzystywane przez mechanizmy kontroli dostępu, nie mogą być modyfikowane przez użytkowników, chyba, że są oni do tego uprawnieni.		x	x	1.0

4.11	Zweryfikuj, czy istnieje scentralizowany mechanizm zabezpieczający dostęp do każdego typu chronionych zasobów (również dla bibliotek wywołujących zewnętrzne usługi autoryzacji).			x	1.0
4.12	Zweryfikuj, czy wszystkie decyzje dotyczące kontroli dostępu mogą być logowane, a wszystkie decyzje zakończone niepowodzeniem są logowane.		x	x	2.0
4.13	Zweryfikuj, czy aplikacja bądź platforma generuje silne, losowe tokeny zabezpieczające przed atakami typu CSRF lub posiadają inne mechanizmy ochrony transakcji.	x	x	x	2.0
4.14	Zweryfikuj, czy system może się ochronić się przed masowym lub ciągłym dostępem do zabezpieczonych funkcji, zasobów bądź danych. Na przykład należy rozważyć wykorzystanie zarządcy zasobów w celu ograniczenia liczby edycji na godzinę lub zapobiec odczytaniu wszystkich danych przez jednego użytkownika w sposób zautomatyzowany (tzw. „scrapping”).		x	x	2.0
4.15	Zweryfikuj, czy aplikacja wykorzystuje dodatkowe uwierzytelnianie (takie jak uwierzytelnianie wieloetapowe bądź biorące pod uwagę profil ryzyka danego użytkownika) dla mniej ważnych systemów i/lub segregację obowiązków dla ważniejszych systemów, aby wymuszać mechanizmy chroniące przed defraudacjami, które są wdrażane bazując na zidentyfikowanym ryzyku aplikacji i dawnych przypadkach defraudacji.		x	x	3.0
4.16	Zweryfikuj, czy aplikacja w poprawny sposób wymusza autoryzację zależną od kontekstu, tak aby nie umożliwiać nieautoryzowanych manipulacji realizowanych poprzez zmiany wartości parametrów.	x	x	x	3.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Authorization
  https://www.owasp.org/index.php/Testing_for_Authorization
- OWASP Cheat Sheet: Access Control
  https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

V5. Wymagania weryfikacji dla obsługi złośliwych danych wejściowych

Cele kontrolne

Najczęstszą słabością spotykaną w webaplikacjach jest niepowodzenie w trakcie poprawnej walidacji danych wejściowych pochodzących od klientów lub z samych środowisk, zanim zostały one użyte. To prowadzi do praktycznie wszystkich najpoważniejszych podatności w aplikacjach webowych , takich jak cross site scripting, SQL injection, interpreter injection, ataki na locale/Unicode, ataki na system plików, przepełnienie bufora. Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- wszystkie dane wejściowe są walidowane aby zapewnić, że są poprawne i dostosowane do zamierzonych celów;
- dane z zewnętrznych źródeł lub od klientów nigdy nie powinny być traktowane jako zaufane i powinny być odpowiednio traktowane.

Wymagania

#	Opis	1	2	3	Od
5.1	Zweryfikuj, czy środowisko uruchomieniowe nie jest podatne na przepełnienia bufora lub czy mechanizmy bezpieczeństwa zapobiegają wystąpieniu przepełnienia bufora.	x	x	x	1.0
5.3	Zweryfikuj, czy wszystkie prowadzone po stronie serwera walidacje wejścia, które zakończone niepowodzeniem, powodują odrzucenie żądania oraz są logowane.	x	x	x	1.0
5.5	Zweryfikuj, czy wszystkie mechanizmy walidacji są egzekwowane po stronie serwera.	x	x	x	1.0
5.6	Zweryfikuj, czy aplikacja wykorzystuje jeden mechanizm walidacji danych wejściowych dla każdego typu przyjmowanych danych.			x	1.0
5.10	Zweryfikuj, czy wszystkie kwerendy SQL, HQL, OSQL, NOSQL, składowane procedury, wywołania do składowanych procedur są chronione poprzez wykorzystywanie przygotowanych zapytań lub sparametryzowanych kwerend, i tym samym nie są podatne na atak SQL Injection.	x	x	x	2.0
5.11	Zweryfikuj, czy aplikacja nie jest podatna na atak LDAP Injection lub mechanizmy bezpieczeństwa zapobiegają	x	x	x	2.0

	wystąpieniu takiego ataku.				
5.12	Zweryfikuj, czy środowisko uruchomieniowe nie jest podatne na atak wstrzyknięcia komend systemu operacyjnego lub mechanizmy bezpieczeństwa zapobiegają wystąpieniu takiego ataku.	x	x	x	2.0
5.13	Zweryfikuj, czy aplikacja nie jest podatna na ataki zdalnego lub lokalnego dołączenia plików (Remote File Inclusion - RFI lub Local File Inclusion - LFI) gdy wykorzystywana jest treść będąca ścieżką do plików.	x	x	x	3.0
5.14	Zweryfikuj, czy aplikacja nie jest podatna na ataki XML Injection, XML External Entity, XPath query lub mechanizmy bezpieczeństwa zapobiegają wystąpieniu takich ataków.	x	x	x	2.0
5.15	Upewnij się, że wszystkie łańcuchy zmiennych włączane w HTML lub inny kod uruchamiany po stronie klienta webowego są albo w poprawny sposób ręcznie wprowadzone w sposób zależny od kontekstu, albo wykorzystują szablony, wprowadzając je automatycznie w sposób zależny od kontekstu aby zapewnić, że aplikacja nie jest podatna na ataki XSS typu „reflected", „stored" i DOM.	x	x	x	2.0
5.16	Jeżeli platforma aplikacji pozwala na masowe przypisywanie parametrów (mass assignment), zwane także automatycznym wiązaniem zmiennych (automatic variable binding) z przychodzącego żądania do modelu, zweryfikuj, czy istotne z punktu widzenia bezpieczeństwa pola (np. takie jak "stan_konta", "rola", "password") są chronione przed złośliwym automatycznym wiązaniem.		x	x	2.0
5.17	Zweryfikuj, czy system jest zabezpieczony przed atakami typu HTTP Parametr Pollution, szczególnie jeżeli platforma aplikacji nie rozróżnia źródła pochodzenia parametrów żądania (GET, POST, nagłówki, ciasteczka, środowisko, itd.)		x	x	2.0
5.18	Zweryfikuj, czy walidacja po stronie klienta jest wykorzystywana jako druga linia obrony - będąca uzupełnieniem walidacji wykonywanej po stronie serwera.		x	x	3.0
5.19	Zweryfikuj, czy wszystkie dane wejściowe są walidowane. Dotyczy to nie tylko pól formularzy HTML, ale wszystkich źródeł wejścia takich jak wywołania REST, parametry zapytań, nagłówki HTTP, ciasteczka, pliki wsadowe, kanały RSS itd. Należy używać walidacji pozytywnej (whitelisting), następnie eliminować znane "złe" ciągi znaków (greylisting)		x	x	3.0

	lub odrzucać znane "złe" wejścia (blacklisting).				
5.20	Zweryfikuj, czy ustrukturyzowane dane są silnie zdefiniowane i walidowane względem schematu uwzględniając: dopuszczone znaki, długość i zgodność ze wzorcem (np. numer karty kredytowej, numer telefonu lub walidację, czy relacja dwóch pól wejściowych jest odpowiednia - np. czy podana lokalizacja jest zgodna z podanym kodem pocztowym).		x	x	3.0
5.21	Zweryfikuj, czy niestrukturalne dane są wyczyszczone i zawierają dozwolone znaki oraz długość, a także czy wszystkie znaki potencjalnie szkodliwe dla danego kontekstu powinny zostać usunięte (np. nazwy lokalne pisane w Unicode oraz apostrofy, jak w słowach ねこ czy O'Hara).		x	x	3.0
5.22	Zweryfikuj, czy niezaufany kod HTML z edytorów WYSIWYG lub podobnych jest wyczyszczony oraz obsługiwany odpowiednio w zakresie walidacji wejścia i enkodowania wyjścia.	x	x	x	3.0
5.23	W przypadku korzystania z technologii szablonów z funkcją automatycznego cytowania, gdy wyłączone jest cytowanie przez UI, zapewnij należyte czyszczenie kodu HTML.		x	x	3.0
5.24	Zweryfikuj, czy dane przenoszone pomiędzy kontekstami DOM używają bezpiecznych metod JavaScript np. .innerText i .val.		x	x	3.0
5.25	Zweryfikuj, czy przy parsowaniu JSON w przeglądarce, metoda JSON.parse jest wykorzystywana do parsowania JSON na kliencie. Nie wykorzystuj eval() parsowania JSON po stronie klienta.		x	x	3.0
5.26	Zweryfikuj, czy dane zapisywane po uwierzytelnieniu w pamięci przeglądarki np. w DOM są czyszczone po zakończeniu sesji.		x	x	3.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Input Validation Testing https://www.owasp.org/index.php/Testing_for_Input_Validation
- OWASP Cheat Sheet: Input Validation https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet

- OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution https://www.owasp.org/index.php/Testing_for_HTTP_Parameter_pollution_%28OTG-INPVAL-004%29
- OWASP LDAP Injection Cheat Sheet https://www.owasp.org/index.php/LDAP_Injection_Prevention_Cheat_Sheet
- OWASP Testing Guide 4.0: Client Side Testing https://www.owasp.org/index.php/Client_Side_Testing
- OWASP Cross Site Scripting Prevention Cheat Sheet https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet
- OWASP Java Encoding Project https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

Aby dowiedzieć się więcej o automatycznym cytowaniu odwiedź:

- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems http://googleonlinesecurity.blogspot.com/2009/03/reducing-xss-by-way-ofautomatic.html
- AngularJS Strict Contextual Escaping https://docs.angularjs.org/api/ng/service/$sce https://cwe.mitre.org/data/definitions/915.html

V6: Output encoding / escaping

Ta sekcja została włączona w skład V5. Wymagania weryfikacji dla obsługi złośliwych danych wejściowych w wersji 2.0 Application Security Verification Standard.

V7. Wymagania weryfikacji dla nieaktywnych mechanizmów kryptograficznych

Cele kontrolne

Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- wszystkie moduły kryptograficzne kończące pracę niepowiedzeniem robią to w sposób bezpieczny;
- w przypadku gdy wymagana jest losowość, jest wykorzystywany odpowiedni generator liczb losowych;
- dostęp do kluczy jest zarządzany w bezpieczny sposób.

Wymagania

#	Opis	1	2	3	Od
7.2	Zweryfikuj, czy wszystkie moduły kryptograficzne, które kończą pracę niepowodzeniem robią to w sposób bezpieczny a błędy są obsługiwane w sposób zapobiegający atakom oracle padding.	x	x	x	1.0
7.6	Zweryfikuj, czy wszystkie liczby losowe, losowe nazwy plików, losowe GUID'y oraz losowe ciągi znaków, których atakujący nie powinien odgadnąć, generowane są z wykorzystaniem zatwierdzonego generatora liczb losowych z modułu kryptograficznego.		x	x	1.0
7.7	Zweryfikuj, czy wszystkie moduły kryptograficzne wykorzystywane przez aplikację zostały sprawdzone na zgodność ze standardem FISP 140-2 lub równorzędnym.	x	x	x	1.0
7.8	Zweryfikuj, czy moduły kryptograficzne pracują w zaakceptowanym trybie, zgodnym dla opublikowanych dla nich polityk bezpieczeństwa.			x	1.0
7.9	Zweryfikuj, czy istnieje jasna polityka określająca jak są zarządzane klucze kryptograficzne (np. jak są generowane, rozprowadzane, unieważniane, w jaki sposób wygasają). Zweryfikuj, czy ta polityka jest właściwie egzekwowana.		x	x	1.0
7.11	Zweryfikuj, czy użytkownicy usług kryptograficznych nie mają bezpośredniego dostępu do kluczowych materiałów. Izoluj procesy kryptograficzne, włączając w to klucz główny,			x	3.0.1

	i rozważ wykorzystanie zwirtualizowanego lub sprzętowego modułu zabezpieczeń (HSM).			
7.12	Dane osobowe powinny być przechowywane w postaci zaszyfrowanej i należy zapewnić aby komunikacja odbywała się za pomocą zabezpieczonych kanałów.	x	x	3.0
7.13	Zweryfikuj, czy hasła wymagające ochrony oraz klucze znajdujące się w pamięci, są nadpisywane zerami gdy tylko przestają być wymagane, aby zabezpieczyć się przed atakami zrzucania pamięci (memory dumping).	x	x	3.0.1
7.14	Zweryfikuj, czy wszystkie klucze i hasła mogą być wymieniane oraz, że są generowane lub zmieniane podczas instalacji.	x	x	3.0
7.15	Zweryfikuj, czy entropia generowanych liczb losowych jest wystarczająca, nawet gdy aplikacja jest mocno obciążona, ew. aplikacja powinna w adekwatny sposób reagować na zmniejszenie się puli entropii.		x	3.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Testing for weak Cryptography
  https://www.owasp.org/index.php/Testing_for_weak_Cryptography
- OWASP Cheat Sheet: Cryptographic Storage
  https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

V8. Wymagania weryfikacji dla obsługi i logowania błędów

Cele kontrolne

Głównym celem obsługi i logowania błędów jest zapewnienie reakcji przydatnej użytkownikom, administratorom i zespołom reagowania na incydenty. Celem nie jest masowe tworzenie logów, lecz logów wysokiej jakości, niosących więcej treści niż odrzucany szum. Logi wysokiej jakości często będą zawierać dane wymagające ochrony i dlatego muszą być chronione zgodnie z lokalnymi przepisami oraz dyrektywami dotyczącymi prywatności. Powinno o obejmować następujące zasady:

- nie gromadzenie lub logowanie informacji wymagających ochrony, jeżeli nie jest to niezbędnie wymagane;
- zapewnienie, że wszystkie logowane informacje są obsługiwane w sposób bezpieczny i chronione zgodnie z klasyfikacją tych danych;
- zapewnienie, że logi nie są przetrzymywane nieustannie, lecz mają zdefiniowaną ważność na okres tak krótki jak to możliwe.

Jeżeli logi zawierają dane osobowe lub informacje podlegające ochronie (których definicje zmieniają się w poszczególnych państwach), to stają się jednymi z najbardziej wrażliwych informacji przetwarzanymi w aplikacji i tym samym stają się atrakcyjnym celem dla atakujących.

Wymagania

#	Opis	1	2	3	Od
8.1	Zweryfikuj, czy aplikacja nie zwraca komunikatów o błędach lub śladów stosu (stack traces), które zawierają dane wrażliwe i które mogą pomóc atakującym. Zawierają się w tym identyfikatory sesji, wersje oprogramowania/ platformy i dane osobowe.	x	x	x	1.0
8.2	Zweryfikuj, czy logika zarządzania błędami w mechanizmach bezpieczeństwa domyślnie zabrania do nich dostępu.		x	x	1.0
8.3	Zweryfikuj, czy mechanizmy logowania zdarzeń bezpieczeństwa zapewniają możliwość rejestracji zdarzeń istotnych z punktu widzenia bezpieczeństwa zarówno zakończonych sukcesem jak i porażką.		x	x	1.0
8.4	Zweryfikuj, czy log każdego zdarzenia zawiera niezbędną informację, która pozwoli na precyzyjną analizę czasową w		x	x	1.0

	przypadku wystąpienia zdarzenia.				
8.5	Zweryfikuj, czy wszystkie zdarzenia zawierające niezaufane dane nie zostaną uruchomione jako kod w oprogramowaniu służącym do przeglądania logów.			x	1.0
8.6	Zweryfikuj, czy logi bezpieczeństwa są chronione przed nieautoryzowanym dostępem i modyfikacją.		x	x	1.0
8.7	Zweryfikuj, czy aplikacja nie loguje wrażliwych danych zdefiniowanych zgodnie z lokalnymi regulacjami lub polityką prywatności, danych wrażliwych z punktu widzenia organizacji zdefiniowanych w ramach szacowania ryzyka, lub wrażliwych danych uwierzytelniających, które mogłyby pomóc atakującemu, włączając w to identyfikatory sesji, hasła, ciągi hash lub tokeny API.		x	x	3.0
8.8	Zweryfikuj, że znaki niedrukowalne oraz separatory pól są prawidłowo zakodowane w rejestrach logów, w sposób zapobiegający wstrzykiwaniu logów.			x	2.0
8.9	Zweryfikuj, że pola logów pochodzące ze źródeł zaufanych i niezaufanych są rozróżnialne we wpisach logów.			x	2.0
8.10	Zweryfikuj, że logi audytowe lub podobne rejestry umożliwiają niezaprzeczalność kluczowych transakcji.		x	x	3.0
8.11	Zweryfikuj, czy logi bezpieczeństwa posiadają jakąś formę kontroli integralności lub mechanizmy zapobiegające nieautoryzowanej modyfikacji.			x	3.0
8.12	Zweryfikuj, czy logi są przechowywane w innej partycji niż ta, w której uruchomiona jest aplikacja, z zapewnieniem odpowiedniej rotacji logów.			x	3.0
8.13	Źródła czasu powinny być synchronizowane aby zapewnić, że logi mają poprawny czas.	x	x	x	3.0.1

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0 content: Testing for Error Handling https://www.owasp.org/index.php/Testing_for_Error_Handling

V9. Wymagania weryfikacji dla mechanizmów ochrony danych

Cele kontrolne

Istnieją trzy elementy aby należycie chronić dane: poufność, integralność i dostępność. Zakłada to, że ochrona danych jest wymuszana na zaufanych systemach, takich jak serwery, które zostały utwardzone i posiadają wystarczające zabezpieczenia. Aplikacje powinny jednak zakładać, że zabezpieczenia wszystkich urządzeń są w jakimś stopniu skompromitowane. Gdy aplikacja przesyła lub przechowuje dane wymagające ochrony na niebezpiecznych urządzeniach, jakich jak współdzielone komputery, telefony i tablety, to aplikacja jest odpowiedzialna za to, że dane te będą zaszyfrowane i nie będzie można ich w łatwy sposób ich zdobyć, zmienić lub ujawnić. Zależy więc zapewnić, że weryfikowana aplikacja w sposób satysfakcjonujący przestrzega następujących ogólnych wymagań bezpieczeństwa:

- poufność: dane powinny być chronione przed nieautoryzowanym podglądem lub ujawnieniem, zarówno podczas transmisji jak i podczas przechowywania;
- integralność: dane powinny być chronione przed złośliwym tworzeniem, zmianą lub kasowaniem przez nieupoważnionych atakujących;
- dostępność: dane powinny być dostępne dla autoryzowanych użytkowników gdy tylko są potrzebne.

Wymagania

#	Opis	1	2	3	Od
9.1	Zweryfikuj, czy dla wszystkich formularzy zawierających dane wrażliwe, wyłączone jest cachowanie po stronie klienta, włączając w to funkcjonalność automatycznego wypełnienia pól.	x	x	x	1.0
9.2	Zweryfikuj, czy zidentyfikowano listę danych wrażliwych przetwarzanych przez aplikację i czy istnieje jasna polityka określająca jak dostęp do tych danych powinien być kontrolowany, szyfrowany i przyznawany zgodnie z odpowiednim prawodawstwem w zakresie ochrony danych.			x	1.0
9.3	Zweryfikuj, czy wszystkie dane wrażliwe są przesyłane do serwera wewnątrz wiadomości HTTP lub jej nagłówka (np. parametry URL nie powinny być używane do przesyłania danych wrażliwych).	x	x	x	1.0
9.4	Zweryfikuj, czy aplikacja posiada ustawienia w nagłówku, uniemożliwiające cachowanie danych odpowiednie do założonego ryzyka aplikacji, np.:	x	x	x	1.0

	Expires: Tue, 03 Jul 2001 06:00:00 GMT Last-Modified: {now} GMT Cache-Control: no-store, no-cache, must-revalidate, max-age=0 Cache-Control: post-check=0, pre-check=0 Pragma: no-cache				
9.5	Zweryfikuj, czy na serwerze wszystkie wrażliwe dane znajdujące się w cache lub kopiach tymczasowych, są chronione przed nieautoryzowanym dostępem lub czyszczone/unieważniane po uzyskaniu dostępu do danych wrażliwych przez upoważnionego użytkownika.		x	x	1.0
9.6	Zweryfikuj, czy istnieje metoda usunięcia z aplikacji każdego typu danych wrażliwych, na koniec ich wymaganego okresu retencji.			x	1.0
9.7	Zweryfikuj, czy aplikacja minimalizuje ilość parametrów wysyłanych w zapytaniach takimi kanałami jak: ukryte pola, zmienne systemu AJAX, ciasteczka i nagłówki wiadomości.		x	x	2.0
9.8	Zweryfikuj, czy aplikacja potrafi wykrywać i alarmować o nienormalnej liczbie zapytań o informacje mogących świadczyć o próbie gromadzenia danych, na przykład poprzez „screen scraping”.			x	2.0
9.9	Zweryfikuj, czy dane przechowywane po stronie klienta (takie jak lokalnie przechowywane informacje o sesjach HTML5, przechowywane sesje, IndexedDB, ciasteczka: zwykłe i Flash), nie zawierają danych podlegających ochronie oraz danych osobowych (PII).	x	x	x	3.0.1
9.10	Zweryfikuj, czy dostęp do danych wrażliwych jest logowany, w przypadku gdy dane są gromadzone zgodnie z odpowiednimi przepisami w zakresie ochrony danych lub gdy logowanie dostępów jest wymagane.		x	x	3.0
9.11	Zweryfikuj, czy dane wymagające ochrony znajdujące się w pamięci, są nadpisywane zerami gdy tylko przestaną być niezbędne, aby zabezpieczyć się przed atakami wykorzystującymi zrzuty pamięci (memory dumping).		x	x	3.0.1

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- User Privacy Protection Cheat Sheet: https://www.owasp.org/index.php/User_Privacy_Protection_Cheat_Sheet

V10. Wymagania weryfikacji dla zabezpieczania komunikacji

Cele kontrolne

Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- TLS jest zawsze wykorzystywany gdy dane wymagające ochrony są transmitowane;
- Silne algorytmy i szyfry są zawsze wykorzystywane.

Wymagania

#	Opis	1	2	3	Od
10.1	Zweryfikuj, czy możliwe jest zbudowanie ścieżki od zaufanego CA do każdego serwerowego certyfikatu Transport Layer Security (TLS) i czy każdy certyfikat serwera jest ważny.	x	x	x	1.0
10.3	Zweryfikuj, czy dla wszystkich połączeń (zewnętrznych i wewnętrznych), które są uwierzytelniane lub związane z wrażliwymi danymi lub funkcjami, jest wykorzystywany TLS a także nie jest możliwe pogorszenie parametrów połączenia do połączenia niezabezpieczonego. Upewnij się, że preferowany jest najsilniejszy dostępny algorytm szyfrowania.	x	x	x	3.0
10.4	Zweryfikuj, czy błędy związane z wewnętrznymi połączeniami TLS pomiędzy komponentami są logowane.			x	1.0
10.5	Zweryfikuj, czy ścieżki certyfikatów są budowane i weryfikowane dla wszystkich certyfikatów klienckich przy użyciu skonfigurowanych powiązań zaufania i informacji dotyczących unieważnionych certyfikatów.			x	1.0
10.6	Zweryfikuj, czy wszystkie połączenia do zewnętrznych systemów, które dotyczą wrażliwych informacji lub funkcji są uwierzytelniane.		x	x	1.0
10.8	Zweryfikuj, czy aplikacja używa implementacji TLS opartej na pojedynczym standardzie, skonfigurowanej do działania w zatwierdzonym trybie pracy.			x	1.0
10.10	Zweryfikuj, czy mechanizm przypinania certyfikatów publicznych TLS (HPKP) jest używany dla kluczy produkcyjnych i zapasowych. Więcej informacji możesz			x	3.0.1

	znaleźć w odnośnikach w niniejszym standardzie.				
10.11	Zweryfikuj, czy nagłówki HTTP Strict Transport Security są włączone do wszystkich zapytań i dla wszystkich poddomen np. w postaci: Strict-Transport-Security: max-age=15724800; includeSubdomains.	x	x	x	3.0
10.12	Zweryfikuj, czy produkcyjny URL został zgłoszony do listy predefiniowanych domen używających mechanizmu Strict Transport Security, utrzymywanych przez producentów przeglądarek. Więcej informacji znajdziesz w odnośnikach poniżej.			x	3.0
10.13	Upewnij się, że w celu zmniejszenia prawdopodobieństwa pasywnych ataków polegających na zapisywaniu ruchu, używane są szyfry wspierające doskonałe utajnienie przekazywania (forward secrecy).	x	x	x	3.0
10.14	Zweryfikuj, czy są włączone i właściwie skonfigurowane mechanizmy unieważniania certyfikatów, takie jak „zszywanie" odpowiedzi Online Certificate Status Protocol (OSCP).	x	x	x	3.0
10.15	Zweryfikuj, czy używane są tylko silne algorytmy, szyfry i protokoły w ramach całej hierarchii certyfikatów, włącznie z certyfikatem root i certyfikatami pośrednimi w ramach wybranego urzędu certyfikacji.	x	x	x	3.0
10.16	Zweryfikuj, czy konfiguracja TLS jest zgodna z bieżącymi najlepszymi praktykami szczególnie dlatego, że popularne ustawienia, szyfry i algorytmy z czasem mogą okazać się niebezpieczne.	x	x	x	3.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP – TLS Cheat Sheet. https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- Uwaga dotycząca zalecanych trybów TLS. W przeszłości ASVS donosił się jedynie do amerykańskiego standardu FIPS 140-2, jednak z racji swojej międzynarodowości zapewnianie zgodności z amerykańskimi standardami może być trudne, sprzeczne lub mylące. Dlatego lepszym sposobem na zapewniania zgodności z 10.8 jest przestrzeganie zalet opisanych w (https://wiki.mozilla.org/Security/Server_Side_TLS), tworzenie poprawnych konfiguracji (https://mozilla.github.io/server-side-tls/ssl-config-generator/), oraz wykorzystywanie znanych narzędzi pozwalających na ocenę poprawności TLS, takich jak sslyze, różnych skanerów podatności lub zaufanych serwisów online służących do oceny TLS. W ogólności uważamy, że niezgodność z tą

sekcją polega na wykorzystaniu przestarzałych lub niebezpiecznych szyfrów i algorytmów, brak doskonałego utajnienia przekazywania (forward secrecy), a także przestarzałe lub niebezpieczne protokoły SSL czy słabe preferowane szyfry.

- Certificate pinning. Aby zdobyć więcej informacji odwiedź: https://tools.ietf.org/html/rfc7469. Uzasadnieniem dla mechanizmów przypinania certyfikatów do kluczy produkcyjnych i zapasowych jest ciągłość działania - odwiedź: https://noncombatant.org/2015/05/01/about-http-public-key-pinning/
- OWASP Certificate Pinning Cheat Sheet https://www.owasp.org/index.php/Pinning_Cheat_Sheet
- OWASP Certificate and Public Key Pinning https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
- Time of first use (TOFU) Pinning https://developer.mozilla.org/en/docs/Web/Security/Public_Key_Pinning
- Pre-loading HTTP Strict Transport Security https://www.chromium.org/hsts

V11. Wymagania weryfikacji dla konfigurowania bezpieczeństwa HTTP

Cele kontrolne

Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- konfiguracja serwera aplikacyjnego jest odpowiednio utwardzona;
- odpowiedzi http zawierają bezpieczny zestaw znaków w nagłówku "„content type”.

Wymagania

#	Opis	1	2	3	Od
11.1	Zweryfikuj, czy aplikacja akceptuje tylko zdefiniowany zestaw metod zapytań HTTP, takich jak GET i POST oraz czy nieużywane metody (takie jak TRACE, PUT, DELETE) są w sposób wyraźny zablokowane.	x	x	x	1.0
11.2	Zweryfikuj, czy każda odpowiedź HTTP zawiera nagłówek „content type”, określający bezpieczny zestaw znaków (np. UTF-8, ISO 8859-1).	x	x	x	1.0
11.3	Zweryfikuj, czy w nagłówki HTTP dodawane przez zaufane proxy lub urządzenia SSO, takie jak token okaziciela, są uwierzytelniane przez aplikację.		x	x	2.0
11.4	Zweryfikuj, czy właściwy nagłówek X-FRAME-OPTIONS jest używany dla witryn, których zawartość nie powinna być wyświetlana w ramkach innych serwisów.		x	x	3.0.1
11.5	Zweryfikuj, czy nagłówki HTTP lub jakakolwiek część odpowiedzi HTTP nie ujawniają szczegółowej informacji o wersjach komponentów systemu.	x	x	x	2.0
11.6	Zweryfikuj, czy wszystkie odpowiedzi z API zawierają nagłówki X-Content-Type-Options: nosniff i Content-Disposition: attachment; filename="api.json" (lub inne nazwy plików odpowiednie dla jego typu).	x	x	x	3.0
11.7	Zweryfikuj, czy mechanizm Content Security Policy V2 (CSP) jest używany aby zapobiegać podatnościom wstrzyknięcia DOM, XSS, JSON i JavaScript	x	x	x	3.0.1
11.8	Zweryfikuj, czy nagłówek X-XSS-Protection: 1; mode=block jest użyty by uruchomić w przeglądarce	x	x	x	3.0

filtry chroniące przed reflected XSS.

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Testing for HTTP Verb Tampering https://www.owasp.org/index.php/Testing_for_HTTP_Verb_Tampering_%28OTGINPVAL-003%29
- Dodawanie do odpowiedzi API nagłówków Content-Disposition pomaga zapobiegać wielu atakom bazującym na niewłaściwym zrozumieniu rodzaju MIME pomiędzy klientem a serwerem, a opcja "filename" pomaga zapobiegać atakom Reflected File Download https://www.blackhat.com/docs/eu-14/materials/eu-14-Hafif-Reflected-FileDownload-A-New-Web-Attack-Vector.pdf
- https://www.owasp.org/index.php?title=Content_Security_Policy_Cheat_Sheet&setlang=en

V12: Wymagania weryfikacji bezpieczeństwa konfiguracji

Ta sekcja została włączona w skład V11. Wymagania weryfikacji dla konfigurowania bezpieczeństwa HTTP w wersji 2.0 Application Security Verification Standard.

V13. Wymagania weryfikacji zabezpieczeń przed złośliwym kodem

Cele kontrolne

Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- złośliwe czynności są obsługiwane w sposób bezpieczny i poprawny, tak aby nie wywrzeć wpływu na resztę aplikacji;
- aplikacja nie ma wbudowanych "bomb czasowych" ani innych ataków bazujących na czasie;
- aplikacja nie próbuje się kontaktować ze złośliwymi lub nieautoryzowanymi lokalizacjami ;
- aplikacja nie ma tylnych furtek, jajek z niespodzianką (easter egg), ataków typu salami czy błędów logicznych, które mogłyby być kontrolowane przez atakującego.

Złośliwy kod jest niezmiernie rzadki i trudno do wykryć. Ręczna analiza kodu linia po linii może wspomóc wykrycie bomb logicznych, ale nawet najbardziej doświadczony analityk z trudem go wykryje - nawet gdy będzie wiedział o jego istnieniu. Nie uda się spełnić wymagań tej sekcji bez dostępu do kodu źródłowego, włączając w to tak dużo zewnętrznych bibliotek jak tylko możliwe.

Wymagania

#	Opis	1	2	3	Od
13.1	Zweryfikuj, czy wszystkie złośliwe działania są odpowiednio sandboxowane, konteneryzowane lub izolowane aby opóźnić i powstrzymać atakujących przed atakami na inne aplikacje.			x	2.0
13.2	Zweryfikuj kod źródłowy aplikacji oraz tyle bibliotek dostarczonych przez strony trzecie, ile to tylko możliwe i potwierdź, że nie zawierają złośliwego kodu, tylnych furtek, ukrytych niespodzianek (Easter eggs) oraz błędów logicznych w ramach uwierzytelniania, kontroli dostępu, walidacji danych wejściowych i logice biznesowej dla transakcji wysokiej wartości.			x	3.0.1

Odnośniki

Aby zdobyć więcej informacji odwiedź:
- http://www.dwheeler.com/essays/apple-goto-fail.html

V14: Wymagania dotyczące weryfikacji bezpieczeństwa wewnętrznego

Ta sekcja została włączona w skład V13. Wymagania weryfikacji zabezpieczeń przed złośliwym kodem w wersji 2.0 Application Security Verification Standard.

V15. Wymagania weryfikacji logiki biznesowej

Cele kontrolne

Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- przepływ logiki biznesowej jest sekwencyjny i uporządkowany;
- logika biznesowa zawiera ograniczania pozwalające na wykrywanie i zapobieganie zautomatyzowanym atakom, takich jak np. nieustanne przekazywanie drobnych sum pieniędzy czy dodawanie na raz miliona przyjaciół;
- przepływy logiki biznesowej o wysokiej wartości biorą pod uwagę nadużycia czy nieuczciwe osoby i tym samym mają wbudowane zabezpieczenia przed podszywaniem się, modyfikowaniem danych, zaprzeczaniem, wyciekiem informacji i eskalacją uprawnień.

Wymagania

#	Opis	1	2	3	Od
15.1	Zweryfikuj, czy aplikacja zezwala na przetwarzanie procesów logiki biznesowej jedynie krok po kroku, wszystkie kroki w procesie przetwarzane są w czasie jaki zajmuje to człowiekowi, kroki ustawione z złej kolejności nie są przetwarzane także nie pozwala się na pomijanie kroków, użycie kroków z procesów innych użytkowników i zbyt szybkie wysłanie operacji.		x	x	2.0
15.2	Zweryfikuj, czy aplikacja posiada limity biznesowe i prawidłowo wymusza je dla każdego użytkownika, a także czy można zdefiniować powiadomienia oraz automatyczną odpowiedź na ataki automatyczne bądź nietypowe operacje.		x	x	2.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Business Logic Testing https://www.owasp.org/index.php/Testing_for_business_logic
- OWASP Cheat Sheet: https://www.owasp.org/index.php/Business_Logic_Security_Cheat_Sheet

V16. Wymagania weryfikacji dla plików i innych zasobów

Cele kontrolne

Upewnij się, że weryfikowana aplikacja w odpowiedni sposób spełnia poniższe ogólne wymagania:

- niezaufane dane z plików powinny być obsługiwane w sposób poprawny i bezpieczny;
- pliki źródłowe otrzymane z niezaufanych źródeł są przechowywane poza katalogiem główny aplikacji (webroot) i z ograniczonymi uprawnieniami.

Wymagania

#	Opis	1	2	3	Od
16.1	Zweryfikuj, czy przeadresowania i przekierowania URL są dozwolone jedynie dla predefiniowanych (white label) stron internetowych i pokazują użytkownikowi ostrzeżenie w sytuacji przekierowania do potencjalnie niezaufanych treści.	x	x	x	2.0
16.2	Zweryfikuj, czy niezaufane dane z plików wysłanych do aplikacji nie są bezpośrednio przekazywane do komend wykonujących operacje na drzewie plików, w szczególności by zapewnić ochronę przed podatnościami typu „Path Traversal”, „Local File Inclusion” i „OS command injection”.	x	x	x	2.0
16.3	Zweryfikuj, czy pliki pozyskane z niezaufanych źródeł są walidowane pod kątem oczekiwanego typu pliku i czy są skanowane programem antywirusowym w celu ochrony przed szkodliwą zawartością.	x	x	x	2.0
16.4	Zweryfikuj, czy niezaufane dane nie są użyte bezpośrednio w mechanizmach ładowania, ładowania klas i zwracania, by zapewnić ochronę przed atakami typu „Remote/Local File Inclusion”.	x	x	x	2.0
16.5	Zweryfikuj, czy niezaufane dane nie są wykorzystywane w ramach mechanizmu „Cross-Origin Resource Sharing” (CORS), w celu zapewnienia ochrony przed potencjalnie złośliwymi zewnętrznymi treściami.	x	x	x	2.0
16.6	Zweryfikuj, czy pliki otrzymane z nieznanych źródeł są umieszczone poza katalogiem głównym aplikacji (webroot), z minimalnymi uprawnieniami i z zalecaną silną walidacją.		x	x	3.0

16.7	Zweryfikuj, czy serwer webowy lub aplikacyjny są domyślnie skonfigurowane tak, aby odrzucać połączenia do zewnętrznych zasobów lub systemów poza serwerem www lub aplikacyjnym.		x	x	2.0
16.8	Zweryfikuj, czy kod aplikacji nie wykonuje danych otrzymanych z niezaufanych źródeł.	x	x	x	3.0
16.9	Nie używaj Flasha, Active-X, Silverlighta, NACL, appletów Java ani innej technologii klienckiej, która nie jest natywnie wspierana przez standardy W3C w przeglądarkach.	x	x	x	2.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- File Extension Handling for Sensitive Information: https://www.owasp.org/index.php/Unrestricted_File_Upload

V17. Wymagania weryfikacji dla aplikacji mobilnych

Cele kontrolne

Ta sekcja zawiera zabezpieczenia specyficzne dla aplikacji mobilnych. Te zabezpieczenia są skopiowane z wersji 2.0 i powinny być rozpatrywane w połączeniu ze wszystkimi innymi sekcjami dla właściwych poziomów weryfikacji ASVS. Aplikacje mobilne powinny:

- mieć taki sam poziom zabezpieczeń w mobilnym kliencie jak na serwerze, poprzez wymuszanie mechanizmów kontrolnych w bezpiecznym środowisku;
- przechowywanie informacji wymagających ochrony na urządzeniach powinno być realizowane w sposób bezpieczny;
- transmitowanie wszystkich danych wymagających ochrony z urządzeń powinno być realizowane z myślą o bezpieczeństwie warstwy transmisji.

Wymagania

#	Opis	1	2	3	Od
17.1	Zweryfikuj, czy wartości identyfikatorów, takie jak UDID czy IMEI, przechowywanych na urządzeniu i odzyskiwanych przez inne aplikacje, nie są wykorzystywane jako tokeny uwierzytelnienia.	x	x	x	2.0
17.2	Zweryfikuj, czy aplikacja mobilna nie przechowuje wrażliwych danych na potencjalnie niezaszyfrowanych zasobach współdzielonych na urządzeniu (karta SD, współdzielone foldery).	x	x	x	2.0
17.3	Zweryfikuj, czy wrażliwe dane nie są przechowywane na urządzeniu w sposób niezabezpieczony, nawet w obszarach chronionych systemowo, takich jak pęki kluczy (key chains).	x	x	x	2.0
17.4	Zweryfikuj, czy klucze, tokeny API lub hasła są generowane dynamicznie przez aplikację.		x	x	2.0
17.5	Zweryfikuj, czy aplikacja zapobiega wyciekom poufnych informacji (np. zrzuty ekranów zapisują się w aktualnej aplikacji podczas gdy aplikacja działa w tle lub zapisuje wrażliwe informacje w konsoli).		x	x	2.0
17.6	Zweryfikuj, czy aplikacja wymaga minimalnych uprawnień dla wymaganych funkcjonalności i zasobów.		x	x	2.0

17.7	Zweryfikuj, czy wrażliwy kod aplikacji umieszczany jest w pamięci w sposób nieprzewidywalny (np. wykorzystując ASLR).	x	x	x	2.0
17.8	Zweryfikuj, czy obecne techniki zapobiegające debugowaniu są wystarczające do zatrzymania lub opóźnienia potencjalnych atakujących przed wstrzyknięciem debuggerów do aplikacji (np. GDB).			x	2.0
17.9	Zweryfikuj, czy aplikacja nie eksportuje wrażliwych danych w obrębie androidowych mechanizmów „activities”, „intents”, „content providers” itp. innym aplikacjom na tym samym urządzeniu.	x	x	x	2.0
17.10	Zweryfikuj, czy dane wymagające ochrony znajdujące się w pamięci, są nadpisywane zerami gdy tylko przestaną być niezbędne, aby zabezpieczyć się przed atakami zrzucania pamięci (memory dumping).			x	3.0.1
17.11	Zweryfikuj, czy aplikacja waliduje poprawność danych wejściowych dla eksportowanych czynności, intencji i dostarczycieli treści.	x	x	x	3.0.1

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Mobile Security Project: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- iOS Developer Cheat Sheet: https://www.owasp.org/index.php/IOS_Developer_Cheat_Sheet

V18. Wymagania weryfikacyjne dla web serwisów

Cele kontrolne

Upewnij się, że weryfikowana aplikacja, która wykorzystuje usługi sieciowe RESTful lub SOAP ma:

- adekwatne uwierzytelnianie, zarządzanie sesją i autoryzację wszystkich serwisów sieciowych;
- walidację wszystkich parametrów wejściowych, które są transmitowane z mniej do bardziej zaufanych warstw;
- podstawową interoperacyjność usług sieciowych SOAP, w celu promowania wykorzystywania API.

Wymagania

#	Opis	1	2	3	Od
18.1	Zweryfikuj, czy ten sam styl kodowania znaków jest wykorzystywany przez klienta i serwer.	x	x	x	3.0
18.2	Zweryfikuj, czy dostęp do funkcji administracyjnych i zarządczych w ramach aplikacji jest ograniczony do wąskiej grupy administratorów.	x	x	x	3.0
18.3	Zweryfikuj, czy schemat XML lub JSON jest stosowany i czy jest weryfikowany przed zaakceptowaniem danych wejściowych.	x	x	x	3.0
18.4	Zweryfikuj, czy wszystkie dane wejściowe mają odpowiednie ograniczenia długości.	x	x	x	3.0
18.5	Zweryfikuj, czy usługi sieciowe oparte o SOAP są zgodne co najmniej z profilem podstawowym Web Services-Interoperability (WS-I Basic Profile). W szczególności dotyczy to szyfrowania TLS.	x	x	x	3.0.1
18.6	Zweryfikuj, czy uwierzytelnienie i autoryzacja dotycząca sesji są wykorzystywane. Proszę skorzystać z sekcji 2, 3 i 4 w celu uzyskania dalszych instrukcji. Należy unikać stosowania statycznych kluczy API i podobnych rozwiązań.	x	x	x	3.0
.18.7	Zweryfikuj, czy usługi REST są chronione przed atakami typu Cross-Site Request Forgery wykorzystując co najmniej jeden mechanizm spośród weryfikacji ORIGIN, podwójnego	x	x	x	3.0.1

	wprowadzania szablonów ciasteczek (cookie pattern), CSRF nonces oraz sprawdzeń punktu odniesienia (referrer checks).				
18.8	Zweryfikuj, czy usługi REST wyraźnie sprawdzają czy przychodzące wartości Content-Type są spodziewanymi, np. czy jest to application/xml lub application/json.		x	x	3.0
18.9	Zweryfikuj, czy treść wiadomości jest podpisana, aby zapewnić bezpieczną transmisję pomiędzy klientem a usługą, wykorzystując JSON Web Signing lub WS-Security dla żądań SOAP.		x	x	3.0.1
18.10	Zweryfikuj, czy nie istnieją alternatywne i mniej bezpieczne ścieżki dostępu.		x	x	3.0

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing https://www.owasp.org/index.php/Testing_for_configuration_management
- OWASP Cross-Site Request Forgery cheat sheet https://www.owasp.org/index.php/CrossSite_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
- JSON Web Tokens (and Signing) https://jwt.io/

V19. Wymagania weryfikacyjne dla procesu konfiguracji

Cele kontrolne

Upewnij się, że weryfikowana aplikacja:

- wykorzystuje aktualne biblioteki i platformy;
- wykorzystuje bezpieczną konfigurację bazową;
- jest wystarczająco utwardzona, tak aby zmiany zainicjowane przez użytkowników w konfiguracji bazowej niekoniecznie eksponowały lub tworzyły słabości bezpieczeństwa lub uszkadzały zasadnicze systemy.

Wymagania

#	Opis	1	2	3	Od
19.1	Wszystkie komponenty powinny być aktualne, z właściwymi ustawieniami dotyczącymi bezpieczeństwa i wersjami. To powinno uwzględniać: usunięcie niepotrzebnych wpisów konfiguracyjnych i folderów takich jak przykładowe aplikacje, dokumentację platformy oraz domyślnych bądź przykładowych użytkowników.	x	x	x	3.0
19.2	Komunikacja pomiędzy komponentami, na przykład pomiędzy serwerem aplikacyjnym a serwerem bazodanowym, powinna być szyfrowana, szczególnie w przypadku gdy komponenty znajdują się w różnych kontenerach lub na różnych systemach.		x	x	3.0
19.3	Komunikacja pomiędzy komponentami, na przykład pomiędzy serwerem aplikacyjnym a serwerem bazodanowym, powinna być uwierzytelniona z użyciem konta z najmniejszymi koniecznymi przywilejami.		x	x	3.0
19.4	Zweryfikuj, czy wdrożone aplikacja są należycie zabezpieczone poprzez mechanizmy typu sandbox, umieszczenie w kontenerach bądź izolowane w celu opóźnienia lub powstrzymania atakujących przed uzyskaniem dostępu do innych aplikacji.		x	x	3.0
19.5	Zweryfikuj, czy proces budowania oraz wdrażania oprogramowania jest wykonany w sposób bezpieczny.		x	x	3.0
19.6	Zweryfikuj, czy uprawnieni administratorzy mają możliwość sprawdzenia spójności wszystkich ustawień istotnych z			x	3.0

	punktu widzenia bezpieczeństwa, aby zapewnić, że nie były one modyfikowane w sposób nieupoważniony.		
19.7	Zweryfikuj, czy wszystkie komponenty aplikacji są podpisane.	x	3.0
19.8	Zweryfikuj, czy zewnętrze biblioteki pochodzą z zaufanych repozytoriów.	x	3.0
19.9	Upewnij się, że w procesie budowania (built process) oprogramowania, włączone są wszystkie mechanizmy takie jak ASLR, DEP i inne związane z bezpieczeństwem.	x	3.0
19.10	Zweryfikuj czy wszystkie zasoby aplikacyjne, takie jak biblioteki JavaScript, arkusze stylów CSS i fonty webowe są hostowane przez aplikację, a nie przez CDN (Content Delivery Network) lub zewnętrznych dostawców.	x	3.0.1

Odnośniki

Aby zdobyć więcej informacji odwiedź:

- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing https://www.owasp.org/index.php/Testing_for_configuration_management

Załącznik A: Co się stało z...

(załącznik został celowo pominięty w tłumaczeniu)

Załącznik B: Słownik

- **Kontrola dostępu (Access Control)** – środki ograniczania dostępu do plików, odwołań do funkcji, adresów URL i danych realizowany w oparciu tożsamość użytkowników i/lub grup do których użytkownicy należą.
- **Randomizacja warstwy przestrzeni adresowej (Address Space Layout Randomization - ASLR)** – technika pomagająca chronić przed atakami przepełnienia bufora,
- **Bezpieczeństwo aplikacyjne (Application Security)** – bezpieczeństwo na poziomie aplikacji realizowane w taki sposób by koncentrować się na analizie elementów składających się na warstwę aplikacji w modelu „Open Systems Interconnection Reference Model" (Model OSI), zamiast skupiać się na przykład na systemie operacyjnym lub sieciach komputerowych.
- **Weryfikacja zabezpieczeń aplikacji (Application Security Verification**) - ocena techniczna zgodności aplikacji z ASVS OWASP.
- **Raport weryfikacji zabezpieczeń aplikacji (Application Security Verification Report)** - raport, który dokumentuje ogólne wyniki i analizę pomocniczą, opracowaną przez testera dla określonej aplikacji.
- **Uwierzytelnianie (Authentication) -** weryfikacja tożsamości użytkownika aplikacji.
- **Automatyczna Weryfikacja (Automated Verification)** - wykorzystanie narzędzi automatycznych (dla analizy dynamicznej, statycznej lub obu jednocześnie), które bazują na sygnaturach podatności bezpieczeństwa w celu ich identyfikacji.
- **Tylne furtki (Back Doors)** - rodzaj złośliwego oprogramowania, który umożliwia nieautoryzowany dostęp do aplikacji.
- **Czarna lista (Blacklist)** - lista danych lub operacji, które nie są dozwolone, na przykład lista znaków, które nie są dozwolone jako dane wejściowe.
- **Kaskadowe arkusze stylów (Cascading Style Sheets - CSS)** - Język arkusza stylów używany do opisu semantyki prezentacji dokumentu napisanego językiem znaczników, np. HTML.
- **Urząd certyfikacji (Certificate Authority - CA)** – Jednostka, która wydaje certyfikaty cyfrowe.
- **Bezpieczeństwo komunikacji** - ochrona danych aplikacji podczas przesyłania między składnikami aplikacji, między klientami a serwerami oraz między systemami zewnętrznymi a aplikacją.
- **Komponent (Component)** - samodzielna jednostka kodu, z powiązanymi interfejsami dyskowymi i sieciowymi, która komunikuje się z innymi elementami kodu.
- **Cross-Site Scripting (XSS)** – podatność bezpieczeństwa zwykle występująca w aplikacjach internetowych, pozwalająca na wstrzyknięcie skryptów do treści aplikacji po stronie klienta.
- **Moduł kryptograficzny** - sprzęt, oprogramowanie i/lub oprogramowanie wbudowane (firmware), które implementuje algorytmy kryptograficzne i/lub generuje klucze kryptograficzne.
- **Ataki odmowy usługi (Denial of Service Attacks - DoS Attacks)** – zalewanie aplikacji większą liczbą żądań niż jest w stanie obsłużyć.
- **Weryfikacja projektu (Design Verification)** - techniczna ocena architektury bezpieczeństwa aplikacji.

- **Dynamiczna weryfikacja (Dynamic Verification)** – użycie zautomatyzowanych narzędzi korzystających z sygnatur w celu wykrycia podatności w trakcie działania aplikacji.
- **jajko z niespodzianką (Easter Eggs)** – rodzaj złośliwego oprogramowania, które nie uruchamia się do czasu wystąpienia konkretnego zdarzenia ze strony użytkownika
- **Systemy zewnętrzne** - aplikacje lub usługi po stronie serwera, które nie są częścią składową aplikacji.
- **FIPS 140-2** – standard, który może służyć jako podstawa do weryfikacji zaprojektowania i wdrażania modułów kryptograficznych.
- **Globally Unique Identifier (GUID)** – unikalny numer referencyjny używany jako identyfikator w oprogramowaniu.
- **Hyper Text Markup Language (HTML)** – główny język znaczników do tworzenia stron internetowych i innych informacji wyświetlanych w przeglądarce internetowej.
- **Hyper Text Transfer Protocol (HTTP)** – protokół aplikacji dla rozproszonych, współpracujących, hipermedialnych systemów informacyjnych. Jest podstawą komunikacji danych dla World Wide Web.
- **Weryfikacja wejściowa (Input Validation)** – konwertowanie kanoniczne i walidacja niezaufanych danych użytkownika.
- **Lightweight Directory Access Protocol (LDAP)** – protokół aplikacyjne wykorzystywany do uzyskiwania dostępu do i rozpowszechniania informacji katalogowych rozproszonych w sieci
- **Złośliwy kod (Malicious Code)** – kod wprowadzony do aplikacji w trakcie jej budowania, nieznany właścicielowi aplikacji, który narusza zamierzoną politykę bezpieczeństwa. Nie jest to tożsame ze złośliwym oprogramowaniem, takim jak wirus lub robak!
- **Złośliwe oprogramowanie (Malware)** - kod wykonywalny wprowadzony do aplikacji podczas jej pracy bez wiedzy użytkownika oraz administratora aplikacji.
- **Open Web Application Security Project (OWASP)** – OWASP jest ogólnoświatową wolną i otwartą społecznością skupiającą się na poprawie bezpieczeństwa aplikacji. Naszym celem jest sprawienie, aby bezpieczeństwo aplikacji stało się „widoczne", dzięki czemu ludzie i organizacje mogą podejmować świadome decyzje dotyczące zagrożeń dla bezpieczeństwa aplikacji. Patrz: http://www.owasp.org/
- **Kodowanie danych wyjściowych (Output encoding)** – konwertowanie kanoniczne i walidacja danych wyjściowych aplikacji w przeglądarkach internetowych i systemach zewnętrznych.
- **Dane osobowe** – to informacje, które można wykorzystać samodzielnie lub z innymi informacjami, aby zidentyfikować, skontaktować się lub znaleźć pojedynczą osobę lub zidentyfikować osobę w kontekście.
- **Pozytywna walidacja** – Zobacz: biała lista (whitelist).
- **Architektura bezpieczeństwa (Security Architecture)** – abstrakcja projektu aplikacji, która identyfikuje i opisuje, gdzie i w jaki sposób są stosowane poszczególne zabezpieczenia, a także identyfikuje i opisuje lokalizację i wrażliwość zarówno danych użytkowników, jak i samej aplikacji.
- **Konfiguracja zabezpieczeń (Security Configuration)** – konfiguracja środowiska wykonawczego aplikacji definiująca sposób użycia zabezpieczeń

- **Mechanizm bezpieczeństwa (Security Control)** – funkcja lub składnik, który określa sprawdzenie stanu zabezpieczenia (na przykład kontroli dostępu) lub elementu wywoływanego w efekcie zdarzenia bezpieczeństwa (np. generowanie wpisu do logów).
- **SQL Injection (SQLi)** – technika wstrzykiwania kodu używana do atakowania aplikacji sterowanych danymi, w których złośliwe instrukcje SQL wprowadzane są do punktów wejścia danych.
- **Weryfikacja statyczna** - użycie zautomatyzowanych narzędzi wykorzystujących sygnatury podatności, w celu znalezienia ich w kodzie źródłowym aplikacji.
- **Cel weryfikacji (Target of Verification - TOV)** – w przypadku gdy wykonujesz weryfikację zabezpieczeń konkretnej aplikacji zgodnie z wymaganiami OWASP ASVS, to ta weryfikacja będzie dotyczyć konkretnej aplikacji. Określa się ją jako "cel weryfikacji" lub po prostu TOV.
- **Modelowanie zagrożeń (Threat Modeling)** – technika polegająca na rozwijaniu coraz bardziej wyrafinowanych architektur bezpieczeństwa w celu identyfikacji agentów zagrożenia, stref bezpieczeństwa, kontroli bezpieczeństwa i ważnych zasobów technicznych i biznesowych.
- **Transport Layer Security (TLS)** – protokoły kryptograficzne zapewniające bezpieczeństwo komunikacji przez Internet,
- **URI/URL/częściowy URL** – „Uniform Resource Identifier” to ciąg znaków używanych do identyfikacji nazwy lub zasobu sieciowego. „Uniform Resource Locator” jest często używany jako odniesienie do zasobu.
- **Środowisko testów akceptacyjnych użytkownika (User acceptance testing - UAT)** – środowisko testowe, które zachowuje się jak środowisko produkcyjne, w którym wszystkie testy oprogramowania przeprowadzane są przed rozpoczęciem wdrożenia na środowisku produkcyjnym.
- **Weryfikator (Verifier)** – osoba lub zespół, który przeprowadza przegląd zgodności aplikacji zgodnie z wytycznymi standardu OWASP ASVS.
- **Biała Lista (Whitelist)** – lista dozwolonych danych lub operacji, na przykład lista znaków, które są dozwolone podczas walidacji danych wejściowych.
- **XML** – język znaczników definiujący zestaw reguł do kodowania dokumentów.

Załącznik C: Odnośniki

Następujące projekty OWASP najprawdopodobniej okażą się najbardziej użyteczne podczas korzystania z niniejszego standardu:

- OWASP Testing Guide
https://www.owasp.org/index.php/OWASP_Testing_Project
- OWASP Code Review Guide
http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project
- OWASP Cheat Sheets
https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series
- OWASP Proactive Controls
https://www.owasp.org/index.php/OWASP_Proactive_Controls
- OWASP Top 10
https://www.owasp.org/index.php/Top_10_2013-Top_10
- OWASP Mobile Top 10
https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks

Podobnie, poniższe strony www najprawdopodobniej okażą się najbardziej użyteczne podczas korzystania z niniejszego standardu:

- MITRE Common Weakness Enumeration - http://cwe.mitre.org/
- PCI Security Standards Council - https://www.pcisecuritystandards.org
- PCI Data Security Standard (DSS) v3.0 Requirements and Security Assessment Procedures
https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf

Załącznik D: Mapowanie standardu

punkt 6.5 standardu PCI DSS pochodzi od listy OWASP Top 10 w wersji 2004/2007, dodając do niego pewne rozszerzenia. ASVS jest znaczącym rozszerzeniem listy OWASP Top 10 2013 (154 elementów w miejsce 10), więc wszystkie kwestie pokryte przez OWASP Top 10 oraz PCI DSS 6.5.x są w ASVS umieszczone w znacznie bardziej granularny sposób. Na przykład "6.5.10 Przerwane uwierzytelnianie oraz zarządzanie sesją." mapuje się na sekcje V2 Uwierzytelnianie oraz V3 Zarządzanie sesją.
Pełne mapowanie jest osiągane przez poziom 3, chociaż już poziom 2 zaadresuje większość z wymagań PCI DSS 6., poza 6.5.3 oraz 6.5.4. Kwestie dotyczące procesów, takie jak wymaganie 6,5,6 PCI DSS, nie są pokryte przez ASVS.

PCI DSS 3.0	ASVS 3.0	Opis
6.5.1 Błędy wstrzykiwania, a szczególnie SQL injection. Należy również rozważyć wstrzykiwanie komend OS, LDAP oraz XPath jak i inne tego typu błędy	5.11, 5.12, 5.13, 8.14, 16.2	Dokładne mapowanie
6.5.2 Przepełnienie bufora	5.1	Dokładne mapowanie
6.5.3 Niebezpieczne przechowywanie materiału kryptograficznego	V7 - cały	Kompleksowe mapowanie od poziomu 1 w górę
6.5.4 Niebezpieczna komunikacja	V10 - cały	Kompleksowe mapowanie od poziomu 1 w górę
6.5.5 Niepoprawna obsługa błędów	3.6, 7.2, 8.1, 8.2	Dokładne mapowanie
6.5.7 Cross-site scripting (XSS)	5.16, 5.20, 5.21, 5.24, 5.25, 5.26, 5.27, 11.4,11.15	ASVS rozkłada XSS na szereg wymagań podkreślając potrzebę kompleksowej ochrony przez XSS, szczególnie dla starszych aplikacji
6.5.8 Niepoprawna kontrola dostępu (np. niebezpieczne bezpośrednie odwołania do obiektów, brak zastosowania restrykcji dla URL, przeglądanie katalogu plików (directory traversal) oraz niepoprawne ograniczanie dostępu użytkowników do funkcji).	V4 - cały	Kompleksowe mapowania od poziomu 1 w górę

PCI DSS 3.0	ASVS 3.0	Opis
6.5.9 Cross-site request forgery (CSRF).	4.13	Dokładne mapowanie. ASVS traktuje obronę przez CSRF jako element kontroli dostępu
6.5.10 Przerwane uwierzytelnianie oraz zarządzanie sesją.	V2 i v3 - cały	Kompleksowe mapowania od poziomu 1 w górę