**OWASP APPLICATION SECURITY VERIFICATION STANDARD 4.0**

**Używanie ASVS**

ASVS ma dwa główne cele

• pomóc firmom tworzyć i konserwować bezpieczne aplikacje.

• pozwolić dostawcom usług bezpieczeństwa, narzędzi bezpieczeństwa i konsumentom na dostosowanie ich ofert i wymagań.

**Poziomy Weryfikacji Bezpieczeństwa Aplikacji**

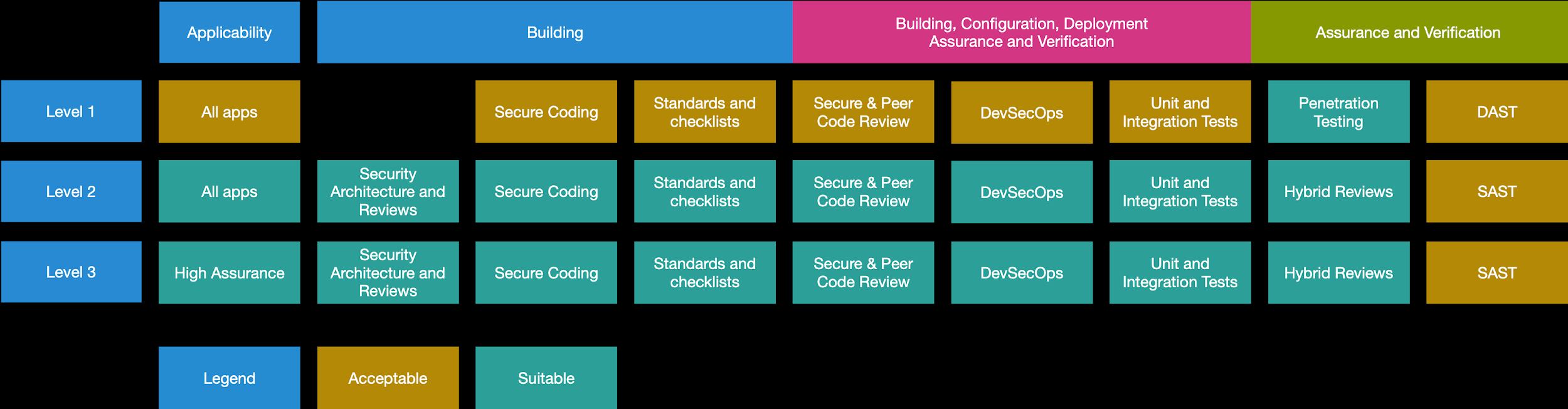
Standard Weryfikacji Bezpieczeństwa Aplikacji definiuje trzy poziomy weryfikacji bezpieczeństwa przy czym każdy poziom jest coraz ważniejszy

• ASVS Poziom 1 zapewnia niski poziom pewności i jest w pełni testowalny pod względem penetracji

• ASVS Poziom 2 jest przeznaczony dla aplikacji zawierających poufne dane, które wymagają ochrony i jest zalecanym poziomem dla większości aplikacji

• ASVS Poziom 3 jest przeznaczony dla najbardziej krytycznych aplikacji - tych, które wykonują transakcje o wysokiej wartości, zawierają wrażliwe dane medyczne lub dla dowolnej aplikacji wymagającej najwyższego poziomu zaufania.

Każdy poziom ASVS zawiera listę wymagań bezpieczeństwa. Każde z tych wymagań może być również mapowane do funkcji i możliwości związanych z bezpieczeństwem, które muszą być wbudowane w oprogramowanie przez programistów.

****

Poziomy OWASP ASVS 4.0

Poziom 1 to jedyny poziom, który jest całkowicie możliwy do przetestowania testami penetracyjnymi przez ludzi. Wszystkie inne wymagają dostępu do dokumentacji, kodu źródłowego, konfiguracji i osób zaangażowanych w proces rozwoju. Jednak nawet jeśli L1 zezwala na testowanie „black box” (bez dokumentacji i źródła), nie jest to skuteczne zabezpieczenie i musimy je zakończyć. Złośliwi atakujący mają dużo czasu, większość testów penetracyjnych kończy się w ciągu kilku tygodni.

Osoby odpowiedzialne za ochronę muszą tworzyć kontrole bezpieczeństwa, chronić, znajdować i rozwiązywać wszystkie słabości oraz wykrywać i reagować na hakerów w rozsądnym czasie. Hakerzy mają zasadniczo nieskończony czas i wymagają tylko jednej dziurawej obrony, pojedynczej słabości lub brakującego wykrywania, aby odnieść sukces. Testowanie “black box”, często wykonywane pod koniec rozwoju, na szybko, lub nie jest wcale wykonywane, nie jest w stanie poradzić sobie z tą asymetrią.

W ciągu ponad 30 ostatnich lat testy “black box” wielokrotnie pokazały, że pomijają problemy związane z bezpieczeństwem, które doprowadziły bezpośrednio do coraz większych naruszeń. Zdecydowanie zachęcamy do korzystania z szerokiej gamy zabezpieczeń i weryfikacji bezpieczeństwa, w tym do zastąpienia testów penetracyjnych testami penetracyjnymi prowadzonymi przez kod źródłowy (hybrydowowe) na poziomie 1, z pełnym dostępem do programistów i dokumentacji przez cały proces rozwoju. Organy nadzoru finansowego nie tolerują zewnętrznych audytów finansowych bez dostępu do ksiąg rachunkowych, przykładowych transakcji ani osób przeprowadzających kontrolę. Przemysł i rząd muszą żądać tego samego standardu przejrzystości w dziedzinie inżynierii oprogramowania.

Zdecydowanie zachęcamy do korzystania z narzędzi bezpieczeństwa, ale w samym procesie rozwoju, tzn używanie narzędzi DAST i SAST w kolejce kompilacji, aby łatwo znaleźć problemy z bezpieczeństwem, które nigdy nie powinny być obecne.

Zautomatyzowane narzędzia i skanowanie online nie są w stanie zrealizować ponad połowy ASVS bez pomocy człowieka. Jeśli wymagana jest wszechstronna automatyzacja testów dla każdej kompilacji, wówczas używana jest kombinacja testów jednostkowych i testów integracyjnych oraz skanów inicjowanych online. Wady logiki biznesowej i testy kontroli dostępu są możliwe tylko przy pomocy człowieka. Powinny one zostać przekształcone w testy jednostkowe i integracyjne.

**Jak używać tego standardu**

Jednym z najlepszych sposobów korzystania ze standardu weryfikacji bezpieczeństwa aplikacji jest wykorzystanie go jako wzoru do utworzenia Listy Kontrolnej Bezpiecznego Kodowania specyficznej dla Twojej aplikacji, platformy lub organizacji. Dostosowanie ASVS do Twoich przypadków użycia (use cases) zwiększy nacisk na wymagania bezpieczeństwa, które są najważniejsze dla twoich projektów i środowisk.

**Level 1 - Pierwsze kroki, zautomatyzowane, lub widok całego portfolio**

Aplikacja osiąga ASVS Level 1, jeśli odpowiednio chroni przed lukami w zabezpieczeniach aplikacji, które są łatwe do wykrycia i zawarte w OWASP Top 10 i innych podobnych listach kontrolnych.

Poziom 1 to absolutne minimum, do którego powinny dążyć wszystkie aplikacje. Jest również przydatny jako pierwszy krok w wielofazowym wysiłku lub gdy aplikacje nie przechowują lub nie obsługują poufnych danych, a zatem nie potrzebują bardziej rygorystycznych kontroli poziomu 2 lub 3. Kontrole poziomu 1 można sprawdzać automatycznie za pomocą narzędzi lub po prostu ręcznie bez dostępu do kodu źródłowego. Uważamy poziom 1 za minimalny wymagany dla wszystkich aplikacji.

Zagrożenia dla aplikacji będą najprawdopodobniej powodowane przez osoby atakujące, które wykorzystują proste i mało wymagające techniki do identyfikacji łatwych do znalezienia i łatwych do wykorzystania luk w zabezpieczeniach. Jest to przeciwieństwo zdeterminowanego hakera, który włoży wysiłek w zaatakowanie aplikacji. Jeśli dane przetwarzane przez twoją aplikację mają wysoką wartość, rzadko powinieneś zatrzymywać się na Level 1.

**Level 2 - Większość aplikacji**

Aplikacja osiąga poziom 2 ASVS (lub Standard), jeśli odpowiednio chroni przed większością zagrożeń związanych z dzisiejszym oprogramowaniem.

Poziom 2 zapewnia, że kontrole bezpieczeństwa są stosowane, skuteczne i używane w aplikacji. Poziom 2 jest zazwyczaj odpowiedni dla aplikacji, które obsługują istotne transakcje między przedsiębiorstwami, w tym transakcje przetwarzające informacje o opiece zdrowotnej, wdrażają krytyczne lub wrażliwe funkcje biznesowe, lub przetwarzają inne wrażliwe aktywa lub branże, w których integralność jest krytycznym aspektem ochrony ich działalności, np. w przemyśle gier, w celu pokrzyżowania planów oszustom i hakerom.

Zagrożeniem dla aplikacji poziomu 2 będą zazwyczaj wykwalifikowanymi i zmotywowanymi atakujący koncentrującymi się na konkretnych celach i wykorzystujący narzędzia i techniki, które są wysoce praktykowane i skuteczne w odkrywaniu i wykorzystywaniu słabych punktów aplikacji.

**Level 3 - Wysoka wartość, wysoka pewność, lub wysokie bezpieczeństwo**

Level 3 to najwyższy poziom weryfikacji w ramach ASVS. Ten poziom jest zazwyczaj zarezerwowany dla aplikacji, które wymagają znacznego poziomu weryfikacji bezpieczeństwa, takich jak te, które można znaleźć w obszarach wojskowych, zdrowia i bezpieczeństwa, infrastruktury krytycznej itp.

Organizacje mogą wymagać ASVS Level 3 dla aplikacji, które wykonują funkcje krytyczne, w których awaria mogłaby znacząco wpłynąć na działania organizacji, a nawet na jej przetrwanie. Przykładowe wskazówki dotyczące stosowania poziomu 3 ASVS przedstawiono poniżej. Aplikacja osiąga ASVS Level 3 (lub Advanced), jeśli odpowiednio chroni przed zaawansowanymi lukami w zabezpieczeniach aplikacji, a także demonstruje zasady dobrego projektu bezpieczeństwa.

Aplikacja na ASVS Level 3 wymaga głębszej analizy architektury, kodowania i testowania niż wszystkie inne poziomy. Bezpieczna aplikacja jest zmodularyzowana w znaczący sposób (w celu ułatwienia elastyczności, skalowalności, a przede wszystkim warstw bezpieczeństwa), a każdy moduł (oddzielony przez połączenie sieciowe i / lub instancję fizyczną) dba o własne obowiązki w zakresie bezpieczeństwa (dogłębna ochrona), które muszą być odpowiednio udokumentowane. Obowiązki obejmują kontrole mające na celu zapewnienie poufności (np. szyfrowanie), integralność (np. transakcje, walidację danych wejściowych), dostępność (np. obsługa ładowania w odpowiedni sposób), uwierzytelnianie (w tym między systemami), niezaprzeczalność, autoryzację i audyt (logowanie).

**Stosowanie ASVS w praktyce**

Różne zagrożenia mają różne motywy. Niektóre branże posiadają unikalne zasoby informacji i technologii oraz wymagania zgodne z przepisami danej dziedziny.

Zaleca się organizacjom, aby dokładnie przyjrzały się swoim unikalnym cechom ryzyka w oparciu o charakter swojej działalności i na podstawie tego ryzyka i wymagań biznesowych określiły odpowiedni poziom ASVS.

**V1. Architektura, projektowanie i modelowanie zagrożeń**

Cel:

Architektura bezpieczeństwa stała się niemal zagubioną sztuką w wielu organizacjach. Dni architekta korporacyjnego minęły za czasów DevSecOps. Pole bezpieczeństwa aplikacji musi nadrobić zaległości i przyjąć zwinne zasady bezpieczeństwa, ponownie wprowadzając wiodące zasady architektury bezpieczeństwa dla praktyków oprogramowania. Architektura nie jest implementacją, ale sposobem myślenia o problemie, który ma potencjalnie wiele różnych odpowiedzi i nie ma jednej „poprawnej” odpowiedzi. Zbyt często bezpieczeństwo jest postrzegane jako nieelastyczne i wymagające, ponieważ programiści muszą naprawiać kod w określony sposób, podczas gdy mogą znać znacznie lepszy sposób rozwiązania problemu. Nie ma jednego, prostego rozwiązania dla architektury, a udawanie, że jest inaczej, jest niekorzystne dla inżynierii oprogramowania.

Specyficzna implementacja aplikacji internetowej prawdopodobnie będzie stale korygowana przez cały okres jej istnienia, ale ogólna architektura prawdopodobnie będzie rzadko się zmieniać i ewoluować powoli. Architektura zabezpieczeń jest identyczna - potrzebujemy dziś uwierzytelnienia, jutro też będziemy tego wymagać, tak jak również za pięć lat. Jeśli dzisiaj podejmiemy rozsądne decyzje, możemy zaoszczędzić dużo wysiłku, czasu a także pieniędzy, jeśli wybierzemy i ponownie wykorzystamy rozwiązania zgodne z architekturą. Na przykład dekadę temu uwierzytelnianie wieloczynnikowe rzadko było wdrażane.

Gdyby programiści zainwestowali w pojedynczy, bezpieczny model dostawcy tożsamości, taki jak tożsamość federacyjna SAML, dostawca tożsamości mógłby zostać zaktualizowany w celu uwzględnienia nowych wymagań, takich jak zgodność z NIST 800-63, bez zmiany interfejsów oryginalnej aplikacji. Gdyby wiele aplikacji korzystało z tej samej architektury zabezpieczeń, a tym samym z tego samego komponentu, wszystkie zyskałyby na tej aktualizacji od razu. Jednak SAML nie zawsze pozostanie najlepszym lub najbardziej odpowiednim rozwiązaniem uwierzytelniającym - może zaistnieć potrzeba wymiany na inne rozwiązania, gdy zmienią się wymagania. Takie zmiany są albo skomplikowane, albo tak kosztowne, że wymagają pełnego przepisania, albo wręcz niemożliwe bez architektury bezpieczeństwa.

W tym rozdziale ASVS omawia podstawowe aspekty solidnej architektury bezpieczeństwa: dostępność, poufność, integralność przetwarzania, niezaprzeczalność i prywatność. Każda z tych zasad bezpieczeństwa musi być wbudowana od początku dla wszystkich aplikacji. Kluczowe jest „przesunięcie w lewo”, począwszy od włączenia programisty do list kontrolnych bezpiecznego kodowania, mentoringu i szkolenia, kodowania i testowania, budowania, wdrażania, konfiguracji i operacji, a kończąc na niezależnych testach, aby upewnić się, że wszystkie mechanizmy bezpieczeństwa są obecne i działające. Ostatnim krokiem było wszystko, co robiliśmy jako przemysł, ale to już nie wystarcza, ponieważ programiści przesyłają kod dziesiątki czy setek razy dziennie. Specjaliści ds. bezpieczeństwa aplikacji muszą nadążać za zwinnymi technikami, co oznacza wdrażanie narzędzi programistycznych, uczenie się kodu i pracę z programistami, zamiast krytykowania projektu przez kilka miesięcy po tym, jak wszyscy inni przeszli dalej.

**V1.1: Weryfikacja wymagań bezpiecznego Cyklu Życia Oprogramowania**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.1.1 | Zweryfikuj użycie bezpiecznego cyklu rozwoju oprogramowania, który dotyczy bezpieczeństwa na wszystkich etapach rozwoju. |  | ✓ | ✓ |  |
| 1.1.2 | Zweryfikuj wykorzystanie modelowania zagrożeń dla każdej zmiany projektu lub szybkie planowanie, aby zidentyfikować zagrożenia, zaplanować środki zaradcze, ułatwić odpowiednie reakcje na ryzyko i przeprowadzić testy bezpieczeństwa. |  | ✓ | ✓ | 1053 |
| 1.1.3 | Sprawdź, czy wszystkie “user stories” i funkcje zawierają funkcjonalne ograniczenia bezpieczeństwa, takie jak „Jako użytkownik powinienem móc wyświetlać i edytować mój profil. Nie powinienem mieć możliwości przeglądania ani edytowania profilu innych osób” |  | ✓ | ✓ | 1110 |
| 1.1.4 | Sprawdź dokumentację i uzasadnienie wszystkich granic zaufania, komponentów i ważnych przepływów danych aplikacji. |  | ✓ | ✓ | 1059 |
| 1.1.5 | Sprawdź definicję i analizę bezpieczeństwa architektury aplikacji i wszystkich połączonych usług zdalnych. |  | ✓ | ✓ | 1059 |
| 1.1.6 | Zweryfikuj implementację scentralizowanej, prostej (ekonomiczna konstrukcja), sprawdzonej, bezpiecznej i wielokrotnego użytku kontroli bezpieczeństwa, aby uniknąć podwójnych, brakujących, nieskutecznych lub niepewnych kontroli. |  | ✓ | ✓ | 637 |
| 1.1.7 | Sprawdź dostępność listy kontrolnej bezpiecznego kodowania, wymagań bezpieczeństwa, wytycznych lub zasad dla wszystkich programistów i testerów. |  | ✓ | ✓ | 637 |

**V1.2: Weryfikacja wymagań architektonicznych dotyczących Autoryzacji**

Cel:

Podczas projektowania autoryzacji nie ma znaczenia, czy masz włączone uwierzytelnianie wieloczynnikowe, jeśli atakujący może zresetować konto, dzwoniąc do centrum obsługi i odpowiadając na powszechnie znane pytania. Podczas sprawdzania tożsamości wszystkie ścieżki autoryzacji muszą mieć tę samą siłę.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.2.1 | Zweryfikuj użycie unikalnych lub specjalnych kont systemu operacyjnego o niskich uprawnieniach dla wszystkich  komponentów aplikacji, usług i serwerów. |  | ✓ | ✓ | 250 |
| 1.2.2 | Sprawdź, czy komunikacja między komponentami aplikacji, w tym API, oprogramowaniem pośrednim i warstwami danych, jest autoryzowana. Komponenty powinny mieć najniższe niezbędne uprawnienia. |  | ✓ | ✓ | 306 |
| 1.2.3 | Sprawdź, czy aplikacja korzysta z pojedynczego sprawdzonego mechanizmu autoryzacji, o którym wiadomo, że jest bezpieczny, że można go rozszerzyć o silne uwierzytelnianie i czy ma wystarczające logi i monitorowanie, aby wykryć nadużycia lub naruszenia konta. |  | ✓ | ✓ | 306 |
| 1.2.4 | Zweryfikuj czy pola wejściowe haseł pozwalają lub zachęcają do używania tekstu szyfrującego i nie zapobiegają wprowadzania haseł, które są długimi lub bardzo złożonymi tekstami szyfrującymi. |  | ✓ | ✓ | 306 |

**V1.3: Wymagania architektoniczne dotyczące Zarządzania Sesją**

**Miejsce na przyszłe wymagania architektoniczne**

**V1.4: Wymagania architektoniczne dotyczące Kontroli dostępu**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.4.1 | Sprawdź, czy zaufane punkty kontaktowe, takie jak bramy kontroli dostępu, serwery i funkcje bez serwera, wymuszają kontrolę dostępu. Nigdy nie wymuszaj kontroli dostępu na kliencie. |  | ✓ | ✓ | 602 |
| 1.4.2 | Sprawdź, czy wybrane rozwiązanie kontroli dostępu jest wystarczająco elastyczne, aby spełnić potrzeby aplikacji. |  | ✓ | ✓ | 284 |
| 1.4.3 | Zweryfikuj egzekwowanie zasady najniższych uprawnień w funkcjach, plikach danych, adresach URL, kontrolerach, usługach i innych zasobach. Oznacza to ochronę przed spoofingiem i zwiększenie uprawnień. |  | ✓ | ✓ | 272 |
| 1.4.4 | Sprawdź, czy aplikacja korzysta z pojedynczego i sprawdzonego mechanizmu kontroli dostępu do chronionych danych i zasobów. Wszystkie żądania muszą przejść przez ten pojedynczy mechanizm, aby uniknąć kopiowania i wklejania lub niezabezpieczonych alternatywnych ścieżek. |  | ✓ | ✓ | 284 |
| 1.4.5 | Sprawdź, czy używana jest kontrola dostępu na bazie funkcji lub atrybutów, dzięki której kod sprawdza autoryzację użytkownika dla funkcji/elementu danych, a nie tylko jego rolę. Uprawnienia należy nadal przydzielać za pomocą ról. |  | ✓ | ✓ | 275 |

**V1.5: Wymagania architektoniczne dotyczące danych wejściowych i wyjściowych**

W wersji 4.0 odeszliśmy od terminu „po stronie serwera” jako terminu nacechowanego granicznym zaufaniem. Granica zaufania nadal obowiązuje - podejmowanie decyzji dotyczących niezaufanych przeglądarek lub urządzeń klienta jest pomijane. Jednak w dzisiejszych głównych instalacjach architektonicznych punkt egzekwowania zaufania znacznie się zmienił. Dlatego też, gdy termin „zaufana warstwa usług” jest używany w ASVS, mamy na myśli każdy zaufany punkt egzekwowania, niezależnie od lokalizacji, taki jak mikroserwis, API bez serwera, po stronie serwera, zaufany interfejs API na urządzeniu klienckim, które ma bezpieczne uruchamianie, partnerskie lub zewnętrzne API, itd.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.5.1 | Sprawdź, czy wymagania dotyczące danych wejściowych i wyjściowych jasno określają, w jaki sposób obsługiwać i przetwarzać dane na podstawie typu, treści i obowiązujących przepisów, regulacji i innych zasad zgodności z polityką. |  | ✓ | ✓ | 1029 |
| 1.5.2 | Sprawdź, czy serializacja nie jest używana podczas komunikacji z niezaufanymi klientami. Jeśli jest to niemożliwe, upewnij się, że odpowiednie kontrole integralności (i ewentualnie szyfrowanie, jeśli wysyłane są dane wrażliwe) są wymuszane, aby zapobiec atakom deserializacji, w tym “object injection”. |  | ✓ | ✓ | 502 |
| 1.5.3 | Sprawdź, czy sprawdzanie poprawności danych wejściowych jest wymuszane na zaufanej warstwie usług |  | ✓ | ✓ | 602 |
| 1.5.4 | Sprawdź, czy kodowanie wyjściowe występuje blisko lub przez interpretera, dla którego jest przeznaczone. |  | ✓ | ✓ | 116 |

**V1.6. Kryptograficzne wymagania architektoniczne**

Aplikacje muszą być zaprojektowane z silną architekturą kryptograficzną, aby chronić zasoby danych zgodnie z ich klasyfikacją. Szyfrowanie wszystkiego jest marnotrawstwem, nie szyfrowanie niczego jest prawnym zaniedbaniem. Należy zachować równowagę, zwykle podczas projektowania architektonicznego lub projektowania wysokiego poziomu, procesu Design Sprint, lub Architectural Spikes. Zaprojektowanie kryptografii w biegu lub jej modernizacja nieuchronnie będzie kosztować znacznie więcej, niż bezpieczne jej wdrożenie od samego początku.

Wymagania architektoniczne są nierozerwalnie związane z całą bazą kodu, a tym samym trudne jeśli chodzi o testy jednostkowe lub integracyjne.

Wymagania architektoniczne wymagają rozpatrzenia w standardach kodowania podczas całej fazy kodowania i powinny być weryfikowane podczas przeglądów architektury bezpieczeństwa, przeglądów naukowych, przeglądów kodu lub retrospektyw.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.6.1 | Sprawdź, czy istnieje wyraźna zasada zarządzania kluczami kryptograficznymi i czy cykl życia klucza kryptograficznego jest zgodny ze standardem zarządzania kluczami, takim jak NIST SP 800-57. |  | ✓ | ✓ | 320 |
| 1.6.2 | Sprawdź, czy konsumenci usług kryptograficznych chronią kluczowe materiały i inne tajemnice za pomocą Key Vaults lub alternatyw opartych na API. |  | ✓ | ✓ | 320 |
| 1.6.3 | Sprawdź, czy wszystkie klucze i hasła są wymienne i są częścią dobrze zdefiniowanego procesu ponownego szyfrowania poufnych danych. |  | ✓ | ✓ | 320 |
| 1.6.4 | Sprawdź, czy klucze symetryczne, hasła lub tajemnice API generowane przez klientów lub udostępniane klientom są używane tylko do ochrony tajemnic niskiego ryzyka, takich jak szyfrowanie pamięci lokalnej lub tymczasowe efemeryczne zastosowania, takie jak zaciemnianie parametrów. Dzielenie się tajemnicami z klientami jest równoznaczne z użyciem zwykłego tekstu architektonicznie powinno być traktowane jako takie. |  | ✓ | ✓ | 320 |

**V1.7: Wymagania architektoniczne dotyczące Obsługi i Rejestrowania Błędów**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.7.1 | Sprawdź, czy w systemie używany jest wspólny format rejestrowania błędów i to samo podejście. |  | ✓ | ✓ | 1009 |
| 1.7.2 | Sprawdź, czy logi są bezpiecznie przesyłane do zdalnego systemu w celu analizy, wykrywania, alarmowania i eskalacji. |  | ✓ | ✓ |  |

**V1.8: Wymagania architektoniczne dotyczące Ochrony Danych i Prywatności**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.8.1 | Sprawdź, czy wszystkie wrażliwe dane zostały zidentyfikowane i sklasyfikowane do poziomów ochrony. |  | ✓ | ✓ |  |
| 1.8.2 | Sprawdź, czy wszystkie poziomy ochrony mają powiązany zestaw wymagań dotyczących bezpieczeństwa, takich jak wymagania dotyczące szyfrowania, integralności, retencji, prywatności i inne wymagania dotyczące poufności, i że są one stosowane w architekturze. |  | ✓ | ✓ |  |

**V1.9: Wymagania architektoniczne dotyczące Komunikacji**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 1.9.1 | Sprawdź, czy aplikacja szyfruje komunikację między komponentami, szczególnie gdy te komponenty znajdują się w różnych kontenerach(?), systemach, witrynach lub u dostawców chmury. |  | ✓ | ✓ | 319 |
| 1.9.2 | Sprawdź, czy komponenty aplikacji weryfikują autentyczność każdej strony w łączu komunikacyjnym, aby zapobiec atakom typu „person-in-the-middle”. Na przykład komponenty aplikacji powinny sprawdzać poprawność certyfikatów TLS i łańcuchów. |  | ✓ | ✓ | 295 |

**V1.10 Wymagania architektoniczne dotyczące złośliwego oprogramowania**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **1** | **2** | **3** | **CWE** |
| 1.10.1 | Sprawdź, czy używany jest system kontroli kodu źródłowego, z procedurami zapewniającymi, że check-inom towarzyszą issues lub tickety zmian. System kontroli kodu źródłowego powinien mieć kontrolę dostępu i identyfikowalnych użytkowników, aby umożliwić śledzenie wszelkich zmian. |  | ✓ | ✓ | 284 |

**V1.11. Wymagania architektoniczne dotyczące Logiki Biznesowej**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **1** | **2** | **3** | **CWE** |
| 1.11.1 | Sprawdź definicję i dokumentację wszystkich komponentów aplikacji pod kątem funkcji biznesowych lub funkcji bezpieczeństwa, które zapewniają. |  | ✓ | ✓ | 1059 |
| 1.11.2 | Sprawdź, czy wszystkie przepływy logiki biznesowej o wysokiej wartości, w tym uwierzytelnianie, zarządzanie sesjami i kontrola dostępu, nie współdzielą stanu niezsynchronizowanego. |  | ✓ | ✓ | 362 |
| 1.11.3 | Sprawdź, czy wszystkie przepływy logiki biznesowej o wysokiej wartości, w tym uwierzytelnianie, zarządzanie sesjami i kontrola dostępu, są bezpieczne dla wątków i odporne na warunki time-of-check i time-of-use. |  |  | ✓ | 367 |

**V1.12: Wymagania architektoniczne dotyczące bezpiecznego przesyłania plików**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **1** | **2** | **3** | **CWE** |
| 1.12.1 | Sprawdź, czy pliki przesłane przez użytkownika są przechowywane poza głównym katalogiem WWW. |  | ✓ | ✓ | 552 |
| 1.12.2 | Sprawdź, czy pliki przesłane przez użytkownika - jeśli są wymagane do wyświetlenia lub pobrania z aplikacji - są obsługiwane przez pobieranie octet stream lub z niepowiązanej domeny, takiej jak wiadro (? bucket) do przechowywania plików w chmurze. Zaimplementuj odpowiednią politykę bezpieczeństwa treści, aby zmniejszyć ryzyko związane z wektorami XSS lub innymi atakami z przesłanego pliku. |  | ✓ | ✓ | 646 |

**V1.13. Wymagania architektoniczne dotyczące API**

**Miejsce na przyszłe wymagania architektoniczne**

**V1.14. Wymagania architektoniczne konfiguracji**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **1** | **2** | **3** | **CWE** |
| 1.14.1 | Sprawdź segregację komponentów o różnych poziomach zaufania za pomocą dobrze zdefiniowanych mechanizmów bezpieczeństwa, reguł zapory, API gateways, odwrotnych proxy, grup zabezpieczeń opartych na chmurze, lub podobnych mechanizmów. |  | ✓ | ✓ | 923 |
| 1.14.2 | Sprawdź, czy wdrażanie plików binarnych na niezaufanych urządzeniach używa podpisów binarnych, zaufanych połączeń i zweryfikowanych punktów końcowych. |  | ✓ | ✓ | 494 |
| 1.14.3 | Sprawdź, czy pipeline kompilacji ostrzega o nieaktualnych lub niezabezpieczonych komponentach i podejmuje odpowiednie działania. |  | ✓ | ✓ | 1104 |
| 1.14.4 | Sprawdź, czy pipeline kompilacji zawiera build step, aby automatycznie budować i weryfikować bezpieczne wdrożenie aplikacji, szczególnie jeśli infrastruktura aplikacji jest zdefiniowana przez oprogramowanie, na przykład skrypty kompilacji środowiska w chmurze. |  | ✓ | ✓ |  |
| 1.14.5 | Sprawdź, czy wdrożenia aplikacji odpowiednio sandboxują, kontenerują i / lub izolują na poziomie sieci, aby opóźnić i powstrzymać atakujących od atakowania innych aplikacji, zwłaszcza gdy wykonują wrażliwe lub niebezpieczne działania, takie jak deserializacja. |  | ✓ | ✓ | 265 |
| 1.14.6 | Upewnij się, czy aplikacja nie korzysta z nieobsługiwanych, niezabezpieczonych lub przestarzałych technologii klienta, takich jak wtyczki NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL lub aplety Java po stronie klienta. |  | ✓ | ✓ | 477 |

Referencje:

• OWASP Threat Modeling Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Threat_Modeling_Cheat_Sheet.md>

• OWASP Attack Surface Analysis Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Attack_Surface_Analysis_Cheat_Sheet.md>

• OWASP Threat modeling <https://www.owasp.org/index.php/Application_Threat_Modeling>

• OWASP Secure SDLC Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets_excluded/Secure_SDLC_Cheat_Sheet.md>

• Microsoft SDL <https://www.microsoft.com/en-us/securityengineering/sdl/>

• NIST SP 800-57 <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf>

**V2: Wymagania dotyczące weryfikacji uwierzytelnienia**

Cel:

Uwierzytelnianie to czynność polegająca na ustanowieniu lub potwierdzeniu osoby (lub rzeczy) jako autentycznej oraz tego, że roszczenia złożone przez osobę lub urządzenie są poprawne, odporne na podszywanie się i uniemożliwiają odzyskanie lub przechwycenie haseł.

Po pierwszej wersji ASVS nazwa użytkownika i hasło były najczęstszą formą uwierzytelniania poza systemami wysokiego bezpieczeństwa. Uwierzytelnianie wieloczynnikowe (MFA) było powszechnie akceptowane w kręgach bezpieczeństwa, ale rzadko wymagane gdzie indziej. Wraz ze wzrostem liczby naruszeń haseł pomysł, że nazwy użytkowników są w jakiś sposób poufne, a hasła nieznane, sprawił, że wiele zabezpieczeń było bezcelowe. Na przykład NIST 800-63 uznaje nazwy użytkowników i uwierzytelnianie oparte na wiedzy (KBA) za informacje publiczne, powiadomienia SMS i e-mail jako „poufne” typy uwierzytelniające oraz hasła jako wstępnie naruszone. Ta rzeczywistość sprawia, że uwierzytelnianie oparte na wiedzy, odzyskiwanie wiadomości SMS i e-mail, historia haseł, złożoność i kontrola rotacji są bezużyteczne. Te regulacje zawsze były mało pomocne, często zmuszając użytkowników do wymyślania słabych haseł co kilka miesięcy, ale po opublikowaniu ponad 5 miliardów naruszeń nazwy użytkownika i hasła nadszedł czas, aby przejść dalej.

Spośród wszystkich sekcji ASVS rozdziały o uwierzytelnianiu i zarządzaniu sesjami najbardziej się zmieniły. Przyjęcie skutecznej, opartej na dowodach praktyki będzie dla wielu wyzwaniem, co jest całkowicie w porządku. Musimy teraz rozpocząć przejście na przyszłość “post-password” - po haśle.

**NIST 800-63 - Nowoczesny, oparty na dowodach standard uwierzytelniania**

NIST 800-63b jest nowoczesnym, opartym na dowodach standardem i stanowi najlepsze dostępne zalecenie. Standard jest pomocny dla wszystkich organizacji na całym świecie, ale jest szczególnie przydatny w przypadku agencji amerykańskich i tych, które mają styczność z agencjami amerykańskimi.

Terminologia NIST 800-63 na początku może być nieco myląca, zwłaszcza jeśli używasz tylko nazwy użytkownika + hasła. Konieczne są postępy we współczesnym uwierzytelnianiu, dlatego musimy wprowadzić terminologię, która stanie się powszechna w przyszłości, ale rozumiemy trudności w zrozumieniu, dopóki przemysł nie przyzwyczai się do nowości. Dlatego, na końcu tego rozdziału przedstawiliśmy słownik. Sformułowaliśmy na nowo wiele wymagań, aby spełnić wymóg, a nie warunek. Na przykład ASVS używa terminu „hasło”, podczas gdy NIST używa „zapamiętanego sekretu” (ang. memorized secret).

ASVS V2 Uwierzytelnianie, V3 Zarządzanie sesją, a w mniejszym stopniu V4 Kontrole Dostępu zostały dopasowane do zgodności z wybranym podzbiorem wybranych regulacji NIST 800-63b, skupionych wokół wspólnych zagrożeń i powszechnie wykorzystywanych słabości uwierzytelniania. Jeśli wymagana jest pełna zgodność z NIST 800-63, należy skonsultować się z NIST 800-63.

**Wybór odpowiedniego poziomu NIST AAL**

Aplikacja Application Security Verification Standard próbowała odwzorować ASVS L1 względem wymagań NIST AAL1, L2 wzgledem AAL2 i L3 względem AAL3. Jednak podejście ASVS Level 1 jako „najważniejszych” kontroli niekoniecznie musi być na tym samym poziomie w AAL. Na przykład, jeśli aplikacja jest aplikacją Level 3 lub ma wymagania regulacyjne jak AAL3, należy wybrać poziom 3 w sekcjach V2 i V3 Zarządzania Sesją. Wybór zgodnego z NIST poziomu potwierdzania autentyczności (AAL) powinien być przeprowadzony zgodnie z wytycznymi NIST 800-63b, jak podano w “Selecting AAL” w NIST 800-63b Rozdział 6.2.

**Legenda**

Aplikacje mogą zawsze przekraczać wymagania obecnego poziomu, zwłaszcza jeśli nowoczesne uwierzytelnianie znajduje się w roadmapie aplikacji. Wcześniej ASVS wymagało obowiązkowego MFA. NIST nie wymaga obowiązkowego MFA. Dlatego w tym rozdziale użyliśmy opcjonalnego oznaczenia, aby wskazać, gdzie ASVS zachęca, ale nie wymaga kontroli. W tym standardzie używane są następujące klucze:

|  |  |
| --- | --- |
| **Znak** | **Opis** |
|  | Nie wymagane |
| o | Zalecane, ale nie wymagane |
| ✓ | Wymagane |

**V 2.1 Wymagania bezpieczeństwa hasła**

Hasła, nazywane przez NIST 800-63 „Memorized Secrets” (Zapamiętane sekrety), zawierają hasła, PIN-y, wzory odblokowania, wybór właściwego kotka lub innego elementu obrazu i passphrases. Są one na ogół uważane za „coś, co wiesz” i często są używane jako pojedyncze czynniki uwierzytelniające. Ciągłe stosowanie uwierzytelniania single-factor wiąże się z poważnymi wyzwaniami, w tym miliardami ważnych nazw użytkowników i haseł ujawnianych w Internecie, domyślnych lub słabych haseł, tablic tęczowych i uporządkowanych słowników najpopularniejszych haseł.

Aplikacje powinny zdecydowanie zachęcać użytkowników do rejestrowania się w uwierzytelnianiu wieloczynnikowym i powinny umożliwiać użytkownikom ponowne wykorzystanie posiadanych już tokenów, takich jak tokeny FIDO lub U2F, lub linkowanie do dostawcy usług uwierzytelniających, który zapewnia uwierzytelnianie wieloczynnikowe.

Dostawcy usług poświadczeń (CSP) zapewniają użytkownikom tożsamość federacyjną. Użytkownicy często mają więcej niż jedną tożsamość z wieloma CSP, takimi jak tożsamość przedsiębiorstwa przy użyciu Azure AD, Okta, Ping Identity lub Google, lub tożsamość konsumentów za pomocą Facebooka, Twittera, Google lub WeChat, aby wymienić tylko kilka popularnych alternatyw. Ta lista nie ma na celu wyrażenia poparcia dla tych firm lub usług, ale jest po prostu zachętą dla deweloperów do rozważenia rzeczywistości, że wielu użytkowników ma wiele ustalonych tożsamości. Organizacje powinny rozważyć integrację z istniejącymi tożsamościami użytkowników, zgodnie z profilem ryzyka sprawdzania tożsamości przez CSP. Na przykład, jest mało prawdopodobne, aby organizacja rządowa zaakceptowała tożsamość mediów społecznościowych jako login dla wrażliwych systemów, ponieważ łatwo jest stworzyć fałszywa tożsamość lub wyrzucić tożsamości, podczas gdy firma produkująca gry mobilne może potrzebować integracji z głównymi platformami mediów społecznościowych. rozwijać swoją aktywną bazę graczy.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **1** | **2** | **3** | **CWE** | **NIST** |
| 2.1.1 | Sprawdź, czy hasła mają co najmniej 12 znaków długości. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1.2 | Sprawdź, czy hasła o długości 64 lub więcej znaków są dopuszczalne. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1.3 | Sprawdź, czy hasła mogą zawierać spacje, a obcinanie nie jest wykonywane. Wielokrotne następujące po sobie spacje MOGĄ być opcjonalnie połączone. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1.4 | Sprawdź, czy w hasłach dozwolone są znaki Unicode. Pojedynczy punkt kodowy Unicode jest uważany za znak, więc 12 znaków emoji lub 64 znaków kanji powinno być prawidłowe i dozwolone. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1.5 | Upewnij się że użytkownicy mogą zmieniać swoje hasła | ✓ | ✓ | ✓ | 620 | 5.1.1.2 |
| 2.1.6 | Sprawdź, czy funkcja zmiany hasła wymaga obecnego i nowego hasła użytkownika.. | ✓ | ✓ | ✓ | 620 | 5.1.1.2 |
| 2.1.7 | Sprawdź, czy hasła przesłane podczas rejestracji konta, logowania i zmiany hasła są sprawdzane w zestawie naruszonych haseł lokalnie (np. 1000 lub 10 000 najczęściej używanych haseł zgodnych z zasadami haseł systemu) lub za pomocą zewnętrznego interfejsu API. W przypadku korzystania z interfejsu API należy użyć dowodu zerowej wiedzy lub innego mechanizmu, aby upewnić się, że hasło zwykłego tekstu nie zostanie wysłane lub użyte do sprawdzenia stanu naruszenia hasła. Jeśli hasło zostanie naruszone, aplikacja musi wymagać od użytkownika ustawienia nowego nienaruszonego hasła. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1.8 | Sprawdź, czy istnieje miernik siły hasła, aby pomóc użytkownikom ustawić silniejsze hasło. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1..9 | Upewnij się, że nie istnieją reguły tworzenia haseł ograniczające dozwolony typ znaków. Nie powinno być wymogu, aby użyte były wielkie lub małe litery lub cyfry lub znaki specjalne. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1.10 | Upewnij się, że nie ma okresowych wymagań dotyczących rotacji poświadczeń lub historii haseł. | ✓ | ✓ | ✓ | 263 | 5.1.1.2 |
| 2.1.11 | Sprawdź, czy funkcja „wklej”, helper haseł przeglądarki i zewnętrzny manager haseł są dozwolone. | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| 2.1.12 | Sprawdź, czy użytkownik może albo tymczasowo wyświetlić całe zakryte hasło, albo tymczasowo wyświetlić ostatni wpisany znak hasła na platformach, które nie mają tego jako natywnej funkcji | ✓ | ✓ |  | 521 | 5.1.1.2 |

Uwaga: Celem umożliwienia użytkownikowi wyświetlenia hasła lub tymczasowego zobaczenia ostatniego znaku jest zwiększenie użyteczności wprowadzania danych uwierzytelniających, szczególnie w przypadku używania dłuższych haseł, haseł i menedżerów haseł. Innym powodem uwzględnienia tego wymogu jest zniechęcenie lub uniemożliwienie raportom testowym, niepotrzebnie wymagającym od organizacji zastąpienia zachowania natywnego pola hasła platformy, w celu usunięcia tego nowoczesnego przyjaznego dla użytkownika bezpieczeństwa.

**V2.2 Ogólne wymagania uwierzytelnienia**

Sprawność uwierzytelniania jest niezbędna dla aplikacji przyszłościowych. Zrefaktoruj weryfikatory aplikacji aby umożliwić dodatkowe uwierzytelnianie zgodnie z preferencjami użytkownika, a także umożliwiając wycofanie przestarzałego lub niebezpiecznego uwierzytelniania w uporządkowany sposób.

NIST uważa e-maile i SMS-y za „ograniczone” typy uwierzytelniania i prawdopodobnie zostaną usunięte z NIST 800-63, a tym samym w przyszłości z ASVS. Aplikacje powinny zaplanować roadmap, który nie wymaga korzystania z poczty e-mail lub SMS.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | 1 | 2 | 3 | CWE | NIST |
| 2.2.1 | Sprawdź, czy mechanizmy kontroli automatyzacji są skuteczne w łagodzeniu testów naruszonych poświadczeń (credentials), brute force, i ataków blokujących konta. Takie kontrole obejmują blokowanie najczęściej naruszanych haseł, miękkie blokady, ograniczanie częstotliwości, CAPTCHA, coraz większe opóźnienia między próbami, ograniczenia adresów IP, lub ograniczenia zależne od ryzyka, takie jak lokalizacja, pierwsze logowanie na urządzeniu, ostatnie próby odblokowania konta, lub podobne. Sprawdź, czy na jednym koncie nie ma więcej niż 100 nieudanych prób na godzinę. | ✓ | ✓ | ✓ | 307 | 5.2.2 /  5.1.1.2 /  5.1.4.2 /  5.1.5.2 |
| 2.2.2 | Sprawdź, czy użycie słabego uwierzytelniania (takiego jak SMS i e-mail) jest ograniczone do drugorzędnej weryfikacji i zatwierdzenia transakcji, a nie do zastąpienia bardziej bezpiecznych metod uwierzytelniania. Upewnij się, że silniejsze metody są oferowane przed słabymi metodami, użytkownicy są świadomi ryzyka lub że istnieją odpowiednie środki ograniczające ryzyko włamań na konta. | ✓ | ✓ | ✓ | 304 | 5.2.10 |
| 2.2.3 | Sprawdź, czy bezpieczne powiadomienia są wysyłane do użytkowników po aktualizacji danych uwierzytelniających, takich jak resetowanie poświadczeń, zmiany adresu e-mail lub adresu, logowanie z nieznanych lub ryzykownych lokalizacji. Preferowane jest korzystanie z push notifications - zamiast SMS-ów lub e-maili, ale w przypadku braku powiadomień push dopuszczalne są SMS lub e-mail, o ile w powiadomieniu nie zostaną ujawnione żadne poufne informacje. | ✓ | ✓ | ✓ | 620 |  |
| 2.2.4 | Zweryfikuj odporność na podszywanie się pod inne osoby związane z phishingiem, czyli użycie uwierzytelniania wieloczynnikowego, urządzenia kryptograficzne (takie jak połączone klucze z ‘push to authenticate’) lub na wyższych poziomach AAL, certyfikaty po stronie klienta. |  |  | ✓ | 308 | 5.2.5 |
| 2.2.5 | Sprawdź, czy w przypadku, gdy dostawca usług uwierzytelniających (CSP) i aplikacja weryfikująca uwierzytelnianie są rozdzielone, między dwoma punktami końcowymi znajduje się wzajemnie uwierzytelniony TLS. |  |  | ✓ | 319 | 5.2.6 |
| 2.2.6 | Zweryfikuj odporność na ataki ‘replay’ poprzez obowiązkowe użycie urządzeń OTP, uwierzytelniaczy kryptograficznych lub kodów lookup. |  |  | ✓ | 308 | 5.2.8 |
| 2.2.7 | Sprawdź zamiar uwierzytelnienia, wymagając wprowadzenia tokenu OTP lub akcji inicjowanej przez użytkownika, takiej jak naciśnięcie przycisku na kluczu sprzętowym FIDO. |  |  | ✓ | 308 | 5.2.9 |

**V2.3: Wymagania dotyczące cyklu życia metod uwierzytelniania (authenticator lifecycle)**

Metody uwierzytelniania to hasła, miękkie tokeny, tokeny sprzętowe i urządzenia biometryczne. Cykl życia metod uwierzytelniających ma zasadnicze znaczenie dla bezpieczeństwa aplikacji - jeśli ktoś może samodzielnie zarejestrować konto bez przedstawiania dowodów tożsamości, może to prowadzić do braku zaufania dotyczącego potwierdzenia tożsamości. W przypadku serwisów społecznościowych, takich jak Reddit, jest to w porządku. W przypadku systemów bankowych większy nacisk na rejestrację, wydawanie danych uwierzytelniających i wykorzystywanie odpowiednich urządzeń ma kluczowe znaczenie dla bezpieczeństwa aplikacji.

Uwaga: Hasła nie powinny mieć maksymalnej żywotności ani nie powinny podlegać rotacji. Hasła należy sprawdzać pod kątem prawdopodobieństwa ich naruszenia, a nie regularnie wymieniać.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.3.1 | Sprawdź, czy początkowe hasła wygenerowane przez system lub kody aktywacyjne POWINNY być bezpiecznie losowo generowane, POWINNY mieć długość co najmniej 6 znaków i MOGĄ zawierać litery i cyfry oraz wygasać po krótkim czasie. Te początkowo wygenerowane hasła nie mogą stać się hasłami długoterminowym. | ✓ | ✓ | ✓ | 330 | 5.1.1.2 / A.3 |
| 2.3.2 | Sprawdź, czy wspierane są rejestracja i korzystanie z urządzeń uwierzytelniających dostarczanych przez subskrybentów, takich jak tokeny U2F lub FIDO. |  | ✓ | ✓ | 308 | 6.1.3 |
| 2.3.3 | Sprawdź, czy instrukcje odnawiania są wysyłane z odpowiednim wyprzedzeniem czasu, wystarczającym do odnowienia uwierzytelnień ustawionych czasowo. |  | ✓ | ✓ | 287 | 6.1.4 |

**V 2.4 Wymagania związane z przechowywaniem poświadczeń (credentials)**

Architekci i programiści powinni stosować się do tej sekcji podczas budowania lub modyfikowania kodu. Ta sekcja może być w pełni zweryfikowana tylko za pomocą przeglądu kodu źródłowego lub za pomocą bezpiecznych testów jednostkowych lub integracyjnych. Testy penetracyjne nie mogą zidentyfikować żadnego z tych problemów.

Lista zatwierdzonych jednokierunkowych funkcji wyprowadzania kluczy jest szczegółowo opisana w NIST 800-63 B sekcja 5.1.1.2 oraz w [BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018)](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=10). Zamiast tych wyborów można wybrać najnowszy krajowy lub regionalny standard dotyczący algorytmu i długości klucza. Ta sekcja nie może być przetestowana używając testów penetracyjnych, więc kontrolki nie są oznaczone jako L1. Jednak ta sekcja ma zasadnicze znaczenie dla bezpieczeństwa poświadczeń, jeśli zostaną skradzione, więc jeśli tworzysz fork ASVS dla wytycznych architektury lub kodowania, lub listy kontrolnej przeglądu kodu źródłowego, umieść te kontrolki z powrotem w L1 w prywatnej wersji.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.4.1 | Sprawdź, czy hasła są przechowywane w formie odpornej na ataki offline. Hasła MUSZĄ być solone (salted) i hashowane za pomocą zatwierdzonej jednokierunkowej pochodnej klucza lub funkcji hashowania hasła. Wyprowadzanie kluczy i funkcje hashowania haseł przyjmują hasło, sól i czynnik kosztu (cost factor) jako dane wejściowe podczas generowania hashu hasła. |  | ✓ | ✓ | 916 | 5.1.1.2 |
| 2.4.2 | Sprawdź, czy sól ma długość co najmniej 32 bitów i zostanie wybrana arbitralnie, aby zminimalizować kolizje wartości soli między przechowywanymi hashami. Dla każdego poświadczenia należy zapisać unikalną wartość soli i wynikowy hash. |  | ✓ | ✓ | 916 | 5.1.1.2 |
| 2.4.3 | Sprawdź, czy w przypadku użycia PBKDF2 liczba iteracji POWINNA być tak duża, na ile pozwala wydajność serwera weryfikacyjnego, zazwyczaj co najmniej 100 000 iteracji |  | ✓ | ✓ | 916 | 5.1.1.2 |
| 2.4.4 | Sprawdź, czy jeśli używany jest bcrypt, współczynnik pracy (work factor) POWINIEN być tak duży, na ile pozwala wydajność serwera weryfikacyjnego, zwykle co najmniej 13. |  | ✓ | ✓ | 916 | 5.1.1.2 |
| 2.4.5 | Sprawdź, czy wykonywana jest dodatkowa iteracja funkcji wyprowadzania klucza, używając wartości soli, która jest tajna i znana tylko weryfikatorowi. Wygeneruj wartość soli za pomocą zatwierdzonego generatora bitów losowych [SP 800-90Ar1] i zapewnij co najmniej minimalny poziom bezpieczeństwa określony w najnowszej wersji SP 800-131A. Tajna wartość soli MUSI być przechowywana oddzielnie od hashowanych haseł (np. w specjalnym urządzeniu, takim jak sprzętowy moduł bezpieczeństwa). |  | ✓ | ✓ | 916 | 5.1.1.2 |

Gdzie wspomniane zostały standardy obowiązujące w USA można użyć regionalnych lub lokalnych standardów oprócz wymaganych standardów amerykańskich.

**V2.5: Wymagania dotyczące odzyskiwania poświadczeń (credentials)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.5.1 | Sprawdź, czy wygenerowane początkowo hasło aktywacyjne lub przypomnienie hasła nie zostały wysłane do użytkownika w postaci zwykłego tekstu. | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| 2.5.2 | Zweryfikuj czy wskazówki dotyczące hasła lub uwierzytelnianie oparte na wiedzy (tak zwane „tajne pytania”) są niedostępne. | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| 2.5.3 | Sprawdź, czy odzyskiwanie hasła w żaden sposób nie ujawnia bieżącego hasła. | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| 2.5.4 | Sprawdź, czy nie są dostępne konta wspólne lub domyślne (np. „Root”, „admin” lub „sa”). | ✓ | ✓ | ✓ | 16 | 5.1.1.2 / A.3 |
| 2.5.5 | Sprawdź, czy użytkownik jest powiadamiany o zmianie lub zastąpieniu czynnika uwierzytelnienia. | ✓ | ✓ | ✓ | 304 | 6.1.2.3 |
| 2.5.6 | Zweryfikuj zapomniane hasło oraz czy inne ścieżki odzyskiwania hasła wykorzystują bezpieczny mechanizm odzyskiwania, taki jak TOTP lub inny token miękki, mobilny push lub inny mechanizm odzyskiwania offline. | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| 2.5.7 | Sprawdź, czy w przypadku utraty OTP lub czynników uwierzytelniania wieloczynnikowego sprawdzanie tożsamości odbywa się na tym samym poziomie, co podczas rejestracji. |  | ✓ | ✓ | 308 | 6.1.2.3 |

**V 2.6: Wymagania dotyczące tajnego weryfikatora wyszukiwania**

Hasła wyszukiwania (Lookup secrets) to wstępnie wygenerowane listy tajnych kodów, podobne do numerów autoryzacji transakcji (TAN), kodów odzyskiwania w mediach społecznościowych lub siatki zawierającej zestaw losowych wartości. Są one bezpiecznie dystrybuowane do użytkowników. Te kody wyszukiwania są używane raz, a po użyciu wszystkich, lista haseł wyszukiwania jest odrzucana. Ten typ uwierzytelnienia jest uważany za „coś, co masz”.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.6.1 | Sprawdź, czy hasła wyszukiwania są używane tylko raz. |  | ✓ | ✓ | 308 | 5.1.2.2 |
| 2.6.2 | Sprawdź, czy hasła wyszukiwania mają wystarczającą losowość (112 bitów entropii), lub jeśli mniej niż 112 bitów entropii, czy są “osolone” (salted) unikalną i losową 32-bitową solą i szyfrowane zatwierdzonym jednokierunkowym hashem. |  | ✓ | ✓ | 330 | 5.1.2.2 |
| 2.6.3 | Sprawdź, czy hasła wyszukiwania są odporne na ataki offline, takie jak przewidywalny przydział wartości. |  | ✓ | ✓ | 310 | 5.1.2.2 |

**V 2.7: Wymagania dotyczące weryfikatora pozapasmowego (out-of-band verificator)**

W przeszłości powszechnym weryfikatorem pozapasowego byłby e-mail lub SMS zawierający link do resetowania hasła. Atakujący używają tego słabego mechanizmu do resetowania kont, których jeszcze nie kontrolują, takich jak przejmowanie konta e-mail osoby i ponowne wykorzystywanie znalezionych linków resetowania. Istnieją lepsze sposoby radzenia sobie z weryfikacją pozapasmową.

Bezpieczne uwierzytelnianie pozapasmowe wykorzystuje urządzenia fizyczne, które mogą komunikować się z weryfikatorem za pośrednictwem bezpiecznego kanału dodatkowego. Przykłady obejmują powiadomienia push na urządzenia mobilne. Ten typ uwierzytelnienia jest uważany za „coś, co masz”. Gdy użytkownik chce się uwierzytelnić, aplikacja weryfikująca wysyła komunikat do uwierzytelnienia pozapasmowego poprzez bezpośrednie połączenie z uwierzytelnieniem, lub pośrednio poprzez usługi strony trzeciej. Wiadomość zawiera kod uwierzytelniający (zazwyczaj losowy sześciocyfrowy numer lub okno dialogowe zatwierdzania modalnego). Aplikacja weryfikująca czeka na otrzymanie kodu uwierzytelniającego przez kanał podstawowy i porównuje wartość hashu odebranej wartości z hashem oryginalnego kodu uwierzytelniającego. Jeśli się zgadzają, weryfikator pozapasmowy może założyć, że użytkownik się uwierzytelnił.

ASVS zakłada, że tylko nieliczni programiści będą opracowywać nowe uwierzytelnianie pozapasmowe, takie jak powiadomienia push, a zatem następujące kontrole ASVS mają zastosowanie do weryfikatorów, takich jak API uwierzytelniania, aplikacje i implementacje pojedynczego logowania. Jeśli tworzysz nowy moduł uwierzytelniania pozapasmowego, zapoznaj się z NIST 800-63B § 5.1.3.1.

Niebezpieczne autoryzacje pozapasmowe, takie jak e-mail i VOIP, są niedozwolone. Uwierzytelnianie PSTN i SMS jest obecnie „ograniczone” przez NIST i powinno być zaniechane na rzecz powiadomień push lub podobnych. Jeśli potrzebujesz użyć uwierzytelniania pozapasmowego w formie telefonu lub SMS-a, zobacz § 5.1.3.3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.7.1 | Upewnij się, że domyślnie nie jest oferowane uwierzytelnianie pozapasmowe (ograniczone przez NIST) wykorzystujące zwykły tekst (clear text), takie jak SMS lub PSTN, a najpierw oferowane są silniejsze alternatywy, takie jak powiadomienia push. | ✓ | ✓ | ✓ | 287 | 5.1.3.2 |
| 2.7.2 | Sprawdź, czy weryfikator pozapasmowy wygasza żądania uwierzytelnienia pozapasmowego, kody lub tokeny po 10 minutach. | ✓ | ✓ | ✓ | 287 | 5.1.2.2 |
| 2.7.3 | Sprawdź, czy żądania uwierzytelnienia pozapasmowego, kody lub tokeny są dostępne tylko raz i tylko dla oryginalnego żądania uwierzytelnienia. | ✓ | ✓ | ✓ | 287 | 5.1.3.2 |
| 2.7.4 | Sprawdź, czy uwierzytelnianie pozapasmowe i weryfikator komunikują się przez bezpieczny niezależny kanał. | ✓ | ✓ | ✓ | 523 | 5.1.3.2 |
| 2.7.5 | Sprawdź, czy weryfikator pozapasmowy zachowuje tylko hashowaną wersję kodu uwierzytelniającego. |  | ✓ | ✓ | 256 | 5.1.3.2 |
| 2.7.6 | Sprawdź, czy początkowy kod uwierzytelniający jest generowany przez bezpieczny generator liczb losowych, zawierający co najmniej 20 bitów entropii (zazwyczaj wystarcza sześciocyfrowy losowy numer). |  | ✓ | ✓ | 310 | 5.1.3.2 |

**V 2.8 Wymagania dotyczące pojedynczego weryfikatora wieloczynnikowego lub jednoczynnikowego**

Hasła jednorazowe (OTP) to fizyczne lub miękkie tokeny, które wyświetlają ciągle zmieniające się pseudolosowe jednorazowe wyzwanie.Urządzenia te utrudniają phishing (podszywanie się), ale nie jest to niemożliwe. Ten typ uwierzytelnienia jest uważany za „coś, co masz”. Tokeny wieloczynnikowe są podobne do jednoczynnikowych OTP, ale wymagają ważnego kodu PIN, odblokowania biometrycznego, wstawienia USB lub parowania NFC lub pewnej dodatkowej wartości (takiej jak kalkulatory podpisywania transakcji), aby wprowadzić ostateczny OTP.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.8.1 | Sprawdź, czy OTP oparte na czasie mają określony czas życia przed wygaśnięciem. | ✓ | ✓ | ✓ | 613 | 5.1.4.2 /  5.1.5.2 |
| 2.8.2 | Sprawdź, czy klucze symetryczne używane do weryfikacji przesłanych OTP są wysoce chronione, na przykład za pomocą sprzętowego modułu zabezpieczeń lub bezpiecznym magazynie kluczy opartym na systemie operacyjnym. |  | ✓ | ✓ | 320 | 5.1.4.2 /  5.1.5.2 |
| 2.8.3 | Sprawdź, czy zatwierdzone algorytmy kryptograficzne są używane do generowania, seedowania i weryfikacji. |  | ✓ | ✓ | 326 | 5.1.4.2 /  5.1.5.2 |
| 2.8.4 | Sprawdź, czy OTP oparte na czasie można wykorzystać tylko raz w okresie ważności. |  | ✓ | ✓ | 287 | 5.1.4.2 /  5.1.5.2 |
| 2.8.5 | Sprawdź, czy jeśli token wieloczynnikowy OTP opartego na czasie jest ponownie wykorzystywany w okresie ważności, jest on rejestrowany i odrzucany z bezpiecznymi powiadomieniami wysyłanymi do posiadacza urządzenia. |  | ✓ | ✓ | 287 | 5.1.5.2 |
| 2.8.6 | Sprawdź, czy fizyczny jednoczynnikowy generator OTP może zostać cofnięty w przypadku kradzieży lub innej straty. Upewnij się, że cofnięcie jest natychmiast skuteczne na wszystkich zalogowanych sesjach, niezależnie od lokalizacji. |  | ✓ | ✓ | 613 | 5.2.1 |
| 2.8.7 | Sprawdź, czy uwierzytelnianie biometryczne jest ograniczone do użycia tylko jako czynniki drugorzędne w połączeniu z czymś, co masz i czymś, co wiesz. |  |  | ✓ | 308 | 5.2.3 |

**V2.9: Wymagania dotyczące oprogramowania i urządzeń kryptograficznych**

Kryptograficzne klucze bezpieczeństwa to karty inteligentne lub klucze FIDO, w których użytkownik musi podłączyć lub sparować urządzenie kryptograficzne z komputerem, aby zakończyć uwierzytelnianie. Weryfikatorzy wysyłają wyzwanie do urządzeń kryptograficznych lub oprogramowania, a urządzenie lub oprogramowanie oblicza odpowiedź w oparciu o bezpiecznie przechowywany klucz kryptograficzny.

Wymagania dotyczące jednoczynnikowych urządzeń kryptograficznych i oprogramowania oraz wieloczynnikowych urządzeń kryptograficznych i oprogramowania są takie same, ponieważ weryfikacja uwierzytelniania kryptograficznego potwierdza posiadanie czynnika uwierzytelniania.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.9.1 | Sprawdź, czy klucze kryptograficzne używane podczas weryfikacji są bezpiecznie przechowywane i chronione przed ujawnieniem, na przykład przy użyciu modułu TPM lub HSM, lub usługi systemu operacyjnego, która może korzystać z tej bezpiecznej pamięci. |  | ✓ | ✓ | 320 | 5.1.7.2 |
| 2.9.2 | Sprawdź, czy wartość wyzwania (challenge nonce) ma długość co najmniej 64 bitów i jest statystycznie unikalna lub unikalna przez cały okres użytkowania urządzenia kryptograficznego. |  | ✓ | ✓ | 330 | 5.1.7.2 |
| 2.9.3 | Sprawdź, czy zatwierdzone algorytmy kryptograficzne są używane do generowania, seedingu i weryfikacji. |  | ✓ | ✓ | 327 | 5.1.7.2 |

**V2.10 Wymagania autoryzacji usługi**

Ta sekcja nie może być przetestowana używając testów penetracyjnych, więc nie ma żadnych wymagań L1. Jeśli jednak zostanie użyta w architekturze, kodowaniu lub bezpiecznym przeglądzie kodu, załóż, że oprogramowanie (podobnie jak Java Key Store) jest minimalnym wymaganiem w L1. Przechowywanie danych poufnych w clear text nie jest dopuszczalne w żadnych okolicznościach.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 2.10.1 | Sprawdź, czy tajemnice integracji nie opierają się na niezmienianych hasłach, takich jak klucze API lub współdzielone konta uprzywilejowane. | Wspoma-gane przez OS | ✓ | HSM | 287 | 5.1.1.1 |
| 2.10.2 | Sprawdź, czy jeśli hasła są wymagane, poświadczenia nie są kontem domyślnym. | Wspoma-gane przez OS | ✓ | HSM | 287 | 5.1.1.1 |
| 2.10.3 | Sprawdź, czy hasła są przechowywane w bezpieczny sposób, aby zapobiec atakom offline recovery, w tym dostępowi do systemu lokalnego. | Wspoma-gane przez OS | ✓ | HSM | 287 | 5.1.1.1 |
| 2.10.4 | Sprawdź, czy hasła, integracje z bazami danych i systemami zewnętrznymi, seedy i tajemnice wewnętrzne, oraz klucze API są zarządzane bezpiecznie i nie są zawarte w kodzie źródłowym lub przechowywane w repozytoriach kodu źródłowego. Takie przechowywanie POWINNO być odporne na ataki offline. Do przechowywania haseł zalecane jest użycie bezpiecznego magazynu kluczy oprogramowania (L1), modułu zaufanej platformy sprzętowej (TPM) lub sprzętowego modułu bezpieczeństwa (L3). | Wspoma-gane przez OS | ✓ | HSM | 523 | 5.1.1.1 |

**Dodatkowe Wymagania Agencji Amerykańskich**

Agencje amerykańskie mają obowiązkowe wymagania dotyczące NIST 800-63. Standard ASVS zawsze dotyczył 80% kontroli, które dotyczą prawie 100% aplikacji, a nie ostatnich 20% zaawansowanych kontroli lub tych, które mają ograniczone zastosowanie. ASVS jest ścisłym podzbiorem NIST 800-63, szczególnie dla klasyfikacji IAL1 / 2 i AAL1 / 2, ale nie jest wystarczająco obszerny, szczególnie w odniesieniu do klasyfikacji IAL3 / AAL3. Usilnie ponaglamy agencje rządowe USA do przeglądu i wdrożenia NIST 800-63 w całości.

**Słownik**

**CSP** - Credential Service Provider / Identity Provider - Dostawca usług poświadczeń / Dostawca tożsamości

**Authenticator** - Element uwierzytelniający - Kod który uwierzytelnia hasło, token, MFA, twierdzenie federacyjne (federated assertion), itp.

**Verifier** - Weryfikator - "Podmiot, który weryfikuje tożsamość wniskodawcy, weryfikując posiadanie i kontrolę przez wnioskodawcę jednego lub dwóch elementów uwierzytelniających za pomocą protokołu uwierzytelniania. W tym celu weryfikator może również wymagać potwierdzenia poświadczeń łączących element uwierzytelniający z identyfikatorem subskrybenta i sprawdzić ich statusy"

**OTP** - One-time password - hasło jednorazowe

**SFA** - Single factor authenticators - pojedyncze czynniki uwierzytelniające - takie jak coś co wiesz (zapamiętane sekrety, hasła, passphrase, PINy), coś czym jesteś (biometryka, odcisk palca, skan twarzy), lub coś co masz (tokeny OTP, urządzenie kryptograficzne jak na przykład smart card)

**MFA** - Multi factor authenticator, wielokrotne czynniki uwierzytelniające - zawierają 2 lub więcej SFA

Referencje:

• NIST 800-63 - Digital Identity Guidelines <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf>

• NIST 800-63 A - Enrollment and Identity Proofing <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63a.pdf>

• NIST 800-63 B - Authentication and Lifecycle Management <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf>

• NIST 800-63 C - Federation and Assertions <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63c.pdf>

• NIST 800-63 FAQ <https://pages.nist.gov/800-63-FAQ/>

• OWASP Testing Guide 4.0: Testing for Authentication <https://www.owasp.org/index.php/Testing_for_authentication>

• OWASP Cheat Sheet - Password storage <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md>

• OWASP Cheat Sheet - Forgot password <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Forgot_Password_Cheat_Sheet.md>

• OWASP Cheat Sheet - Choosing and using security questions <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Choosing_and_Using_Security_Questions_Cheat_Sheet.md>

**V3: Wymagania dotyczące weryfikacji zarządzania sesją**

Cel kontroli

Jednym z podstawowych komponentów dowolnej aplikacji internetowej lub pełnostanowego interfejsu API jest mechanizm, za pomocą którego kontroluje ona i utrzymuje stan użytkownika lub urządzenia, które z nim współpracuje. Zarządzanie sesją zmienia protokół bezstanowy na pełnostanowy, co ma kluczowe znaczenie dla odróżnienia różnych użytkowników lub urządzeń.

Upewnij się, że weryfikowana aplikacja spełnia następujące wymagania dotyczące zarządzania sesjami wysokiego poziomu:

• Sesje są unikalne dla każdej osoby i nie można ich odgadnąć ani udostępnić.

• Sesje są unieważniane, gdy nie są już potrzebne, a limit czasu upłynął podczas braku aktywności.

Jak wcześniej wspomniano, wymagania te zostały dostosowane do zgodnego podzbioru wybranych elementów NIST 800-63b, skupionych wokół wspólnych zagrożeń i powszechnie wykorzystywanych słabości uwierzytelniania. Poprzednie wymagania dotyczące weryfikacji zostały wycofane z użycia, pozbawione duplikatu lub w większości przypadków dostosowane tak, aby były ściśle powiązane z celami obowiązkowych wymagań NIST 800-63b.

**Wymagania dotyczące weryfikacji bezpieczeństwa**

**V3.1: Wymagania podstawowego zarządzania sesjami**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 3.1.1 | Sprawdź, czy aplikacja nigdy nie ujawnia tokenów sesji w parametrach URL lub komunikatach o błędach. | ✓ | ✓ | ✓ | 598 |  |

**V3.2: Wymagania dotyczące powiązania sesji**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 3.2.1 | Sprawdź, czy aplikacja generuje nowy token sesji na uwierzytelnianiu użytkownika. | ✓ | ✓ | ✓ | 384 | 7.1 |
| 3.2.2 | Sprawdź, czy tokeny sesji posiadają co najmniej 64 bity entropii. | ✓ | ✓ | ✓ | 331 | 7.1 |
| 3.2.3 | Sprawdź, czy aplikacja przechowuje tylko tokeny sesji w przeglądarce przy użyciu bezpiecznych metod, takich jak odpowiednio zabezpieczone pliki cookie (patrz sekcja 3.4) lub przechowywanie sesji HTML 5. | ✓ | ✓ | ✓ | 539 | 7.1 |
| 3.2.4 | Sprawdź, czy token sesji jest generowany przy użyciu zatwierdzonych algorytmów kryptograficznych. |  | ✓ | ✓ | 331 | 7.1 |

TLS lub inny bezpieczny kanał transportowy jest obowiązkowy dla zarządzania sesją. Jest to omówione w rozdziale Bezpieczeństwo komunikacji.

**V3.3: Wymagania dotyczące wylogowania i limitu czasu sesji**

Limity czasu sesji zostały dostosowane do NIST 800-63, co pozwala na znacznie dłuższe przekroczenie limitu czasu sesji niż tradycyjnie dopuszczają to standardy bezpieczeństwa. Organizacje powinny przejrzeć poniższą tabelę, a jeśli dłuższy okres czasu jest wymagany w oparciu o ryzyko aplikacji, wartość NIST powinna być górną granicą limitu czasu bezczynności sesji.

L1 w tym kontekście to IAL1 / AAL1, L2 to IAL2 / AAL3, L3 to IAL3 / AAL3. W przypadku IAL2 / AAL2 i IAL3 / AAL3 krótszy limit czasu bezczynności to dolna granica czasu bezczynności do wylogowania lub ponowne uwierzytelnienie w celu wznowienia sesji.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 3.3.1 | Sprawdź, czy wylogowanie i wygaśnięcie unieważnia token sesji, tak że przycisk Wstecz lub strona zależna niższego szczebla nie wznawia uwierzytelnionej sesji, w tym między stronami zaufanymi. | ✓ | ✓ | ✓ | 613 | 7.1 |
| 3.3.2 | Jeśli uwierzytelnianie zezwala użytkownikom na pozostanie zalogowanym, należy sprawdzić, czy ponowne uwierzytelnianie odbywa się okresowo zarówno w przypadku aktywnego użytkowania, jak i po okresie bezczynności. | 30 dni | 12 godzin lub 30 minut bezczynności, opcjonalnie 2FA | 12 godzin lub 15 minut bezczynności z 2FA | 613 | 7.2 |
| 3.3.3 | Sprawdź, czy aplikacja kończy wszystkie inne aktywne sesje po pomyślnej zmianie hasła i że jest to skuteczne w całej aplikacji, federacyjnym logowaniu (jeśli jest obecne) i wszystkich stronach zaufanych. |  | ✓ | ✓ | 613 |  |
| 3.3.4 | Sprawdź, czy użytkownicy są w stanie wyświetlić i wylogować się z dowolnej lub wszystkich aktualnie aktywnych sesji i urządzeń. |  | ✓ | ✓ | 613 | 7.1 |

**V 3.4 Zarządzanie sesją oparte na cookies**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 3.4.1 | Sprawdź, czy tokeny sesji oparte na plikach cookie mają ustawiony atrybut „Bezpieczny”. | ✓ | ✓ | ✓ | 614 | 7.1.1 |
| 3.4.2 | Sprawdź, czy tokeny sesji oparte na plikach cookie mają ustawiony atrybut „HttpOnly” | ✓ | ✓ | ✓ | 1004 | 7.1.1 |
| 3.4.3 | Sprawdź, czy tokeny sesji oparte na plikach cookie wykorzystują atrybut „SameSite”, aby ograniczyć ekspozycję na ataki CSRF. | ✓ | ✓ | ✓ | 16 | 7.1.1 |
| 3.4.4 | Sprawdź, czy tokeny sesji oparte na plikach cookie używają prefiksu „\_\_Host-”, aby zapewnić poufność plików cookie sesji. | ✓ | ✓ | ✓ | 16 | 7.1.1 |
| 3.4.5 | Sprawdź, czy jeśli aplikacja jest publikowana pod nazwą domeny z innymi aplikacjami, które ustawiają lub wykorzystują pliki cookie sesji, które mogą zastąpić lub ujawnić pliki cookie sesji, ustaw atrybut ścieżki w tokenach sesji opartych na plikach cookie, korzystając z najbardziej precyzyjnej możliwej ścieżki. | ✓ | ✓ | ✓ | 16 | 7.1.1 |

Zarządzanie sesjami opartymi na tokenach obejmuje klucze JWT, OAuth, SAML i API. Klucze API są słabe i nie powinny być używane w nowym kodzie.

**V3.5: Zarządzanie sesją oparte na tokenach**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 3.5.1 | Zarządzanie sesjami opartymi na tokenach obejmuje klucze JWT, OAuth, SAML i API. Z wszystkich wymienionych klucze API są znane jako słabe i nie powinny być używane w nowym kodzie. |  | ✓ | ✓ | 290 | 7.1.2 |
| 3.5.2 | Sprawdź, czy aplikacja używa tokenów sesji, a nie statycznych haseł i kluczy API, z wyjątkiem starszych implementacji. |  | ✓ | ✓ | 798 |  |
| 3.5.3 | Sprawdź, czy bezstanowe tokeny sesji używają podpisów cyfrowych, szyfrowania i innych środków zaradczych w celu ochrony przed manipulacją, podpisywaniem danych (enveloping), atakami przez powtórzenie (replay), szyfrowaniem zerowym i atakami podstawiania kluczy. |  | ✓ | ✓ | 345 |  |

**V3.6: Ponowne uwierzytelnienie z federacyjnego zarządzania tożsamością lub potwierdzenia**

Ta sekcja jest dla tych, którzy piszą zaufane strony (RP - relying party) lub dla dostawców usług uwierzytelniających (CSP - credential service provider). Jeśli polegasz na kodzie implementującym te funkcje, upewnij się, że te problemy są poprawnie obsługiwane.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 3.6.1 | Sprawdź, czy strony zaufane określają maksymalny czas uwierzytelniania dla dostawców CSP i czy dostawcy CSP ponownie uwierzytelniają subskrybenta, jeśli nie korzystał z sesji w tym okresie. |  |  | ✓ | 613 | 7.2.1 |
| 3.6.2 | Sprawdź, czy dostawcy CSP informują strony zaufane o ostatnim zdarzeniu uwierzytelniającym, aby umożliwić im ustalenie, czy muszą ponownie uwierzytelnić użytkownika. |  |  | ✓ | 613 | 7.2.1 |

**V3.7 Ochrona przeciwko expliotom zarządzania sesją**

Istnieje niewielka liczba ataków na zarządzanie sesjami, niektóre związane z doświadczeniem użytkownika (UX) sesji. Wcześniej, w oparciu o wymagania ISO 27002, ASVS wymagało blokowania wielu jednoczesnych sesji. Blokowanie równoczesnych sesji nie jest już odpowiednie, nie tylko ze względu na to, że współcześni użytkownicy mają wiele urządzeń, albo aplikacja jest API bez sesji przeglądarki, ale w większości tych implementacji wygrywa ostatni element uwierzytelniający, którym często jest atakujący. Ta sekcja zawiera najważniejsze wskazówki dotyczące powstrzymywania, opóźniania i wykrywania ataków zarządzania sesjami za pomocą kodu.

Opis ataku półotwartego (half-open attack)

Na początku 2018 kilka instytucji finansowych zostało zaatakowanych sposobem który atakujący nazywali „atakami półotwartymi”. Ta nazwa pozostała w branży. Atakujący uderzyli w wiele instytucji z różnymi zastrzeżonymi bazami kodowymi, i rzeczywiście wydaje się, że różne bazy kody wewnątrz instytucji... [niedokończone zdanie w oryginale]. Atak półotwarty wykorzystuje wadę wzorca projektowego powszechnie występującą w wielu istniejących systemach uwierzytelniania, zarządzania sesjami i kontroli dostępu.

Atakujący rozpoczynają atak półotwarty, próbując zablokować, zresetować lub odzyskać poświadczenie. Popularny wzór projektu zarządzania sesją ponownie wykorzystuje obiekty sesji / modele profilu użytkownika między nieuwierzytelnionym, półuwierzytelnionym (resetowanie hasła, zapomniana nazwa użytkownika) i w pełni uwierzytelnionym kodem. Ten wzorzec projektowy wypełnia prawidłowy obiekt sesji lub token zawierający profil ofiary, w tym hashe haseł i role. Jeśli kontrolki kontroli dostępu w kontrolerach lub routerach nie weryfikują poprawnie, że użytkownik jest w pełni zalogowany, osoba atakująca będzie mogła działać jako użytkownik. Ataki mogą obejmować zmianę hasła użytkownika na znaną wartość, zmianę adresu e-mail by zresetować hasło, wyłączenie MFA, lub rejestrację nowego urządzenia MFA, ujawnienie lub zmianę kluczy API i tak dalej.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** | **NIST** |
| 3.7.1 | Sprawdź, czy aplikacja zapewnia prawidłową sesję logowania lub wymaga drugorzędnego uwierzytelnienia lub weryfikacji wtórnej przed zezwoleniem na jakiekolwiek wrażliwe transakcje lub modyfikacje konta. | ✓ | ✓ | ✓ | 778 |  |

Referencje:

• OWASP Testing Guide 4.0: Session Management Testing <https://www.owasp.org/index.php/Testing_for_Session_Management>

• OWASP Session Management Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md>

• Set-Cookie \_\_Host- prefix details <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie#Directives>

**V4: Wymagania weryfikacji kontroli dostępu**

Cel:

Autoryzacja zezwala na dostęp do zasobów tylko tym, którzy są uprawnieni do ich używania. Upewnij się, że weryfikowana aplikacja spełnia następujące wymagania wysokiego poziomu:

* Osoby uzyskujące dostęp do zasobów posiadają do tego aktualne dane uwierzytelniające.
* Użytkownicy są powiązani z dobrze zdefiniowanym zestawem ról i uprawnień.
* Metadane roli i uprawnień są chronione przed powtórzeniami (replay) lub manipulacjami.

**Wymagania dotyczące weryfikacji bezpieczeństwa**

**V4.1: Ogólne zasady projektowania kontroli dostępu**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 4.1.1 | Sprawdź, czy aplikacja wymusza reguły kontroli dostępu na zaufanej warstwie usług, zwłaszcza jeśli kontrola dostępu po stronie klienta jest obecna i mogłaby zostać pominięta. | ✓ | ✓ | ✓ | 602 |
| 4.1.2 | Sprawdź, czy wszystkie atrybuty użytkowników i danych oraz informacje o polityce używane przez kontrolę dostępu nie mogą być zmanipulowane przez użytkowników końcowych, chyba że zostaną specjalnie upoważnieni. | ✓ | ✓ | ✓ | 639 |
| 4.1.3 | Sprawdź, czy istnieje zasada najniższych uprawnień - użytkownicy powinni mieć dostęp tylko do funkcji, plików danych, adresów URL, kontrolerów, usług i innych zasobów, dla których posiadają określone uprawnienia. Oznacza to ochronę przed spoofingiem i zwiększanie uprawnień. | ✓ | ✓ | ✓ | 285 |
| 4.1.4 | Sprawdź, czy domyślnie istnieje zasada odmowy, zgodnie z którą nowi użytkownicy/role rozpoczynaja z minimalnymi lub żadnymi uprawnieniami, a użytkownicy/role nie otrzymują dostępu do nowych funkcji, dopóki dostęp nie zostanie wyraźnie przypisany. | ✓ | ✓ | ✓ | 276 |
| 4.1.5 | Sprawdź, czy kontrola dostępu zawiesza się w sposób bezpieczny, w tym także w przypadku wystąpienia wyjątku. | ✓ | ✓ | ✓ | 285 |

**V4.2 Kontrola dostępu na poziomie operacji**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 4.2.1 | Sprawdź, czy wrażliwe dane i interfejsy API są chronione przed bezpośrednimi atakami obiektowymi, kierowanymi w tworzenie, czytanie, aktualizowanie i usuwanie rekordów, takimi jak tworzenie lub aktualizowanie cudzego rekordu, przeglądanie rekordów wszystkich lub usuwanie wszystkich rekordów. | ✓ | ✓ | ✓ | 639 |
| 4.2.2 | Sprawdź, czy aplikacja lub framework wymusza silny mechanizm anty-CSRF w celu ochrony uwierzytelnionej funkcjonalności, a skuteczna anty-automatyzacja lub anty-CSRF chroni nieuwierzytelnione funkcje. | ✓ | ✓ | ✓ | 352 |

**V4.3: Pozostałe kwestie związane z kontrolą dostępu**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 4.3.1 | Sprawdź, czy interfejsy administracyjne używają odpowiedniego uwierzytelniania wieloczynnikowego, aby zapobiec nieautoryzowanemu użyciu. | ✓ | ✓ | ✓ | 419 |
| 4.3.2 | Sprawdź, czy przeglądanie katalogów jest wyłączone, chyba że jest to pożądane. Ponadto aplikacje nie powinny zezwalać na wyszukiwanie lub ujawnianie metadanych plików lub katalogów, takich jak foldery Thumbs.db, .DS\_Store, .git lub .svn. | ✓ | ✓ | ✓ | 548 |
| 4.3.3 | Sprawdź, czy aplikacja posiada dodatkową autoryzację (taką jak uwierzytelnianie stopniowe lub adaptacyjne) dla systemów o niższej wartości i/lub podział obowiązków dla aplikacji o wysokiej wartości w celu egzekwowania kontroli mających zapobiegać oszustwom zgodnie z ryzykiem aplikacji i oszustwami z przeszłości. |  | ✓ | ✓ | 732 |

Referencje:

• OWASP Testing Guide 4.0: Authorization <https://www.owasp.org/index.php/Testing_for_Authorization>

• OWASP Cheat Sheet: Access Control <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Access_Control_Cheat_Sheet.md>

• OWASP CSRF Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.md>

• OWASP REST Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/REST_Security_Cheat_Sheet.md>

**V5 Wymagania dotyczące walidacji, oczyszczania i weryfikacji kodowania**

Cel:

Najczęstszą słabością zabezpieczeń aplikacji internetowych jest brak prawidłowej walidacji danych wejściowych pochodzących od klienta lub środowiska zanim zostaną one użyte bez kodowania danych wyjściowych. Ta podatność prowadzi do prawie wszystkich znaczących luk w aplikacjach internetowych, takich jak XSS, SQL injection, iniekcja interpretera, ataki locale / Unicode, ataki systemu plików i przepełnienia bufora.

Upewnij się, że zweryfikowana aplikacja spełnia następujące wymagania wysokiego poziomu:

• Walidacja danych wejściowych i architektura kodowania wyjściowego mają uzgodnioną kolejkę, aby zapobiec atakom wstrzykiwania.

• Dane wejściowe są silnie typowane, zatwierdzane, sprawdzane pod kątem zakresu lub długości, lub w najgorszym przypadku są oczyszczane lub filtrowane.

• Dane wyjściowe są kodowane lub zbierane zgodnie z kontekstem danych tak blisko interpretera, jak to możliwe.

Dzięki nowoczesnej architekturze aplikacji internetowych kodowanie danych wyjściowych jest ważniejsze niż kiedykolwiek. Trudno jest zapewnić solidne sprawdzanie poprawności danych wejściowych w pewnych scenariuszach, więc użycie bezpieczniejszego API, takiego jak sparametryzowane zapytania, auto-escaping templating frameworks, lub starannie wybranego kodowania wyjściowego, ma kluczowe znaczenie dla bezpieczeństwa aplikacji.

**V5.1: Wymagania dotyczące walidacji danych wejściowych**

Odpowiednio zaimplementowane kontrole walidacji danych wejściowych, z wykorzystaniem pozytywnej białej listy (ang. white list) i silnego typowania danych, mogą wyeliminować ponad 90% wszystkich ataków typu injection. Kontrola czasu trwania i zasięgu może jeszcze bardziej to zmniejszyć. Konieczność sprawdzania poprawności danych wejściowych jest wymagana w architekturze aplikacji, design sprints, kodowaniu oraz testowaniu jednostek i integracji. Chociaż wielu z tych elementów nie można znaleźć w testach penetracyjnych, następstwa ich niewdrożenia można znalexć w wersji V5.3 - Kodowanie wyjściowe i wymagania dotyczące zapobiegania injection. Zaleca się, aby programiści i osoby dokonujące przeglądu bezpieczeństwa kodu traktowali tę sekcję tak, jakby L1 był wymagany dla wszystkich elementów w celu zapobiegania injection.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 5.1.1 | Sprawdź, czy aplikacja posiada zabezpieczenia przed atakami zanieczyszczeń parametrów HTTP, szczególnie jeśli struktura aplikacji nie rozróżnia źródła parametrów żądania (GET, POST, cookies, nagłówków lub zmiennych środowiskowych). | ✓ | ✓ | ✓ | 235 |
| 5.1.2 | Sprawdź, czy frameworki chronią przed atakami “mass assignment” lub czy aplikacja ma środki zaradcze w celu ochrony przed niebezpiecznym przypisaniem parametrów, takim jak oznaczanie pól jako prywatne lub podobne. | ✓ | ✓ | ✓ | 915 |
| 5.1.3 | Sprawdź, czy wszystkie dane wejściowe (pola formularza HTML, żądania REST, parametry URL, nagłówki HTTP, pliki cookie, pliki wsadowe, kanały RSS itp.) są sprawdzane za pomocą pozytywnej walidacji (whitelisting). | ✓ | ✓ | ✓ | 20 |
| 5.1.4 | Sprawdź, czy dane ustrukturyzowane są silnie wpisane/wprowadzone i sprawdzane pod kątem zdefiniowanego schematu, w tym dozwolonych znaków, czasu trwania i wzoru (np. numerów kart kredytowych lub telefonu, lub sprawdzenia, czy dwa pola są powiązane racjonalnie, np. sprawdzanie zgodności lokalizacji przedmieść i kodu pocztowego) | ✓ | ✓ | ✓ | 20 |
| 5.1.5 | Sprawdź, czy URL przekierowuje i przekazuje tylko do miejsc docelowych z białej listy lub pokazuje ostrzeżenie podczas przekierowywania do potencjalnie niezaufanych treści. | ✓ | ✓ | ✓ | 601 |

**V5.2 Wymagania oczyszczania i sandboxingu**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 5.2.1 | Sprawdź, czy wszystkie niezaufane dane wejściowe HTML z edytorów WYSIWYG są odpowiednio oczyszczone za pomocą biblioteki HTML sanitizer lub funkcji frameworku. | ✓ | ✓ | ✓ | 116 |
| 5.2.2 | Sprawdź, czy dane nieustrukturyzowane są oczyszczane, aby wymusić środki bezpieczeństwa, takie jak dozwolone znaki i długość. | ✓ | ✓ | ✓ | 138 |
| 5.2.3 | Sprawdź, czy aplikacja oczyszcza dane wejściowe wprowadzone przez użytkownika przed przekazaniem do systemów pocztowych w celu ochrony przed wstrzyknięciem SMTP lub IMAP. | ✓ | ✓ | ✓ | 147 |
| 5.2.4 | Sprawdź, czy aplikacja unika użycia funkcji eval() lub innych funkcji dynamicznego wykonywania kodu. Tam, gdzie nie ma alternatywy, wszelkie dane wejściowe użytkownika, które zostaną uwzględnione, muszą zostać oczyszczone lub sandboxowane przed wykonaniem. | ✓ | ✓ | ✓ | 95 |
| 5.2.5 | Sprawdź, czy aplikacja chroni przed atakami wstrzykiwania szablonów, upewniając się, że dane wejściowe użytkownika są oczyszczane lub sandboxowane. | ✓ | ✓ | ✓ | 94 |
| 5.2.6 | Sprawdź, czy aplikacja chroni przed atakami SSRF, weryfikując lub oczyszczając niezaufane dane lub metadane pliku HTTP, takie jak nazwy plików i pola URL danych wejściowych, użyj białej listy protokołów, domen, ścieżek i portów. | ✓ | ✓ | ✓ | 918 |
| 5.2.7 | Sprawdź, czy aplikacja oczyszcza, wyłącza lub sandboxuje dostarczaną przez użytkownika zawartość skryptową SVG, zwłaszcza w odniesieniu do XSS wynikającego ze skryptów wbudowanych (inline) i foreignObject. | ✓ | ✓ | ✓ | 159 |
| 5.2.8 | Sprawdź, czy aplikacja oczyszcza, wyłącza lub izoluje treść skryptową lub treść języka szablonów wyrażeń dostarczonych przez użytkownika, takich jak Markdown, CSS lub XSL, BBCode lub podobne. | ✓ | ✓ | ✓ | 94 |

**V 5.3 Wymagania dotyczące kodowania danych wyjściowych i zapobiegania wstrzyknięciu**

Kodowanie danych wyjściowych blisko lub w sąsiedztwie używanego interpretera ma kluczowe znaczenie dla bezpieczeństwa dowolnej aplikacji. Zazwyczaj kodowanie wyjściowe nie jest trwałe, ale używane do zabezpieczenia danych wyjściowych w odpowiednim kontekście wyjściowym do natychmiastowego użycia. Niezakodowanie danych wyjściowych spowoduje że stworzona aplikacja będzie niebezpieczna, podatna na wstrzyknięcia i niezabezpieczona.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 5.3.1 | Sprawdź, czy kodowanie wyjściowe jest istotne dla wymaganego interpretera i kontekstu. Na przykład użyj koderów specjalnie dla wartości HTML, atrybutów HTML, JavaScript, parametrów URL, nagłówków HTTP, SMTP i innych, w zależności od kontekstu, zwłaszcza z niezaufanych danych wejściowych (np. Nazwy z Unicode lub apostrofy, takie jak ね こ lub O'Hara) | ✓ | ✓ | ✓ | 116 |
| 5.3.2 | Sprawdź, czy kodowanie danych wyjściowych zachowuje wybrany zestaw znaków i ustawienia regionalne użytkownika, tak aby każdy punkt znakowy Unicode był prawidłowy i bezpiecznie obsługiwany. | ✓ | ✓ | ✓ | 176 |
| 5.3.3 | Sprawdź, czy escaping danych wyjściowych kontekstowy, najlepiej zautomatyzowany - lub co najmniej manualny- chronić przed odbijanym, przechowywanym i opartym na DOM atakiem XSS. | ✓ | ✓ | ✓ | 79 |
| 5.3.4 | Sprawdź, czy wybór danych lub zapytania do bazy danych (np. SQL, HQL, ORM, NoSQL) używają sparametryzowanych zapytań, ORM, entity frameworks, lub są w inny sposób chronione przed atakami database injection. | ✓ | ✓ | ✓ | 89 |
| 5.3.5 | Sprawdź, czy tam, gdzie nie ma sparametryzowanych lub bezpieczniejszych mechanizmów, stosowane jest kodowanie danych wyjściowych specyficzne dla kontekstu w celu ochrony przed atakami wstrzyknięcia, takimi jak użycie SQL escaping w celu ochrony przed wstrzyknięciem SQL. | ✓ | ✓ | ✓ | 89 |
| 5.3.6 | Sprawdź, czy aplikacja chroni przed atakami wstrzyknięcia JavaScript lub JSON, w tym przed atakami eval, zdalnmi JavaScript includes, obejściami CSP, DOM XSS i oceną wyrażenia JavaScript. | ✓ | ✓ | ✓ | 830 |
| 5.3.7 | Sprawdź, czy aplikacja chroni przed lukami związanymi z wstrzykiwaniem LDAP, lub czy zaimplementowano określone mechanizmy bezpieczeństwa, aby zapobiec wstrzykiwaniu LDAP. | ✓ | ✓ | ✓ | 943 |
| 5.3.8 | Sprawdź, czy aplikacja chroni przed wstrzykiwaniem poleceń systemu operacyjnego (OS command) i czy wywołania systemu operacyjnego używają sparametryzowanych zapytań systemu operacyjnego lub czy używają kodowania kontekstowego wiersza poleceń danych wyjściowych. | ✓ | ✓ | ✓ | 78 |
| 5.3.9 | Sprawdź, czy aplikacja chroni przed atakami Local File Inclusion (LFI) lub Remote File Inclusion (RFI). | ✓ | ✓ | ✓ | 829 |
| 5.3.10 | Sprawdź, czy aplikacja chroni przed wstrzyknięciem XPath lub atakami wstrzyknięcia XML. | ✓ | ✓ | ✓ | 643 |

Uwaga: Używanie sparametryzowanych zapytań lub escaping SQL nie zawsze jest wystarczające; nazw tabel i kolumn, ORDER BY itd. nie można pominąć. Załączenie danych pominiętych (escaped) przez użytkownika w tych polach powoduje nieudane zapytania lub wstrzyknięcie SQL.

Uwaga: Format SVG wyraźnie zezwala na skrypt ECMA w prawie wszystkich kontekstach, więc zablokowanie wszystkich wektorów SVG XSS może nie być możliwe. Jeśli wymagane jest przesyłanie SVG, zdecydowanie zalecamy obsługiwanie tych przesłanych plików jako text / plain lub użycie oddzielnej domeny treści dostarczonej przez użytkownika, aby uniemożliwić pomyślne przejęcie aplikacji przez XSS.

**V5.4: Wymagania dotyczące pamięci, ciągów i niezarządzanego kodu**

Następujące wymagania będą miały zastosowanie tylko wtedy, gdy aplikacja używa języka systemowego lub kodu niezarządzanego.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 5.4.1 | Sprawdź, czy aplikacja używa ciągu bezpiecznego dla pamięci, bezpieczniejszego kopiowania pamięci i arytmetycznego wskaźnika, aby wykryć lub zapobiec przepełnieniu stosu, bufora lub sterty. |  | ✓ | ✓ | 120 |
| 5.4.2 | Sprawdź, czy ciągi formatu (ang. Format strings) nie przyjmują potencjalnie wrogich danych wejściowych i czy są stałe. |  | ✓ | ✓ | 134 |
| 5.4.3 | Sprawdź, czy techniki sprawdzania poprawności znaku, zakresu i wejścia są używane w celu zapobiegania przekroczenia zakresu liczb całkowitych. |  | ✓ | ✓ | 190 |

**V5.5: Wymagania dotyczące zapobiegania deserializacji**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 5.5.1 | Sprawdź, czy serializowane obiekty używają kontroli integralności lub są szyfrowane, aby zapobiec wrogiemu tworzeniu obiektów lub manipulowaniu danymi. | ✓ | ✓ | ✓ | 502 |
| 5.5.2 | Sprawdź, czy aplikacja poprawnie ogranicza parsery XML aby używana była tylko najbardziej restrykcyjna konfiguracja i upewnij się, że niebezpieczne funkcje, takie jak rozwiązywanie podmiotów zewnętrznych (ang. External Entity), są wyłączone, aby zapobiec XXE. | ✓ | ✓ | ✓ | 611 |
| 5.5.3 | Sprawdź, czy uniknięto deserializacji niezaufanych danych lub czy są one chronione zarówno w kodzie niestandardowym, jak i bibliotekach zewnętrznych (takich jak parsery JSON, XML i YAML). | ✓ | ✓ | ✓ | 502 |
| 5.5.4 | Sprawdź czy podczas parsowania JSONa w przeglądarkach lub backendach opartych na JavaScript, JSON.parse jest używany do parsowania dokumentu JSON. Nie używaj eval() do parsowania JSONa. | ✓ | ✓ | ✓ | 95 |

Referencje:

• OWASP Testing Guide 4.0: Input Validation Testing <https://www.owasp.org/index.php/Testing_for_Input_Validation>

• OWASP Cheat Sheet: Input Validation <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md>

• 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://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.md>

• 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://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md>

• OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.md>

• OWASP Java Encoding Project <https://www.owasp.org/index.php/OWASP_Java_Encoder_Project>

• OWASP Mass Assignment Prevention Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Mass_Assignment_Cheat_Sheet.md>

• DOMPurify - Client-side HTML Sanitization Library <https://github.com/cure53/DOMPurify>

• XML External Entity (XXE) Prevention Cheat Sheet) <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.md>

Więcej na temat auto escaping:

• Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems <https://security.googleblog.com/2009/03/reducing-xss-by-way-of-automatic.html>

• AngularJS Strict Contextual Escaping <https://docs.angularjs.org/api/ng/service/$sce>

• AngularJS ngBind <https://docs.angularjs.org/api/ng/directive/ngBind>

• Angular Sanitization <https://angular.io/guide/security#sanitization-and-security-contexts>

• Angular Template Security <https://angular.io/guide/template-syntax#content-security>

• ReactJS Escaping <https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks>

• Improperly Controlled Modification of Dynamically-Determined Object Attributes <https://cwe.mitre.org/data/definitions/915.html>

For more information on deserialization, please see:

• OWASP Deserialization Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Deserialization_Cheat_Sheet.md>

• OWASP Deserialization of Untrusted Data Guide <https://www.owasp.org/index.php/Deserialization_of_untrusted_data>

**V6: Wymagania dotyczące przechowywanej kryptografii**

Cel:

Upewnij się, że weryfikowana aplikacja spełnia następujące wymagania wysokiego poziomu:

* Wszystkie moduły kryptograficzne ulegają awarii w bezpieczny sposób i obsługa błędów jest przeprowadzana poprawnie.
* Używany jest odpowiedni generator liczb losowych.
* Dostęp do kluczy jest zarządzany w bezpieczny sposób.

**V6.1: Klasyfikacja danych**

Najważniejszym aspektem są dane przetwarzane, przechowywane lub przesyłane przez aplikację. Zawsze wykonuj ocenę wpływu prywatności, aby prawidłowo zidentyfikować potrzeby ochrony wszystkich przechowywanych danych.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 6.1.1 | Sprawdź czy regulowane prawnie dane prywatne są przechowywane jako zaszyfrowane podczas gdy są nieaktywne (at rest), dotyczy to danych osobowych (PII), wrażliwych danych osobowych lub danych, które podlegają unijnej dyrektywie RODO. |  | ✓ | ✓ | 311 |
| 6.1.2 | Sprawdź czy regulowane prawnie dane dotyczące zdrowia są przechowywane jako zaszyfrowane podczas gdy są nieaktywne (at rest), dotyczy to dokumentacji medycznej, szczegółów dotyczących urządzeń medycznych lub pozbawionych anonimowości danych badawczych. |  | ✓ | ✓ | 311 |
| 6.1.3 | Sprawdź czy regulowane prawnie dane finansowe są przechowywane jako zaszyfrowane podczas gdy są nieaktywne (at rest), dotyczy to kont bankowych, zaległości w spłacie lub historii kredytowej, rejestrów podatkowych lub pozbawionych anonimowości danych badawczych i rynkowych. |  | ✓ | ✓ | 311 |

**V 6.2 Algorytmy**

Ostatnie postępy w kryptografii oznaczają, że wcześniej bezpieczne algorytmy i długości kluczy nie są już bezpieczne ani wystarczające do ochrony danych. Dlatego powinna istnieć możliwość zmiany algorytmów.

Chociaż ta sekcja nie jest łatwa do przetestowania używając testów penetracyjnych, programiści powinni uważać tę całą sekcję za obowiązkową, mimo że w większości elementów brakuje L1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 6.2.1 | Sprawdź, czy wszystkie moduły kryptograficzne ulegają awariom w bezpieczny sposób, a błędy są obsługiwane w sposób, który nie pozwala na ataki Padding Oracle. | ✓ | ✓ | ✓ | 310 |
| 6.2.2 | Sprawdź, czy używane są sprawdzone lub zatwierdzone przez rząd algorytmy kryptograficzne, tryby i biblioteki, zamiast niestandardowo kodowanej kryptografii. |  | ✓ | ✓ | 327 |
| 6.2.3 | Sprawdź, czy wektor inicjalizacji szyfrowania, konfiguracja szyfrów i tryby blokowania są skonfigurowane bezpiecznie przy użyciu najnowszych porad. |  | ✓ | ✓ | 326 |
| 6.2.4 | Sprawdź, czy algorytmy losowej liczby, szyfrowania lub hashowania, długości kluczy, cykle (rounds), szyfry lub tryby, mogą być w dowolnym momencie ponownie skonfigurowane, zaktualizowane lub zamienione, aby chronić się przed przerwami kryptograficznymi. |  | ✓ | ✓ | 326 |
| 6.2.5 | Sprawdź, czy znane niebezpieczne tryby blokowania (np. EBC itp.), padding modes (np. PKCS # 1 v1.5, itp.), szyfry o małych rozmiarach bloków (np. Triple-DES, Blowfish itp.), oraz słabe algorytmy hashowania (tj. MD5, SHA1 itp.) nie są używane, chyba że są wymagane do wstecznej kompatybilności. |  | ✓ | ✓ | 326 |
| 6.2.6 | Sprawdź, czy nonces, wektory inicjalizacyjne i inne numery jednorazowe nie mogą być używane więcej niż raz z danym kluczem szyfrowania. Metoda generowania musi być odpowiednia dla używanego algorytmu. |  | ✓ | ✓ | 326 |
| 6.2.7 | Sprawdź, czy zaszyfrowane dane są uwierzytelniane za pomocą podpisów, uwierzytelnionych trybów szyfrowania, lub HMAC, aby upewnić się, że zaszyfrowany tekst nie zostanie zmieniony przez nieupoważnioną osobę. |  |  | ✓ | 326 |
| 6.2.8 | Sprawdź, czy wszystkie operacje kryptograficzne są constatnt-time, bez operacji 'short-circuit' w porównaniach, obliczeniach lub zwrotach, aby uniknąć wycieku informacji. |  |  | ✓ | 385 |

**V 6.3 Losowe wartości**

Prawdziwa generacja liczb pseudolosowych (PRNG) jest niezwykle trudna do uzyskania. Ogólnie rzecz biorąc, dobre źródła entropii w systemie zostaną szybko wyczerpane, jeśli będą nadmiernie wykorzystywane, ale źródła o mniejszej przypadkowości mogą prowadzić do przewidywalnych kluczy i tajemnic.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 6.3.1 | Sprawdź, czy wszystkie losowe liczby, losowe nazwy plików, losowe GUID, i losowe ciągi są generowane przy użyciu zatwierdzonego kryptograficznie bezpiecznego generatora liczb losowych modułu kryptograficznego, gdy te losowe wartości mają być niemożliwe do odgadnięcia przez atakującego. |  | ✓ | ✓ | 338 |
| 6.3.2 | Sprawdź, czy losowe GUID są tworzone przy użyciu algorytmu GUID v4 i zabezpieczonego kryptograficznie generatora liczb pseudolosowych (CSPRNG). GUID utworzone przy użyciu innych generatorów liczb pseudolosowych mogą być przewidywalne. |  | ✓ | ✓ | 338 |
| 6.3.3 | Sprawdź, czy losowe liczby są tworzone z odpowiednią entropią, nawet gdy aplikacja jest pod dużym obciążeniem, lub że aplikacja ulega degradacji w bezpieczny sposób w takich okolicznościach. |  |  | ✓ | 338 |

**V 6.4 Zarządzanie sekretami (hasłami)**

Chociaż ta sekcja nie jest łatwa do przetestowania używając testów penetracyjnych, programiści powinni uważać tę całą sekcję za obowiązkową, mimo że w większości elementów brakuje L1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 6.4.1 | Sprawdź, czy rozwiązanie do zarządzania tajemnicami (hasłami), takie jak magazyn kluczy, jest używane do bezpiecznego tworzenia, przechowywania, kontrolowania dostępu i niszczenia tajemnic (haseł). |  | ✓ | ✓ | 798 |
| 6.4.2 | Sprawdź, czy kluczowy materiał nie jest widoczny dla aplikacji, ale zamiast tego korzysta z izolowanego modułu zabezpieczeń, takiego jak sejf (vault) dla operacji kryptograficznych. |  | ✓ | ✓ | 320 |

Referencje

• 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://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cryptographic_Storage_Cheat_Sheet.md>

• FIPS 140-2 <https://csrc.nist.gov/publications/detail/fips/140/2/final>

**V7 Wymagania weryfikacji obsługi i rejestracji błędów**

Cel kontroli

Podstawowym celem obsługi i rejestrowania błędów jest dostarczenie przydatnych informacji dla użytkownika, administratorów i zespołów reagowania na incydenty. Celem tutaj nie jest tworzenie ogromnych ilości logów, ale logów wysokiej jakości, które zawierają więcej sygnałów, niż odrzucanego hałasu.

Logi wysokiej jakości często zawierają poufne dane i muszą być chronione zgodnie z lokalnymi przepisami lub dyrektywami dotyczącymi prywatności danych. Powinno to obejmować:

• Niegromadzenie lub nierejestrowanie poufnych informacji, chyba że jest to wymagane.

• Zapewnienie, że wszystkie zarejestrowane informacje są bezpiecznie obsługiwane i chronione zgodnie z klasyfikacją danych.

• Zapewnienie, że logi nie są przechowywane na zawsze, ale mają możliwie najkrótszy czas życia. Jeśli logi zawierają prywatne lub poufne dane, których definicja różni się w poszczególnych krajach, logi stają się jednymi z najbardziej wrażliwych informacji przechowywanych przez aplikację, a zatem są bardzo atrakcyjne dla atakujących.

Ważne jest również, aby upewnić się, że jeśli aplikacja ulegnie awarii to odbędzie się to w sposób bezpieczny i błędy nie będą ujawniać niepotrzebnych informacji.

**V 7.1 Wymagania treści logów**

Rejestrowanie poufnych informacji jest niebezpieczne - logi same stają się poufne co oznacza, że muszą być szyfrowane, podlegać zasadom przechowywania i muszą być ujawniane podczas audytów bezpieczeństwa. Upewnij się, że tylko niezbędne informacje są przechowywane w logach i na pewno nie zawierają żadnych informacji o płatnościach, poświadczeniach (w tym tokenów sesji), poufnych lub osobistych informacji.

V7.1 Obejmuje OWASP Top 10 2017: A10. Ponieważ 2017: A10 i ta sekcja nie są mogą być przetestowane z użyciem testów penetracyjnych, to ważne żeby:

• Deweloperzy zapewnili pełną zgodność z tą sekcją, tak jakby wszystkie elementy były oznaczone jako L1

• Testerzy penetracji potwierdzili pełną zgodność wszystkich elementów w wersji 7.1 za pośrednictwem wywiadu, zrzutów ekranu lub zapewnienia.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 7.1.1 | Sprawdź, czy aplikacja nie rejestruje poświadczeń lub szczegółów płatności. Tokeny sesji powinny być przechowywane tylko w logach w nieodwracalnej, hashowanej formie. | ✓ | ✓ | ✓ | 532 |
| 7.1.2 | Sprawdź, czy aplikacja nie rejestruje innych poufnych danych zgodnie z lokalnymi przepisami dotyczącymi prywatności lub odpowiednią polityką bezpieczeństwa. | ✓ | ✓ | ✓ | 532 |
| 7.1.3 | Sprawdź, czy aplikacja rejestruje zdarzenia związane z bezpieczeństwem, w tym udane i nieudane zdarzenia uwierzytelniania, awarie kontroli dostępu, błędy deserializacji i błędy sprawdzania poprawności danych wejściowych. |  | ✓ | ✓ | 778 |
| 7.1.4 | Sprawdź, czy każde zdarzenie dziennika zawiera niezbędne informacje, które pozwolą na szczegółowe zbadanie osi czasu, gdy zdarzenie ma miejsce. |  | ✓ | ✓ | 778 |

**V 7.2 Wymagania związane z przetwarzaniem logów**

Rejestrowanie w odpowiednim czasie ma kluczowe znaczenie dla zdarzeń audytowych, selekcji (triage) i eskalacji. Upewnij się, że logi aplikacji są przejrzyste i można je łatwo monitorować i analizować lokalnie lub logować przeniesione do zdalnego systemu monitorowania.

V7.2 Obejmuje OWASP Top 10 2017: A10. Ponieważ 2017: A10 i ta sekcja nie są mogą być przetestowane z użyciem testów penetracyjnych, to ważne żeby:

• Deweloperzy zapewnili pełną zgodność z tą sekcją, tak jakby wszystkie elementy były oznaczone jako L1

• Testerzy penetracji potwierdzili pełną zgodność wszystkich elementów w wersji 7.2 za pośrednictwem wywiadu, zrzutów ekranu lub zapewnienia.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 7.2.1 | Sprawdź, czy wszystkie decyzje dotyczące uwierzytelniania są rejestrowane, bez zapisywania poufnych identyfikatorów sesji lub haseł. Powinno to obejmować zapytania z odpowiednimi metadanymi potrzebnymi do dochodzeń związanych z bezpieczeństwem. |  | ✓ | ✓ | 778 |
| 7.2.2 | Sprawdź, czy wszystkie decyzje dotyczące kontroli dostępu mogą być rejestrowane, a wszystkie nieudane decyzje są rejestrowane. Powinno to obejmować zapytania z odpowiednimi metadanymi potrzebnymi do dochodzeń związanych z bezpieczeństwem. |  | ✓ | ✓ | 285 |

**V7.3 Wymagania związane z ochroną logów**

Logi, które mogą być z łatwością zmodyfikowane lub usunięte są bezużyteczne dla dochodzeń i postępowań karnych. Ujawnienie dzienników może ujawniać wewnętrzne szczegóły dotyczące aplikacji lub zawartych w niej danych. Należy zachować ostrożność podczas ochrony dzienników przed nieuprawnionym ujawnieniem, modyfikacją lub usunięciem.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 7.3.1 | Sprawdź, czy aplikacja odpowiednio koduje dane dostarczone przez użytkownika, aby zapobiec wstrzyknięciu dziennika. |  | ✓ | ✓ | 117 |
| 7.3.2 | Sprawdź, czy wszystkie zdarzenia są chronione przed wstrzykiwaniem podczas przeglądania w programie do przeglądania logów. |  | ✓ | ✓ | 117 |
| 7.3.3 | Sprawdź czy logi bezpieczeństwa są chronione przed nieautoryzowanym dostępem i modyfikacją. |  | ✓ | ✓ | 200 |
| 7.3.4 | Sprawdź, czy źródła czasu (time sources) są zsynchronizowane z prawidłową godziną i strefą czasową. Poważnie rozważ rejestrowanie tylko w UTC, jeśli systemy są globalne, aby pomóc w analizie kryminalistycznej po incydencie. |  | ✓ | ✓ |  |

Uwaga: Kodowanie logów(7.3.1) jest trudne do przetestowania i przeglądu za pomocą zautomatyzowanych narzędzi dynamicznych i testów penetracyjnych, ale architekci, programiści i recenzenci kodu źródłowego powinni uznać to za wymaganie L1.

**V7.4 Obsługa błędów**

Celem obsługi błędów jest umożliwienie aplikacji dostarczenia zdarzeń istotnych dla bezpieczeństwa w celu monitorowania, selekcji i eskalacji. Celem nie jest tworzenie logów. Podczas rejestrowania zdarzeń związanych z bezpieczeństwem należy upewnić się, że tworzenie danego loga ma cel i że można go rozpoznać za pomocą oprogramowania SIEM lub innych programów służących do analizy.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 7.4.1 | Sprawdź, czy ogólny komunikat jest wyświetlany, gdy wystąpi nieoczekiwany lub wrażliwy błąd bezpieczeństwa, potencjalnie z unikalnym ID, które może być użyte przez dział supportu do zbadania. | ✓ | ✓ | ✓ | 210 |
| 7.4.2 | Sprawdź, czy obsługa wyjątków (lub ich odpowiednik funkcjonalny) jest używana w bazie kodu, aby uwzględnić oczekiwane i nieoczekiwane warunki błędów. |  | ✓ | ✓ | 544 |
| 7.4.3 | Sprawdź, czy zdefiniowano error handler typu „Ostatnia deska ratunku”, która przechwyci wszystkie nieobsłużone wyjątki. |  | ✓ | ✓ | 460 |

Uwaga: Niektóre języki, takie jak Swift i Go - i poprzez powszechną praktykę projektową - również wiele języków funkcjonalnych, nie obsługuje wyjątków ani handlerów typu “ostatnia deska ratunku”. W takim przypadku architekci i programiści powinni używać wzorca, języka lub przyjaznego dla frameworku sposobu, aby aplikacje mogły bezpiecznie obsługiwać wyjątkowe, nieoczekiwane lub związane z bezpieczeństwem zdarzenia.

Referencje:

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

**V8: Wymagania dotyczące weryfikacji ochrony danych**

Cel:

Istnieją trzy kluczowe elementy dobrej ochrony danych: poufność, integralność i dostępność (CIA). Standard ten zakłada, że ochrona danych jest egzekwowana na zaufanym systemie, takim jak serwer, który został wzmocniony (hardening) i ma wystarczającą ochronę.

Aplikacje muszą zakładać, że wszystkie urządzenia użytkownika są w jakiś sposób narażone. Gdy aplikacja przesyła lub przechowuje poufne informacje na temat niezabezpieczonych urządzeń, takich jak współdzielone komputery, telefony i tablety, aplikacja jest odpowiedzialna za zapewnienie, że dane przechowywane na tych urządzeniach są szyfrowane i nie można ich uzyskać, zmienić lub ujawnić w łatwy sposób.

Upewnij się, że weryfikowana aplikacja spełnia następujące wymagania ochrony danych na wysokim poziomie:

* Poufność: dane powinny być chronione przed nieuprawnionym wglądem lub ujawnieniem podczas przesyłania jak i przechowywania.
* Integralność: dane powinny być chronione przed złośliwym tworzeniem, zmianą lub usunięciem przez nieautoryzowanych użytkowników.
* Dostępność: dane powinny być dostępne dla autoryzowanych użytkowników zgodnie z wymogami.

**V8.1: Ogólna ochrona danych**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 8.1.1 | Sprawdź czy aplikacja chroni wrażliwe dane przed cachowaniem w komponentach serwera takich jak load balancer i pamięć podręczna aplikacji. |  | ✓ | ✓ | 524 |
| 8.1.2 | Sprawdź, czy wszystkie cachowane lub tymczasowe kopie poufnych danych przechowywanych na serwerze są chronione przed nieautoryzowanym dostępem lub oczyszczone / unieważnione po uzyskaniu dostępu do poufnych danych przez autoryzowanego użytkownika. |  | ✓ | ✓ | 524 |
| 8.1.3 | Sprawdź, czy aplikacja minimalizuje liczbę parametrów w żądaniu, takich jak ukryte pola, zmienne Ajax, pliki cookie i wartości nagłówków. |  | ✓ | ✓ | 233 |
| 8.1.4 | Sprawdź, czy aplikacja może wykryć i powiadomić o nienormalnej liczbie żądań, takich jak adres IP, użytkownika, suma na godzinę lub dzień, lub cokolwiek ma sens dla aplikacji. |  | ✓ | ✓ | 770 |
| 8.1.5 | Sprawdź, czy wykonywane są regularne kopie zapasowe ważnych danych i czy wykonano testowe przywracanie danych. |  |  | ✓ | 19 |
| 8.1.6 | Sprawdź, czy kopie zapasowe są bezpiecznie przechowywane, aby zapobiec kradzieży lub uszkodzeniu danych. |  |  | ✓ | 19 |

**V8.2: Ochrona danych po stronie klienta**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 8.2.1 | Sprawdź, czy aplikacja ustawia wystarczającą liczbę nagłówków chroniących przed cachingiem, aby wrażliwe dane nie były cachowane w nowoczesnych przeglądarkach. | ✓ | ✓ | ✓ | 525 |
| 8.2.2 | Sprawdź, czy dane przechowywane po stronie klienta (takie jak pamięć lokalna HTML5, pamięć sesji, indeksowana baza danych, zwykłe pliki cookie lub pliki cookie Flash) nie zawierają danych wrażliwych lub danych PII. | ✓ | ✓ | ✓ | 922 |
| 8.2.3 | Sprawdź, czy uwierzytelnione dane są usuwane z pamięci klienta, takiej jak przeglądarka DOM, po zakończeniu działania klienta lub sesji. | ✓ | ✓ | ✓ | 922 |

**V8.3: Wrażliwe dane prywatne**

Ta sekcja pomaga chronić wrażliwe dane przed tworzeniem, odczytywaniem, aktualizowaniem lub usuwaniem bez autoryzacji, szczególnie w sposób masowy.

Zgodność z tą sekcją oznacza zgodność z “V4 Kontrola Dostępu”, a w szczególności z V4.2. Na przykład, aby chronić się przed nieautoryzowanymi aktualizacjami lub ujawnieniem poufnych danych osobowych, należy przestrzegać V4.2.1. Prosimy o stosowanie się do warunków tej sekcji i V4 w celu uzyskania pełnej ochrony.

Uwaga: Przepisy i regulacje dotyczące prywatności, takie jak Australijskie Zasady Ochrony Prywatności APP-11 lub RODO, mają bezpośredni wpływ na sposób, w jaki aplikacje muszą podejść do przechowywania, używania i przekazywania poufnych danych osobowych. Zapoznaj się z lokalnymi przepisami i regulacjami, a w razie potrzeby skonsultuj się z wykwalifikowanym specjalistą ds. prywatności lub prawnikiem.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 8.3.1 | Sprawdź, czy wrażliwe dane są wysyłane na serwer w treści wiadomości HTTP lub nagłówkach, a parametry query string z dowolnego czasownika HTTP nie zawierają danych wrażliwych. | ✓ | ✓ | ✓ | 319 |
| 8.3.2 | Sprawdź, czy użytkownicy mają metodę usuwania lub eksportowania swoich danych na żądanie. | ✓ | ✓ | ✓ | 212 |
| 8.3.3 | Sprawdź, czy użytkownicy otrzymują w jasny sposób informacje dotyczące gromadzenia i wykorzystywania dostarczonych danych osobowych oraz czy użytkownicy wyrazili zgodę na korzystanie z tych danych przed ich wykorzystaniem w jakikolwiek sposób. | ✓ | ✓ | ✓ | 285 |
| 8.3.4 | Sprawdź, czy wszystkie wrażliwe dane utworzone i przetworzone przez aplikację zostały zidentyfikowane i upewnij się, że istnieje polityka dotycząca postępowania z wrażliwymi danymi. | ✓ | ✓ | ✓ | 200 |
| 8.3.5 | Sprawdź, czy dostęp do poufnych danych jest audytowany (bez rejestrowania samych danych wrażliwych), czy dane są gromadzone zgodnie z odpowiednimi dyrektywami o ochronie danych lub gdzie wymagane jest rejestrowanie dostępu. |  | ✓ | ✓ | 532 |
| 8.3.6 | Sprawdź, czy poufne informacje zawarte w pamięci są nadpisywane za pomocą zer lub losowych danych, gdy tylko poufne dane nie będą już potrzebne do złagodzenia ataków zrzutu pamięci (memory dumping attack). |  | ✓ | ✓ | 226 |
| 8.3.7 | Sprawdź, czy poufne lub prywatne informacje są szyfrowane za pomocą zatwierdzonych algorytmów zapewniających poufność i integralność. |  | ✓ | ✓ | 327 |
| 8.3.8 | Sprawdź, czy wrażliwe dane osobowe podlegają klasyfikacji w zakresie przechowywania danych, tak że stare lub nieaktualne dane są usuwane automatycznie, zgodnie z harmonogramem lub w zależności od sytuacji. |  | ✓ | ✓ | 285 |

Myśląc o ochronie danych, należy przede wszystkim rozważyć masową ekstrakcję lub modyfikację lub nadmierne użycie. Na przykład wiele systemów mediów społecznościowych pozwala użytkownikom dodawać 100 nowych znajomych dziennie, ale nie jest ważny system, z którego pochodzą te żądania. Platforma bankowa może chcieć zablokować powyżej 5 transakcji na godzinę, w której przekazywane jest ponad 1000 euro instytucjom zewnętrznym. Wymagania każdego systemu mogą być bardzo różne, więc decyzja o „nieprawidłowościach” musi uwzględniać model zagrożenia i ryzyko biznesowe. Ważnymi kryteriami są zdolność do wykrywania, odstraszania lub blokowania takich nieprawidłowych działań masowych.

Referencje:

• Consider using Security Headers website to check security and anti-caching headers <https://securityheaders.com/>

• OWASP Secure Headers project <https://www.owasp.org/index.php/OWASP_Secure_Headers_Project>

• OWASP Privacy Risks Project <https://www.owasp.org/index.php/OWASP_Top_10_Privacy_Risks_Project>

• OWASP User Privacy Protection Cheat Sheet <https://www.owasp.org/index.php/User_Privacy_Protection_Cheat_Sheet>

• European Union General Data Protection Regulation (GDPR) overview <https://edps.europa.eu/data-protection_en>

• European Union Data Protection Supervisor - Internet Privacy Engineering Network <https://edps.europa.eu/data-protection/ipen-internet-privacy-engineering-network_en>

**V9 Wymagania weryfikacji komunikacji**

Cel kontroli

Upewnij się, że zweryfikowana aplikacja spełnia następujące wymagania wysokiego poziomu:

• TLS lub silne szyfrowanie jest zawsze używane, niezależnie od wrażliwości przesyłanych danych

• Najnowsze, wiodące porady dotyczące konfiguracji służą do włączania i zamawiania preferowanych algorytmów i szyfrów

• Słabe lub wkrótce przestarzałe algorytmy i szyfry są zamawiane jako ostateczność

• Przestarzałe lub znane niezabezpieczone algorytmy i szyfry są wyłączone.

Wiodące porady w branży dotyczące bezpiecznej konfiguracji TLS zmienia się często, przeważnie ze względu na katastrofalne wady w istniejących algorytmach i szyfrach. Zawsze używaj najnowszych wersji narzędzi do konfiguracji konfiguracji TLS (takich jak SSLyze lub inne skanery TLS), aby skonfigurować preferowaną kolejność i wybór algorytmu. Konfiguracja powinna być okresowo sprawdzana, aby zapewnić, że bezpieczna konfiguracja komunikacji jest zawsze obecna i skuteczna.

**V9.1 Wymagania bezpieczeństwa komunikacji**

Cała komunikacja z klientem powinna odbywać się tylko zaszyfrowanymi ścieżkami komunikacji. W szczególności korzystanie z TLS 1.2 lub nowszego jest zasadniczo wymagane przez nowoczesne przeglądarki i wyszukiwarki. Konfiguracja powinna być regularnie weryfikowana za pomocą narzędzi online, aby upewnić się, że wdrożono najnowsze wiodące praktyki.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 9.1.1 | Sprawdź, czy zabezpieczony TLS jest używany do wszystkich połączeń klienta i nie powraca do niezabezpieczonych lub niezaszyfrowanych protokołów w przypadku trudności. | ✓ | ✓ | ✓ | 319 |
| 9.1.2 | Zweryfikuj, używając narzędzi online lub aktualnych narzędzi testujących TLS, że tylko silne algorytmy, szyfry i protokoły są włączone, z najsilniejszymi algorytmami i szyframi ustawionymi jako preferowane. | ✓ | ✓ | ✓ | 326 |
| 9.1.3 | Sprawdź, czy stare wersje protokołów SSL, TLS, algorytmów, szyfrów i konfiguracji są wyłączone, takie jak SSLv2, SSLv3 lub TLS 1.0 i TLS 1.1. Najnowsza wersja TLS powinna być preferowanym zestawem szyfrów. | ✓ | ✓ | ✓ | 326 |

**V9.2 Wymagania bezpieczeństwa komunikacji z serwerem**

Komunikacja z serwerem to coś więcej niż tylko HTTP. Bezpieczne połączenia do i od innych systemów, takich jak systemy monitorowania, narzędzia do zarządzania, dostęp zdalny i ssh, oprogramowanie pośrednie, bazy danych, komputery mainframe, systemy partnerskie lub zewnętrzne źródła - muszą być zastosowane. Wszystko to musi być zaszyfrowane, aby zapobiec sytuacji „trudne na zewnątrz, trywialnie łatwe do przechwycenia wewnątrz”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 9.2.1 | Sprawdź, czy połączenia do i od serwera korzystają z zaufanych certyfikatów TLS. Jeśli używane są certyfikaty generowane wewnętrznie lub samopodpisane, serwer musi być skonfigurowany tylko do zaufania określonym wewnętrznym CA i określonym samopodpisanym certyfikatom. Wszystkie pozostałe powinny zostać odrzucone. |  | ✓ | ✓ | 295 |
| 9.2.2 | Sprawdź, czy szyfrowana komunikacja, taka jak TLS, jest używana do wszystkich połączeń przychodzących i wychodzących, w tym do portów zarządzania, monitorowania, uwierzytelniania, API lub webservice calls, bazy danych, chmury, serwera, połączeń bezserwerowych, mainframe, zewnętrznych i partnerskich. Serwer nie może powrócić do niezabezpieczonych lub niezaszyfrowanych protokołów w razie napotkania trudności. |  | ✓ | ✓ | 319 |
| 9.2.3 | Sprawdź, czy uwierzytelniane są wszystkie zaszyfrowane połączenia z systemami zewnętrznymi, które zawierają poufne informacje lub funkcje. |  | ✓ | ✓ | 287 |
| 9.2.4 | Sprawdź, czy odpowiednie cofnięcie certyfikacji, takie jak Online Certificate Status Protocol (OCSP) Stapling, jest włączone i skonfigurowane. |  | ✓ | ✓ | 299 |
| 9.2.5 | Upewnij się że niepowodzenia połączeń backend TLS są rejestrowane. |  |  | ✓ | 544 |

Referencje:

• OWASP – TLS Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md>

• Mozilla's Server Side TLS <https://wiki.mozilla.org/Security/Server_Side_TLS>

**V10 Wymagania dotyczące weryfikacji złośliwego kodu**

Cel kontroli

Upewnij się, że kod spełnia następujące wymagania wysokiego poziomu:

• Złośliwe działanie jest obsługiwane bezpiecznie i prawidłowo, aby nie wpływać na resztę aplikacji.

• Nie ma time bombs ani innych ataków opartych na czasie.

• Nie wykonuje „phone home” do złośliwych lub nieautoryzowanych miejsc docelowych.

• Nie ma backdoor, easter eggs, salami attacks, rootkitów ani nieautoryzowanego kodu, który może być kontrolowany przez atakującego.

Znalezienie złośliwego kodu jest dowodem negatywności, którego nie można całkowicie zweryfikować. Należy dołożyć wszelkich starań, aby zapewnić, że kod nie ma wrodzonego złośliwego kodu lub niepożądanej funkcjonalności.

**V10.1 Kontrole integralności kodu**

Najlepszą obroną przed złośliwym kodem jest podejście „ufaj, ale weryfikuj”. Wprowadzenie nieautoryzowanego lub złośliwego kodu do kodu jest często przestępstwem kryminalnym w wielu jurysdykcjach. Polityki i procedury powinny jasno określać sankcje dotyczące złośliwego kodu.

Główni programiści powinni regularnie sprawdzać check-in kodu, w szczególności te, które mogą mieć dostęp do czasu, I/O lub funkcji sieciowych.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 10.1.1 | Sprawdź, czy używane jest narzędzie do analizy kodu, które może wykryć potencjalnie złośliwy kod, takie jak funkcje czasu, niebezpieczne operacje na plikach i połączenia sieciowe. |  |  | ✓ | 749 |

**V10.2 Wyszukiwanie złośliwego kodu**

Złośliwy kod jest niezwykle rzadki i trudny do wykrycia. Ręczny przegląd kodu wiersz po wierszu może pomóc w poszukiwaniu bomb logicznych, ale nawet najbardziej doświadczony recenzent kodów będzie miał trudności ze znalezieniem złośliwego kodu, nawet jeśli wie, że istnieje.

Przestrzeganie tej sekcji nie jest możliwe bez pełnego dostępu do kodu źródłowego, w tym bibliotek zewnętrznych.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 10.2.1 | Sprawdź, czy kod źródłowy aplikacji i bibliotek zewnętrznych nie zawierają nieautoryzowanych funkcji “phone home” lub gromadzenia danych. Jeśli taka funkcjonalność istnieje, uzyskaj zgodę użytkownika na jej działanie przed zebraniem jakichkolwiek danych. |  | ✓ | ✓ | 359 |
| 10.2.2 | Sprawdź, czy aplikacja nie żąda niepotrzebnych lub nadmiernych uprawnień do funkcji lub czujników związanych z prywatnością, takich jak kontakty, kamery, mikrofony lub lokalizacja. |  | ✓ | ✓ | 272 |
| 10.2.3 | Sprawdź, czy kod źródłowy aplikacji i bibliotek stron trzecich nie zawierają backdoors, takich jak zakodowane lub dodatkowe nieudokumentowane konta lub klucze, zaciemnianie kodu, nieudokumentowane obiekty binarne typu blob, rootkity lub anty-debugowanie, niezabezpieczone funkcje debugowania lub inne przestarzałe, niezabezpieczone lub ukryte funkcje, które mogą zostać użyte złośliwie, jeśli zostaną wykryte. |  |  | ✓ | 507 |
| 10.2.4 | Sprawdź, czy kod źródłowy aplikacji i biblioteki stron trzecich nie zawierają bomb czasowych, poprzez szukanie funkcji związanych z datą i godziną. |  |  | ✓ | 511 |
| 10.2.5 | Sprawdź, czy kod źródłowy aplikacji i biblioteki stron trzecich nie zawierają złośliwego kodu, takiego jak ataki salami, obejścia logiczne lub bomby logiczne. |  |  | ✓ | 511 |
| 10.2.6 | Sprawdź, czy kod źródłowy aplikacji i biblioteki stron trzecich nie zawierają easter eggów ani żadnej innej potencjalnie niepożądanej funkcjonalności. |  |  | ✓ | 507 |

**V10. 3 Wdrożone kontrole integralności aplikacji**

Po wdrożeniu aplikacji złośliwy kod nadal może być wstawiony. Aplikacje muszą chronić się przed typowymi atakami, takimi jak wykonywanie niepodpisanego kodu z niezaufanych źródeł i przejęcia subdomeny.

Przestrzeganie tej sekcji może być operacyjne i ciągłe.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 10.3.1 | Sprawdź, czy jeśli aplikacja ma funkcję automatycznej aktualizacji klienta lub serwera, aktualizacje powinny być pobierane przez bezpieczne kanały i podpisane cyfrowo. Kod aktualizacji musi zweryfikować podpis cyfrowy aktualizacji przed zainstalowaniem lub wykonaniem aktualizacji. | ✓ | ✓ | ✓ | 16 |
| 10.3.2 | Sprawdź, czy aplikacja wykorzystuje zabezpieczenia integralności, takie jak podpisywanie kodu lub integralność pod-zasobów. Aplikacja nie może ładować ani wykonywać kodu z niezaufanych źródeł, takich jak ładowanie includów, modułów, wtyczek, kodu lub biblioteki z niezaufanych źródeł lub Internetu. | ✓ | ✓ | ✓ | 353 |
| 10.3.3 | Sprawdź, czy aplikacja ma ochronę przed przejęciami subdomeny, jeśli aplikacja opiera się na wpisach DNS lub subdomenach DNS, takich jak wygasłe nazwy domen, nieaktualne wskaźniki DNS lub CNAME, projekty, których ważność wygasła w repozytoriach publicznych kodów źródłowych, lub przejściowe API chmury (transient cloud APIs), funkcje bez serwera lub storage buckets (autogen-bucket-id.cloud.example.com) lub podobne. Ochrona może obejmować zapewnienie, że nazwy DNS używane przez aplikacje są regularnie sprawdzane pod kątem wygaśnięcia lub zmiany. | ✓ | ✓ | ✓ | 350 |

Referencje:

• Hostile Sub-Domain Takeover, Detectify Labs <https://labs.detectify.com/2014/10/21/hostile-subdomain-takeover-using-herokugithubdesk-more/>

• Hijacking of abandoned subdomains part 2, Detectify Labs <https://labs.detectify.com/2014/12/08/hijacking-of-abandoned-subdomains-part-2/>

**V11: Wymagania dotyczące weryfikacji logiki biznesowej**

Cel:

Upewnij się, że weryfikowana aplikacja spełnia następujące wymagania wysokiego poziomu:

• Przepływ logiki biznesowej jest sekwencyjny, przetwarzany w kolejności i nie można go pominąć.

• Logika biznesowa obejmuje ograniczenia wykrywania i zapobiegania zautomatyzowanym atakom, takim jak ciągłe przekazywanie niewielkich środków pieniężnych lub dodawanie miliona znajomych za jednym razem, itd

• Przepływy logiki biznesowej o wysokiej wartości bierze pod uwagę przypadki nadużyć i podstępne podmioty i ma zabezpieczenia przed spoofingiem, sabotażem, odrzucaniem, ujawnianiem informacji i atakami, w których zwiększono uprawnienia.

**V11.1 Wymagania bezpieczeństwa logiki biznesowej**

Bezpieczeństwo logiki biznesowej jest tak bardzo indywidualne dla każdej aplikacji, że żadna uniwersalna lista kontrolna nigdy nie będzie miała zastosowania. Zabezpieczenia logiki biznesowej muszą być ‘wprojektowane’, aby chronić przed prawdopodobnymi zagrożeniami zewnętrznymi - nie można ich dodawać za pomocą firewalli aplikacji ani bezpiecznej komunikacji. Zalecamy stosowanie modelowania zagrożeń podczas sprintu projektowego, na przykład przy użyciu OWASP Cornucopia lub podobnych narzędzi.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 11.1.1 | Sprawdź, czy aplikacja będzie przetwarzać przepływy logiki biznesowej tylko dla tego samego użytkownika w porządku sekwencyjnym i bez pomijania kroków. | ✓ | ✓ | ✓ | 841 |
| 11.1.2 | Sprawdź, czy aplikacja będzie przetwarzać tylko przepływy logiki biznesowej ze wszystkimi etapami przetwarzanymi w realistycznym czasie ludzkim, tj. transakcje nie są przesyłane zbyt szybko. | ✓ | ✓ | ✓ | 779 |
| 11.1.3 | Sprawdź, czy aplikacja ma odpowiednie limity dla określonych działań biznesowych lub transakcji, które są poprawnie egzekwowane w zależności od użytkownika. | ✓ | ✓ | ✓ | 770 |
| 11.1.4 | Sprawdź, czy aplikacja ma wystarczającą kontrolę przeciw automatyzacji, aby wykryć i chronić przed eksfiltracją danych, nadmiernymi żądaniami logiki biznesowej, nadmiernym przesyłaniem plików lub atakami DoS. | ✓ | ✓ | ✓ | 770 |
| 11.1.5 | Sprawdź, czy aplikacja posiada ograniczenia logiki biznesowej lub walidację w celu ochrony przed prawdopodobnymi zagrożeniami lub ryzykiem biznesowym, zidentyfikowanymi przy użyciu modelowania zagrożeń lub podobnych metodologii. | ✓ | ✓ | ✓ | 841 |
| 11.1.6 | Sprawdź, czy aplikacja ma problemy z „time of check to time of use” (TOCTOU) lub innymi race conditions dla wrażliwych operacji. |  | ✓ | ✓ | 367 |
| 11.1.7 | Sprawdź, czy aplikacja monitoruje nietypowe zdarzenia lub działania z perspektywy logiki biznesowej. Na przykład, próby wykonania działań poza kolejnością lub działania, których normalny użytkownik nigdy by nie próbował. |  | ✓ | ✓ | 754 |
| 11.1.8 | Sprawdź, czy aplikacja ma konfigurowalne alarmowanie po wykryciu automatycznych ataków lub nietypowych działań. |  | ✓ | ✓ | 390 |

Referencje:

• 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>

• OWASP AppSensor <https://www.owasp.org/index.php/OWASP_AppSensor_Project>

• OWASP Automated Threats to Web Applications <https://www.owasp.org/index.php/OWASP_Automated_Threats_to_Web_Applications>

• OWASP Cornucopia <https://www.owasp.org/index.php/OWASP_Cornucopia>

**V12: Wymagania dotyczące weryfikacji plików i zasobów**

Cel:

Upewnij się, że weryfikowana aplikacja spełnia następujące wymagania wysokiego poziomu:

• Niezaufane dane pliku powinny być traktowane odpowiednio i w bezpieczny sposób.

• Niezaufane dane pliku uzyskane z niezaufanych źródeł są przechowywane poza głównym katalogiem WWW i mają ograniczone uprawnienia.

**V12.1: Wymagania dotyczące przesyłania plików**

Chociaż bomby dekompresyjne zip są szczególnie testowane za pomocą technik testowania penetracji, są uważane za L2 i wyższe aby zachęcić do projektowania i rozwoju przy starannym ręcznym testowaniu oraz w celu uniknięcia automatycznego lub niewykwalifikowanego ręcznego testowania penetracji warunku odmowy usługi.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 12.1.1 | Sprawdź, czy aplikacja nie akceptuje dużych plików, które mogłyby zapełnić pamięć lub spowodować atak typu „odmowa usługi”. | ✓ | ✓ | ✓ | 400 |
| 12.1.2 | Sprawdź, czy skompresowane pliki są sprawdzane pod kątem „bomby zip” - małe pliki wejściowe, które po rozpakowaniu stają się ogromnymi plikami potrafiącymi wyczerpać limity przechowywania plików. |  | ✓ | ✓ | 409 |
| 12.1.3 | Sprawdź, czy wymuszany jest limit rozmiaru pliku i maksymalna liczba plików na użytkownika, aby upewnić się, że pojedynczy użytkownik nie może wypełnić pamięci zbyt dużą ilością plików lub plikami o zbyt dużych rozmiarach. |  | ✓ | ✓ | 770 |

**V12.2 Wymagania integralności plików**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 12.2.1 | Sprawdź, czy pliki uzyskane z niezaufanych źródeł są sprawdzane, czy ich typ zgadza się z typem pliku, który sugeruje jego zawartość. |  | ✓ | ✓ | 434 |

**V12.3 Wymagania dotyczące wykonywania plików**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 12.3.1 | Sprawdź, czy przesłane przez użytkownika metadane ‘filename’ nie są używane bezpośrednio z plikiem systemowym lub plikiem frameworku i API URL w celu ochrony przed path traversal. | ✓ | ✓ | ✓ | 22 |
| 12.3.2 | Sprawdź, czy przesłane przez użytkownika metadane ‘filename’ są sprawdzane lub ignorowane, aby zapobiec ujawnieniu, utworzeniu, aktualizacji lub usunięciu lokalnych plików (LFI). | ✓ | ✓ | ✓ | 73 |
| 12.3.3 | Sprawdź, czy przesłane przez użytkownika metadane ‘filename’ są sprawdzane lub ignorowane, aby zapobiec ujawnieniu lub wykonaniu zdalnych plików (RFI), co może również prowadzić do SSRF. | ✓ | ✓ | ✓ | 98 |
| 12.3.4 | Sprawdź, czy aplikacja chroni przed reflective file download (RFD), sprawdzając poprawność lub ignorując nazwy plików przesłane przez użytkownika w parametrze JSON, JSONP lub URL, nagłówek Content-Type odpowiedzi powinien być ustawiony na text / plain, a nagłówek Content-Disposition powinien mieć stałą nazwę pliku. | ✓ | ✓ | ✓ | 641 |
| 12.3.5 | Sprawdź, czy metadane niezaufanego pliku nie są używane bezpośrednio z systemowym interfejsem API lub bibliotekami, aby zabezpieczyć się przed OS command injection. | ✓ | ✓ | ✓ | 78 |
| 12.3.6 | Sprawdź, czy aplikacja nie zawiera i nie wykonuje funkcji z niezaufanych źródeł, takich jak niezweryfikowane sieci dystrybucji treści, biblioteki JavaScript, biblioteki node npm, lub biblioteki DLL po stronie serwera. |  | ✓ | ✓ | 829 |

**V12.4: Wymagania dotyczące przechowywania plików**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 12.4.1 | Sprawdź, czy pliki uzyskane z niezaufanych źródeł są przechowywane poza głównym katalogiem WWW, z ograniczonymi uprawnieniami, najlepiej z silną walidacją. | ✓ | ✓ | ✓ | 922 |
| 12.4.2 | Sprawdź, czy pliki uzyskane z niezaufanych źródeł są skanowane przez skanery antywirusowe, aby zapobiec przesyłaniu znanych szkodliwych treści. | ✓ | ✓ | ✓ | 509 |

**V12.5: Wymagania dotyczące pobierania plików**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 12.5.1 | Sprawdź, czy poziom warstwy webowej jest skonfigurowany do obsługi tylko plików z określonymi rozszerzeniami, aby zapobiec wyciekom niewłaściwych informacji i kodu źródłowego. Na przykład pliki kopii zapasowych (np. .bak), tymczasowe pliki robocze (np. .swp), skompresowane pliki (np. .zip,.tar.gz itp.) i inne rozszerzenia powszechnie używane przez edytorów powinny być blokowane, chyba że jest wymagane inaczej. | ✓ | ✓ | ✓ | 552 |
| 12.5.2 | Sprawdź, czy bezpośrednie żądania przesłanych plików nigdy nie będą wykonywane jako zawartość w formacie HTML/JavaScript. | ✓ | ✓ | ✓ | 434 |

**V12.6: Wymagania ochrony SSRF**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 12.6.1 | Sprawdź, czy serwer WWW lub aplikacji jest skonfigurowany z białą listą (whitelist) zasobów lub systemów, do których serwer może wysyłać żądania lub z których ładować dane/pliki. | ✓ | ✓ | ✓ | 918 |

Referencje:

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

• Reflective file download by Oren Hafif <https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/reflected-file-download-a-new-web-attack-vector/>

• OWASP Third Party JavaScript Management Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.md>

**V13: Wymagania dotyczące weryfikacji API i usług sieciowych**

Cel:

Upewnij się, że weryfikowana aplikacja korzystająca z zaufanej warstwy usług API (zwykle używającej JSON lub XML lub GraphQL) ma:

• Odpowiednie uwierzytelnianie, zarządzanie sesjami i autoryzacje wszystkich usług internetowych.

• Weryfikacje poprawności danych wejściowych dla wszystkich parametrów, które przechodzą z niższego do wyższego poziomu zaufania.

• Skuteczne kontrole bezpieczeństwa dla wszystkich typów API, w tym chmury i API bez serwera.

Przeczytaj ten rozdział w połączeniu ze wszystkimi innymi rozdziałami z tego samego poziomu; nie powielamy problemów związanych z uwierzytelnianiem lub zarządzaniem sesjami API.

**V13.1 Ogólne wymagania dotyczące weryfikacji bezpieczeństwa web service**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 13.1.1 | Sprawdź, czy wszystkie komponenty aplikacji używają tych samych kodowań i parserów, aby uniknąć parsowania ataków wykorzystujących różne zachowanie URI lub parsowania plików, które mogłyby zostać wykorzystane w atakach SSRF i RFI. | ✓ | ✓ | ✓ | 116 |
| 13.1.2 | Sprawdź, czy dostęp do funkcji administracyjnych i zarządzania jest ograniczony do autoryzowanych administratorów. | ✓ | ✓ | ✓ | 419 |
| 13.1.3 | Sprawdź, czy adresy URL API nie ujawniają poufnych informacji, takich jak klucz API, tokeny sesji itp. | ✓ | ✓ | ✓ | 598 |
| 13.1.4 | Sprawdź, czy decyzje dotyczące autoryzacji są podejmowane zarówno w identyfikatorze URI, wymuszonym przez programowe lub deklaratywne zabezpieczenia na kontrolerze lub routerze, jak i na poziomie zasobów, wymuszane przez uprawnienia oparte na modelu. |  | ✓ | ✓ | 285 |
| 13.1.5 | Sprawdź, czy żądania zawierające nieoczekiwane lub brakujące typy treści są odrzucane za pomocą odpowiednich nagłówków (status odpowiedzi HTTP 406 Unacceptable lub 415 Unsupported Media Type). |  | ✓ | ✓ | 434 |

**V13.2: Wymagania dotyczące weryfikacji usługi sieci Web RESTful**

Sprawdzanie poprawności schematu JSON jest obecnie na wstępnym etapie standaryzacji. Rozważając użycie sprawdzania poprawności schematu JSON, które jest najlepszą praktyką dla usług sieciowych SOAP, rozważ użycie następujących dodatkowych strategii sprawdzania poprawności danych w połączeniu ze sprawdzaniem poprawności schematu JSON:

• Sprawdzanie poprawności parsowania obiektu JSON, na przykład w przypadku braku lub istnienia dodatkowych elementów.

• Walidacja wartości obiektu JSON przy użyciu standardowych metod sprawdzania poprawności danych wejściowych, takich jak typ danych, format danych, itp.

• i formalna walidacja schematu JSON.

Po sformalizowaniu standardu sprawdzania schematu JSON, ASVS zaktualizuje swoje porady w tym obszarze. Uważnie monitoruj wszelkie zmiany w schematach JSON, ponieważ będą one musiały być regularnie aktualizowane do czasu sformalizowania standardu i wyeliminowania błędów z implementacji referencyjnych.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 13.2.1 | Sprawdź, czy włączone metody HTTP RESTful są poprawnym wyborem dla użytkownika lub akcji i na przykład uniemożliwiają normalnym użytkownikom korzystanie z DELETE lub PUT na chronionym API lub zasobach. | ✓ | ✓ | ✓ | 650 |
| 13.2.2 | Sprawdź czy walidacja schematu JSON jest stosowana i weryfikowana przed zaakceptowaniem danych wejściowych. | ✓ | ✓ | ✓ | 20 |
| 13.2.3 | Sprawdź, czy usługi sieci Web RESTful, które wykorzystują pliki cookie, są chronione przed atakami CSRF poprzez użycie co najmniej jednego lub więcej z następujących elementów: potrójny lub podwójny wzór cookie, CSRF nonces lub sprawdzanie nagłówka żądania ORIGIN. | ✓ | ✓ | ✓ | 352 |
| 13.2.4 | Sprawdź, czy usługi REST mają zabezpieczenia przed automatyzacją w celu ochrony przed nadmiernymi wywołaniami, zwłaszcza jeśli interfejs API jest nieuwierzytelniony. |  | ✓ | ✓ | 779 |
| 13.2.5 | Sprawdź, czy usługi REST jawnie sprawdzają przychodzący Content-Type jako oczekiwany, jak np. application/xml lub application/JSON. |  | ✓ | ✓ | 436 |
| 13.2.6 | Sprawdź, czy nagłówki wiadomości i ładunki są godne zaufania i nie zostały zmodyfikowane podczas przesyłania. Wymaganie silnego szyfrowania dla transportu (tylko TLS) może być wystarczające w wielu przypadkach, ponieważ zapewnia zarówno poufność, jak i ochronę integralności. Podpisy cyfrowe dla poszczególnych wiadomości mogą zapewnić dodatkowe zabezpieczenie jako dodatek do zabezpieczeń obecnych podczas przesyłania dla aplikacji o wysokim poziomie bezpieczeństwa, ale przynoszą one dodatkową złożoność i potrzebę ocenę czy korzyści przewyższają ryzyko. |  | ✓ | ✓ | 345 |

**V13.3 Wymagania weryfikacji SOAP web service**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 13.3.1 | Sprawdź, czy walidacja schematu XSD ma miejsce, aby zapewnić prawidłowe utworzenie dokumentu XML, a następnie sprawdź poprawność każdego pola danych wejściowych, zanim nastąpi przetwarzanie tych danych. | ✓ | ✓ | ✓ | 20 |
| 13.3.2 | Sprawdź, czy payload wiadomości jest podpisany przy użyciu WS-Security, aby zapewnić rzetelny transport między klientem a usługą. |  | ✓ | ✓ | 345 |

Uwaga: Z powodu problemów z atakami XXE przeciwko DTD, walidacja DTD nie powinna być używana, a framework oceny DTD wyłączony zgodnie z wymaganiami określonymi w V14.

**V13.4 GraphQL i inne wymagania bezpieczeństwa warstwy danych web service**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 13.4.1 | Sprawdź, czy walidacja schematu XSD ma miejsce, aby zapewnić prawidłowe utworzenie dokumentu XML, a następnie sprawdź poprawność każdego pola danych wejściowych, zanim nastąpi przetwarzanie tych danych. |  | ✓ | ✓ | 770 |
| 13.4.2 | Sprawdź, czy payload wiadomości jest podpisany przy użyciu WS-Security, aby zapewnić rzetelny transport między klientem a usługą. |  | ✓ | ✓ | 285 |

Referencje:

• OWASP Serverless Top 10 <https://www.owasp.org/images/5/5c/OWASP-Top-10-Serverless-Interpretation-en.pdf>

• OWASP Serverless Project <https://www.owasp.org/index.php/OWASP_Serverless_Top_10_Project>

• 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://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.md#triple-submit-cookie>

• OWASP XML External Entity Prevention Cheat Sheet - General Guidance\* JSON Web Tokens (and Signing) <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.md#general-guidance>

• REST Security Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/REST_Security_Cheat_Sheet.md>

• JSON Schema <https://json-schema.org/specification.html>

• XML DTD Entity Attacks <https://www.vsecurity.com//download/publications/XMLDTDEntityAttacks.pdf>

• Orange Tsai - A new era of SSRF Exploiting URL Parser In Trending Programming Languages <https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf>

**V14 Wymagania weryfikacji konfiguracji**

Cel kontroli

Upewnij się, że zweryfikowana aplikacja ma:

• Bezpieczne, powtarzalne, zautomatyzowane środowisko kompilacji.

• Wzmocnione biblioteki stron trzecich, zarządzanie zależnościami i konfiguracją, tak by nieaktualne lub niezabezpieczone komponenty nie były uwzględniane przez aplikację.

• Bezpieczną konfigurację domyślną, tak aby administratorzy i użytkownicy musieli osłabić domyślny stan zabezpieczeń.

Konfiguracja aplikacji po wyjęciu z pudełka powinna być bezpieczna w Internecie, co oznacza bezpieczną konfigurację po wyjęciu z pudełka.

**V14.1 Kompilacja**

Kolejka kompilacji (build pipeline) jest podstawą powtarzalnego bezpieczeństwa - za każdym razem, gdy zostanie wykryte coś niebezpiecznego, można to rozwiązać w kodzie źródłowym, skryptach kompilacji lub wdrażania i przetestować automatycznie. Zdecydowanie zachęcamy do używania kolejek kompilacji z automatycznym sprawdzaniem bezpieczeństwa i zależności, które ostrzegają lub łamią kompilację (break the build), aby zapobiec wdrożeniu znanych problemów bezpieczeństwa do produkcji. Ręczne kroki wykonywane nieregularnie bezpośrednio prowadzą do możliwych do uniknięcia błędów bezpieczeństwa.

W miarę jak branża przechodzi na model DevSecOps, ważne jest zapewnienie ciągłej dostępności i integralności wdrożenia i konfiguracji w celu osiągnięcia stanu „known good”. W przeszłości, jeśli system został zhakowany, udowodnienie, że nie doszło do kolejnych włamań, zajęłoby dni lub miesiące. Dzisiaj, wraz z pojawieniem się infrastruktury definiowanej przez oprogramowanie, szybkich wdrożeń A / B z zerowym przestojem i zautomatyzowanych kontenerowych kompilacji, możliwe jest automatyczne i ciągłe budowanie, wzmacnianie i wdrażanie zamiennika „known good” dla dowolnego zagrożonego systemu.

Jeśli tradycyjne modele są nadal stosowane, należy podjąć ręczne kroki, aby wzmocnić i uzupełnić tę konfigurację, aby umożliwić szybką wymianę zagrożonych systemów na wysoce integralne i niezagrożone systemy.

Zgodność z tą sekcją wymaga zautomatyzowanego systemu kompilacji oraz dostępu do skryptów kompilacji i wdrażania.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 14.1.1 | Sprawdź, czy procesy tworzenia i wdrażania aplikacji są wykonywane w bezpieczny i powtarzalny sposób, taki jak automatyzacja CI / CD, automatyczne zarządzanie konfiguracją i zautomatyzowane skrypty wdrażania. |  | ✓ | ✓ |  |
| 14.1.2 | Sprawdź, czy flagi kompilatora są skonfigurowane by włączyć wszystkie dostępne zabezpieczenia i ostrzeżenia o przepełnieniu bufora, w tym do randomizacji stosu (stack randomization), zapobiegania wykonywaniu danych i do łamania kompilacji, jeśli zostanie znaleziony niebezpieczny wskaźnik (pointer), pamięć, ciąg formatu (format string), liczba całkowita lub operacje ciągu. |  | ✓ | ✓ | 120 |
| 14.1.3 | Sprawdź, czy konfiguracja serwera jest wzmocniona zgodnie z zaleceniami serwera aplikacji i używanych frameworków. |  | ✓ | ✓ | 16 |
| 14.1.4 | Sprawdź, czy aplikacja, konfiguracja i wszystkie zależności mogą zostać ponownie wdrożone za pomocą zautomatyzowanych skryptów wdrażania, zbudowanych z udokumentowanego i przetestowanego runbooka w rozsądnym czasie lub przywróconych z kopii zapasowych w odpowiednim czasie. |  | ✓ | ✓ |  |
| 14.1.5 | Sprawdź, czy autoryzowani administratorzy mogą zweryfikować integralność wszystkich konfiguracji związanych z bezpieczeństwem w celu wykrycia manipulacji. |  |  | ✓ |  |

**V14.2: Zależność**

Zarządzanie zależnością ma kluczowe znaczenie dla bezpiecznego działania dowolnej aplikacji dowolnego typu. Przestarzałe lub słabo zabezpieczone zależności są główną przyczyną największych i najdroższych ataków do tej pory.

Uwaga: Na poziomie 1, 14.2.1 zgodność dotyczy obserwacji lub wykrycia bibliotek i innych komponentów po stronie klienta, a nie dokładniejszej analizy statycznej kodu kompilacji lub analizy zależności. Te bardziej dokładne techniki mogą być wykrywane w razie potrzeby w drodze przesłuchania.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 14.2.1 | Sprawdź, czy wszystkie komponenty są aktualne, najlepiej używając narzędzia do sprawdzania zależności w kompilacji lub podczas kompilacji. | ✓ | ✓ | ✓ | 1026 |
| 14.2.2 | Sprawdź, czy wszystkie niepotrzebne funkcje, dokumentacja, sample, konfiguracje zostały usunięte, takie jak np. przykładowe aplikacje, dokumentacja platformy oraz domyślni lub przykładowi użytkownicy. | ✓ | ✓ | ✓ | 1002 |
| 14.2.3 | Sprawdź, czy jeśli zasoby aplikacji, takie jak biblioteki JavaScript, arkusze stylów CSS lub czcionki sieciowe, są hostowane zewnętrznie w sieci dostarczania treści (CDN) lub przez zewnętrznego dostawcę, Subresource Integrity (SRI) jest używana do sprawdzenia integralności tych zasobów. | ✓ | ✓ | ✓ | 714 |
| 14.2.4 | Sprawdź, czy komponenty firm zewnętrznych pochodzą ze wstępnie zdefiniowanych, zaufanych i stale obsługiwanych repozytoriów. |  | ✓ | ✓ | 829 |
| 14.2.5 | Sprawdź, czy prowadzony jest katalog wszystkich używanych bibliotek zewnętrznych. |  | ✓ | ✓ |  |
| 14.2.6 | Sprawdź, czy powierzchnia ataku jest zmniejszona poprzez sandboxing lub enkapsulację bibliotek zewnętrznych, aby umożliwić w aplikacji tylko wymagane zachowanie. |  | ✓ | ✓ | 265 |

**V14.3 Wymagania dotyczące niezamierzonego ujawnienia bezpieczeństwa**

Konfiguracje produkcji powinny zostać wzmocnione, aby chronić się przed typowymi atakami, takimi jak konsole do debugowania, podnieść poprzeczkę dla XSS i RFI, a także wyeliminować trywialne „podatności” w odkrywaniu informacji, które są niepożądaną cechą wielu raportów testów penetracyjnych.

Wiele z tych problemów rzadko jest ocenianych jako znaczące ryzyko, ale są one powiązane z innymi lukami. Jeśli te problemy nie występują domyślnie, to podnosi to poprzeczkę, zanim większość ataków się uda.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 14.3.1 | Sprawdź, czy komunikaty o błędach serwera Web, serwera aplikacji i frameworku są skonfigurowane tak, aby dostarczać wartościowych dla użytkownika niestandardowych odpowiedzi w celu wyeliminowania wszelkich niezamierzonych ujawnień zabezpieczeń. | ✓ | ✓ | ✓ | 209 |
| 14.3.2 | Sprawdź, czy tryby debugowania serwera web lubserwera aplikacji i frameworku aplikacji są wyłączone w produkcji, aby wyeliminować funkcje debugowania, konsole programistów i niezamierzone ujawnienia zabezpieczeń. | ✓ | ✓ | ✓ | 497 |
| 14.3.3 | Sprawdź, czy nagłówki HTTP lub dowolna część odpowiedzi HTTP nie ujawniają szczegółowych informacji o wersji komponentów systemu. | ✓ | ✓ | ✓ | 200 |

**V14.4 Wymagania bezpieczeństwa nagłówków**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 14.4.1 | Sprawdź, czy każda odpowiedź HTTP zawiera nagłówek typu zawartości określający bezpieczny zestaw znaków (np. UTF-8, ISO 8859-1). | ✓ | ✓ | ✓ | 173 |
| 14.4.2 | Sprawdź, czy wszystkie odpowiedzi API zawierają Content-Disposition: attachment; filename = "api.json" (lub inna odpowiednia nazwa pliku dla typu zawartości). | ✓ | ✓ | ✓ | 116 |
| 14.4.3 | Sprawdź, czy istnieje strategia bezpieczeństwa treści (CSPv2), która pomaga zmniejszyć wpływ ataków XSS, takich jak wstrzyknięcia HTML, DOM, JSON i JavaScript. | ✓ | ✓ | ✓ | 1021 |
| 14.4.4 | Sprawdź, czy wszystkie odpowiedzi zawierają X-Content-Type-Options: nosniff. | ✓ | ✓ | ✓ | 116 |
| 14.4.5 | Sprawdź, czy nagłówki HTTP Strict Transport Security są uwzględnione we wszystkich odpowiedziach i dla wszystkich poddomen, takich jak Strict-Transport-Security: max-age=15724800; includeSubdomains. | ✓ | ✓ | ✓ | 523 |
| 14.4.6 | Sprawdź, czy dołączony jest odpowiedni nagłówek "Referrer-Policy", jak "no-referrer" lub "same-origin". | ✓ | ✓ | ✓ | 116 |
| 14.4.7 | Sprawdź, czy odpowiedni nagłówek X-Frame-Options lub Content-Security-Policy: frame-ancestors jest używany w przypadku witryn, w których treść nie powinna być osadzona w witrynie zewnętrznej. | ✓ | ✓ | ✓ | 346 |

**V14.5 Wymagania walidacji nagłówka żądania HTTP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **L1** | **L2** | **L3** | **CWE** |
| 14.5.1 | Sprawdź, czy serwer aplikacji akceptuje tylko metody HTTP używane przez aplikację lub API, w tym pre-flight OPTIONS. | ✓ | ✓ | ✓ | 749 |
| 14.5.2 | Sprawdź, czy dostarczony nagłówek Origin nie jest używany do uwierzytelniania lub decyzji dotyczących kontroli dostępu, ponieważ nagłówek Origin może być łatwo zmieniony przez atakującego. | ✓ | ✓ | ✓ | 364 |
| 14.5.3 | Sprawdź, czy nagłówek udostępniania zasobów między domenami (CORS) Access-Control-Allow-Origin używa ścisłej białej listy zaufanych domen i nie obsługuje “null” Origin. | ✓ | ✓ | ✓ | 346 |
| 14.5.4 | Sprawdź, czy nagłówki HTTP dodane przez zaufany serwer proxy lub urządzenia SSO, takie jak token na okaziciela, są uwierzytelniane przez aplikację. |  | ✓ | ✓ | 306 |

Referencje:

• OWASP Testing Guide 4.0: Testing for HTTP Verb Tampering <https://www.owasp.org/index.php/Testing_for_HTTP_Verb_Tampering_%28OTG-INPVAL-003%29>

• Reflected File Download attacks.<https://www.blackhat.com/docs/eu-14/materials/eu-14-Hafif-Reflected-File-Download-A-New-Web-Attack-Vector.pdf>

• Content Security Policy Cheat Sheet <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets_excluded/Content_Security_Policy_Cheat_Sheet.md>

• Exploiting CORS misconfiguration for BitCoins and Bounties <https://portswigger.net/blog/exploiting-cors-misconfigurations-for-bitcoins-and-bounties>

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

• Sandboxing third party components <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.md>

**Załącznik A: Słowniczek**

* **Dwuskładnikowe uwierzytelnianie (2FA)** - dodaje drugi poziom uwierzytelnienia przy logowaniu na konto.
* **Address Space Layout Randomization (ASLR)** - technika utrudniająca wykorzystywanie błędów związanych z uszkodzeniem pamięci.
* **Bezpieczeństwo aplikacji** - bezpieczeństwo na poziomie aplikacji koncentruje się na analizie komponentów, które składają się na warstwę aplikacji Open Systems Interconnection Reference Model (Model OSI), a nie na przykład na podstawowym systemie operacyjnym lub połączonych sieciach.
* **Weryfikacja bezpieczeństwa aplikacji** - ocena techniczna aplikacji według OWASP ASVS.
* **Raport weryfikacji bezpieczeństwa aplikacji** - raport dokumentujący ogólne wyniki i analizę pomocniczą opracowaną przez weryfikatora dla konkretnej aplikacji.
* **Uwierzytelnianie** - weryfikacja deklarowanej tożsamości użytkownika.
* **Zautomatyzowana weryfikacja** - wykorzystanie zautomatyzowanych narzędzi (narzędzi do dynamicznej analizy, narzędzi do analizy statycznej lub obu) wykorzystujących sygnatury luk w celu znalezienia problemów.
* **Testowanie czarnej skrzynki** - Jest to metoda testowania oprogramowania, która bada funkcjonalność aplikacji bez zaglądania do jej wewnętrznych struktur lub działań.
* **Komponent** - samodzielna jednostka kodu z powiązanymi interfejsami dysku i sieci, które komunikują się z innymi komponentami.
* **Cross-Site Scripting (XSS)** - luka w zabezpieczeniach zwykle występująca w aplikacjach internetowych, umożliwiająca wstrzykiwanie skryptów po stronie klienta do treści.
* **Moduł kryptograficzny** - sprzęt, oprogramowanie i/lub firmware, które implementuje algorytmy kryptograficzne i/lub generuje klucze kryptograficzne.
* **CWE** - Common Weakness Enumeration (CWE) to opracowana przez społeczność lista typowych podatności w zabezpieczeniach oprogramowania.Jest to wspólny język i miara narzędzi bezpieczeństwa oprogramowania oraz jako podstawa do identyfikacji, łagodzenia i zapobiegania słabościom.
* **DAST** - Technologie dynamicznego testowania bezpieczeństwa aplikacji, które mają na celu wykrywanie warunków wskazujących na lukę w zabezpieczeniach aplikacji w czasie jej działania.
* **Weryfikacja designu** - techniczna ocena architektury bezpieczeństwa aplikacji.
* **Weryfikacja dynamiczna** - wykorzystanie zautomatyzowanych narzędzi wykorzystujących sygnatury luk w zabezpieczeniach w celu znalezienia problemów podczas wykonywania aplikacji.
* **Globalnie unikalny identyfikator (GUID)** - unikalny numer referencyjny używany jako identyfikator w oprogramowaniu.
* **Hyper Text Transfer Protocol (HTTPS)** - protokół aplikacji dla systemów rozproszonych oraz opartych na współpracy. Jest to podstawa komunikacji danych w sieci World Wide Web.
* **Klucze zakodowane** na sztywno - klucze kryptograficzne przechowywane w systemie plików, czy to w kodzie, komentarzach czy plikach.
* **Walidacja danych wejściowych** - transformacja i walidacja niezaufanych danych wejściowych użytkownika.
* **Złośliwy kod** - kod wprowadzony do aplikacji podczas jej tworzenia bez wiedzy właściciela aplikacji, kod ten omija zamierzoną politykę bezpieczeństwa aplikacji. Nie jest to to samo, co złośliwe oprogramowanie, takie jak wirus lub “robak internetowy”!
* **Złośliwe oprogramowanie** - kod wykonywalny, który jest wprowadzany do aplikacji w czasie przetwarzania bez wiedzy użytkownika aplikacji lub administratora.
* **Open Web Application Security Project (OWASP**) - ogólnoświatowa wolna i otwarta społeczność skupiona na poprawie bezpieczeństwa aplikacji. Naszą misją jest uczynienie bezpieczeństwa aplikacji „widocznym”, aby ludzie i organizacje mogli podejmować świadome decyzje dotyczące zagrożeń bezpieczeństwa aplikacji. Zobacz: <https://www.owasp.org/>
* **Dane osobowe umożliwiające identyfikację (PII)** - to informacje, które można wykorzystać samodzielnie lub z innymi informacjami w celu identyfikacji, kontaktu lub zlokalizowania pojedynczej osoby lub w celu zidentyfikowania osoby w kontekście.
* **PIE** - plik wykonywalny niezależny od położenia (PIE) to treść kodu maszynowego, która umieszczona gdzieś w pamięci podstawowej, wykonuje się prawidłowo niezależnie od jej adresu bezwzględnego.
* **PKI** - Infrastruktura klucza publicznego (Public Key Infrastructure) to rozmieszczenie kluczy publicznych z odpowiednimi tożsamościami podmiotów. Wiązanie odbywa się podczas procesu rejestracji i wydawania certyfikatów w urzędzie certyfikacji (CA).
* **SAST** - Statyczne testowanie bezpieczeństwa aplikacji (Static application security testing) to zestaw technologii zaprojektowanych do analizowania kodu źródłowego aplikacji, kodu bajtowego i plików binarnych dla warunków kodowania i projektowania, które wskazują na luki w zabezpieczeniach. Rozwiązania SAST analizują aplikację od wewnątrz w czasie kiedy nie jest uruchomiona.
* **SDLC** - Cykl życia oprogramowania.
* **Architektura bezpieczeństwa** - projekt aplikacji, który określa i opisuje, gdzie i jak stosowane są kontrole bezpieczeństwa, a także określa i opisuje lokalizację i wrażliwość danych użytkownika i aplikacji.
* **Konfiguracja zabezpieczeń** - konfiguracja działającej aplikacji, która wpływa na sposób używania zabezpieczeń.
* **Kontrola bezpieczeństwa** - funkcja lub komponent, który przeprowadza kontrolę bezpieczeństwa (np. sprawdzenie kontroli dostępu) lub gdy wywołany powoduje efekt bezpieczeństwa (np. generowanie rejestru kontroli).
* **SQL Injection (SQLi)** - technika iniekcji kodu używana do atakowania aplikacji opartych na danych, w których wprowadzane są złośliwe zapytania SQL.
* **Uwierzytelnianie SSO** - Single Sign On (SSO) występuje, gdy użytkownik loguje się do jednej aplikacji, a następnie automatycznie jest również logowany do innych aplikacji bez konieczności ponownego uwierzytelniania. Na przykład po zalogowaniu się do Google zostaniesz automatycznie zalogowany do innych usług Google, takich jak YouTube, Dokumenty Google i Gmail.
* **Modelowanie zagrożeń** - technika polegająca na rozwijaniu coraz bardziej wyrafinowanych architektur bezpieczeństwa w celu identyfikacji zagrożeń, stref bezpieczeństwa, kontroli bezpieczeństwa oraz ważnych zasobów technicznych i biznesowych.
* **Transport Layer Security** - Protokoły kryptograficzne zapewniające bezpieczeństwo komunikacji przez połączenie sieciowe
* **Fragmenty URI / URL / URL** - Uniform Resource Identifier to ciąg znaków używanych do identyfikacji nazwy lub zasobu internetowego.
* **Weryfikator** - osoba lub zespół dokonujący przeglądu aplikacji pod kątem wymagań OWASP ASVS.
* **Biała lista** - lista dozwolonych danych lub operacji, na przykład lista znaków, które mogą wykonywać walidację danych wejściowych.
* **Certyfikat X.509** - certyfikat cyfrowy, który wykorzystuje powszechnie akceptowany międzynarodowy standard infrastruktury klucza publicznego X.509 do sprawdzania, czy klucz publiczny należy do tożsamości użytkownika, komputera lub usługi zawartej w certyfikacie .

**Załącznik B: Referencje**

Następujące projekty OWASP mogą być pomocne dla użytkowników/ osób stosujących ten standard:

**Projekty OWASP Core**

1. OWASP Top 10 Project: <https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project>

2. OWASP Testing Guide: <https://www.owasp.org/index.php/OWASP_Testing_Project>

3. OWASP Proactive Controls: <https://www.owasp.org/index.php/OWASP_Proactive_Controls>

4. OWASP Security Knowledge Framework:

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

5. OWASP Software Assurance Maturity Model (SAMM):

<https://www.owasp.org/index.php/OWASP_SAMM_Project>

**Projekty Mobile Security**

1. OWASP Mobile Security Project: <https://www.owasp.org/index.php/OWASP_Mobile_Security_Project>

2. OWASP Mobile Top 10 Risks:

<https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks>

3. OWASP Mobile Security Testing Guide:

<https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide>

**Projekty OWASP Internet of Things**

1. OWASP Internet of Things Project: <https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project>

OWASP Serverless projects

1. OWASP Serverless Project: <https://www.owasp.org/index.php/OWASP_Serverless_Top_10_Project>

**Inne**

Poniższe strony mogą również okazać się pomocne:

1. SecLists Github: <https://github.com/danielmiessler/SecLists>

2. MITRE Common Weakness Enumeration: <https://cwe.mitre.org/>

3. PCI Security Standards Council: <https://www.pcisecuritystandards.org>

4. PCI Data Security Standard (DSS) v3.2.1 Requirements and Security Assessment Procedures:

<https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf>

5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures:

<https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf>

6. PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures:

<https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf>

**Załącznik C: Wymagania weryfikacji Internetu Rzeczy**

Ta sekcja była pierwotnie w głównej gałęzi, ale dzięki pracy zespołu IoT OWASP nie ma sensu utrzymywanie dwóch różnych standardów na ten temat. W wersji 4.0 przenosimy to do Załączników i zalecamy każdemu, kto tego potrzebuje, raczej do korzystania z głównego projektu OWASP IoT

Cel kontroli

Urządzenia Embedded/IoT powinny:

* mieć taki sam poziom kontroli bezpieczeństwa wewnątrz urządzenia, jaki mają na serwerze, poprzez wymuszanie kontroli bezpieczeństwa w zaufanym środowisku.
* Wrażliwe dane powinny być przechowywane na urządzeniu w bezpieczny sposób używając sprzętowo wspieranego przechowywania takiego jak bezpieczne elementy.
* Wszystkie wrażliwe dane przesyłane z urządzenia powinny wykorzystywać zabezpieczenia warstwy transportowej.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Opis** | **1** | **2** | **3** | **Od** |
| C1 | Sprawdź, czy interfejsy debugowania warstwy aplikacji, takie jak USB, UART lub inne seryjne warianty są wyłączone lub chronione skomplikowanym hasłem. | ✓ | ✓ | ✓ | 4.0 |
| C2 | Sprawdź, czy klucze kryptograficzne są unikalne dla każdego urządzenia | ✓ | ✓ | ✓ | 4.0 |
| C3 | Jeśli ma to zastosowanie, to sprawdź, czy elementy sterujące ochroną pamięci, takie jak ASLR i DEP, są włączone w systemie operacyjnym embedded / IoT. | ✓ | ✓ | ✓ | 4.0 |
| C4 | Sprawdź, czy interfejsy debugowania na chipie, takie jak JTAG lub SWD, są wyłączone lub czy dostępny mechanizm ochrony jest włączony i odpowiednio skonfigurowany. | ✓ | ✓ | ✓ | 4.0 |
| C5 | Sprawdź, czy zaufane wywołanie jest zaimplementowane i włączone, jeśli jest dostępne na urządzeniu SoC lub CPU. | ✓ | ✓ | ✓ | 4.0 |
| C6 | Sprawdź, czy poufne dane, klucze prywatne i certyfikaty są bezpiecznie przechowywane w Secure Element, TPM, TEE (Trusted Execution Environment) lub zabezpieczone przy użyciu silnej kryptografii. | ✓ | ✓ | ✓ | 4.0 |
| C7 | Sprawdź, czy aplikacje firmware chronią dane podczas przesyłu za pomocą zabezpieczeń transportowych. | ✓ | ✓ | ✓ | 4.0 |
| C8 | Sprawdź, czy aplikacje firmware sprawdzają podpis cyfrowy połączeń z serwerami. |  |  |  | 4.0 |
| C9 | Sprawdź, czy komunikacja bezprzewodowa jest wzajemnie uwierzytelniana. | ✓ | ✓ | ✓ | 4.0 |
| C10 | Sprawdź, czy komunikacja bezprzewodowa jest wysyłana przez zaszyfrowany kanał. | ✓ | ✓ | ✓ | 4.0 |
| C11 | Sprawdź, czy jakiekolwiek użycie zabronionych funkcji C zostało zastąpione odpowiednimi bezpiecznymi równoważnymi funkcjami. | ✓ | ✓ | ✓ | 4.0 |
| C12 | Sprawdź, czy każde firmware ma listę materiałów oprogramowania katalogujących zewnętrzne, wersje i opublikowane luki w zabezpieczeniach. | ✓ | ✓ | ✓ | 4.0 |
| C13 | Sprawdź czy cały kod, w tym pliki binarne innych firm, biblioteki, frameworki są sprawdzane pod kątem zakodowanych na sztywno danych uwierzytelniających (backdoorów). | ✓ | ✓ | ✓ | 4.0 |
| C14 | Sprawdź, czy komponenty aplikacji i firmware nie są podatne na OS Command Injection, wywołując shell command wrappers, skrypty lub że zabezpieczenia uniemożliwiają OS Command Injection. | ✓ | ✓ | ✓ | 4.0 |
| C15 | Sprawdź, czy aplikacje firmware przypinają podpis cyfrowy do zaufanego serwera(ów). |  | ✓ | ✓ | 4.0 |
| C16 | Sprawdzić obecność odporności na ingerencję i / lub funkcji wykrywania. |  | ✓ | ✓ | 4.0 |
| C17 | Sprawdź, czy włączone są wszystkie dostępne technologie ochrony własności intelektualnej dostarczone przez producenta chipów. |  | ✓ | ✓ | 4.0 |
| C18 | Sprawdź, czy istnieją mechanizmy kontroli bezpieczeństwa, które utrudniają odwrotną inżynierię oprogramowania firmware (np. usunięcie pełnych ciągów debugowania). |  | ✓ | ✓ | 4.0 |
| C19 | Sprawdź, czy urządzenie sprawdza poprawność podpisu boot image przed załadowaniem. |  | ✓ | ✓ | 4.0 |
| C20 | Sprawdź, czy proces aktualizacji firmware nie jest podatny na ataki time-of-check vs time-of-use. |  | ✓ | ✓ | 4.0 |
| C21 | Sprawdź, czy urządzenie używa podpisywania kodu i sprawdza poprawność plików aktualizacji oprogramowania przed instalacją. |  | ✓ | ✓ | 4.0 |
| C22 | Sprawdź, czy urządzenia nie można zmienić na starszą wersję prawidłowego firmware. |  | ✓ | ✓ | 4.0 |
| C23 | Sprawdź użycie kryptograficznie bezpiecznego generatora liczb pseudolosowych na urządzeniu wbudowanym (np. Za pomocą generatorów liczb losowych dostarczanych przez chipy). |  | ✓ | ✓ | 4.0 |
| C24 | Sprawdź, czy firmware może wykonywać automatyczne aktualizacje firmware po uprzednio zdefiniowanym harmonogramie. |  | ✓ |  | 4.0 |
| C25 | Sprawdź, czy urządzenie czyści firmware i dane poufne po wykryciu naruszenia lub odebraniu nieprawidłowego komunikatu. |  |  | ✓ | 4.0 |
| C26 | Sprawdź, czy używane są tylko mikrokontrolery obsługujące wyłączanie interfejsów debugowania (np. JTAG, SWD). |  |  | ✓ | 4.0 |
| C27 | Sprawdź, czy używane są tylko mikrokontrolery zapewniające istotną ochronę przed atakami typu "de-capping" i “side channel”. |  |  | ✓ | 4.0 |
| C28 | Sprawdź, czy poufne ślady użytkowania nie są wystawione na zewnętrzne warstwy obwodu drukowanego. |  |  | ✓ | 4.0 |
| C29 | Sprawdź, czy komunikacja między chipami jest szyfrowana. |  |  | ✓ | 4.0 |
| C30 | Sprawdź, czy urządzenie używa podpisywania kodu i waliduje kod przed wykonaniem. |  |  | ✓ | 4.0 |
| C31 | Sprawdź, czy poufne informacje przechowywane w pamięci są nadpisywane zerami, gdy tylko nie są już potrzebne. |  |  | ✓ | 4.0 |
| C32 | Sprawdź, czy aplikacje firmware wykorzystują “pojemniki” kernel do izolacji między aplikacjami. |  |  | ✓ | 4.0 |
| C33 | Sprawdź, czy bezpieczne flagi kompilatora, takie jak -fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap są skonfigurowane do kompilacji firmware. |  |  | ✓ | 4.0 |
| C34 | Sprawdź, czy mikrokontrolery są skonfigurowane z ochroną kodu (jeśli dotyczy). |  |  | ✓ | 4.0 |

Referencje:

• OWASP Internet of Things Top 10 <https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf>

• OWASP Embedded Application Security Project <https://www.owasp.org/index.php/OWASP_Embedded_Application_Security>

• OWASP Internet of Things Project <https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project>

• Trudy TCP Proxy Tool <https://github.com/praetorian-inc/trudy>