Niniejszy dokument zawiera uszczegółowione wymagania bezpieczeństwa dla aplikacji, która jest przedmiotem RFP. Wymagania podzielone są na 11 obszarów.

Prosimy o podanie informacji, czy aplikacja spełnia wymagania zawarte w tabeli. Możliwe jest również wprowadzenie dodatkowego komentarza o sposobie realizacji wymagania.

Spis treści

[I. Architektura 2](#_Toc61949628)

[II. Kontrola dostępu 4](#_Toc61949629)

[III. Identyfikacja i uwierzytelnianie 8](#_Toc61949630)

[IV. Kontrola integralności i ochrona przed złośliwym oprogramowaniem 15](#_Toc61949631)

[V. Kryptografia 19](#_Toc61949632)

[VI. Logowanie zdarzeń i kontrola rozliczalności 27](#_Toc61949633)

[VII. Wytyczne aplikacji mobilnych 30](#_Toc61949634)

[*1.* *Aplikacje Mobilne - Architektura, projektowanie i modelowanie zagrożeń* 30](#_Toc61949635)

[*2.* *Aplikacje Mobilne - Przechowywanie danych i ochrona prywatności* 32](#_Toc61949636)

[*3.* *Aplikacje Mobilne – Kryptografia* 33](#_Toc61949637)

[*4.* *Aplikacje Mobilne - Uwierzytelnianie i zarządzanie sesjami* 34](#_Toc61949638)

[*5.* *Aplikacje Mobilne – Komunikacja sieciowa* 35](#_Toc61949639)

[*6.* *Aplikacje Mobilne – Interakcja między platformami* 36](#_Toc61949640)

[*7.* *Aplikacje Mobilne - Jakość kodu i ustawienia tworzenia* 37](#_Toc61949641)

[*8.* *Aplikacje Mobilne – Odporność na działanie inżynierii odwrotnej* 38](#_Toc61949642)

[VIII. Dokumentacja i szkolenia 39](#_Toc61949643)

[IX. Ciągłość działania 40](#_Toc61949644)

[X. Warunki wsparcia 41](#_Toc61949645)

[XI. Zgodność ze standardem korporacyjnym 42](#_Toc61949646)

[Przypisy 42](#_Toc61949647)

## Architektura

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Architektura systemu musi być co najmniej trójwarstwowa przy czym podstawowy podział na warstwy musi uwzględniać: warstwa danych (np. bazy danych, pliki), warstwa aplikacyjna (serwery aplikacyjne), warstwa prezentacji (serwery obsługujące interfejsy użytkownika, dostęp kliencki do systemu). |  |  |
|  | Architektura systemu powinna pozwalać na fizyczną lub logiczną separacje poszczególnych jego warstw w odniesieniu do maszyn, segmentów sieci (np. VLAN), technologii NAT/PAT oraz umożliwiać separację za pomocą firewalli lub ruterów. |  |  |
|  | Poszczególne komponenty systemu muszą potrafić komunikować się w warunkach opisanych powyżej, przy czym połączenia pomiędzy tymi komponentami muszą być precyzyjnie zdefiniowane z dokładnością do portów i protokołów. Każdy z otwieranych portów czy protokołów musi być uzasadniony. Nie jest dopuszczalne otwieranie zakresu portów w celu uproszczenia definicji połączeń lub z powodu braku znajomości bardziej ograniczonej ich listy. |  |  |
|  | System powinien posiadać fizyczną separację danych dostępnych dla:   * użytkowników wewnętrznych (np. pracownicy ORLEN S.A.) * zewnętrznych (np. klienci).   W przypadku braku takiej możliwości powinna istnieć logiczna separacja oparta na sprawdzonych mechanizmach uwierzytelniania i autoryzacji. |  |  |
|  | Architektura systemu musi umożliwić separację dostępu. Użytkownik zewnętrzny nie może mieć możliwości uzyskania dostępu za pomocą dedykowanych kanałów dla użytkowników wewnętrznych (np. VPN) i na odwrót. |  |  |
|  | System powinien posiadać oddzielne mechanizmy uwierzytelniania użytkowników wewnętrznych i zewnętrznych. W szczególności nie jest dopuszczalne logowanie domenowe bezpośrednio z sieci Internet. |  |  |
|  | Użytkownik może komunikować się wyłącznie przy użyciu warstwy prezentacji. Nie jest dozwolony bezpośredni dostęp użytkowników do pozostałych warstw/komponentów systemu (serwery aplikacyjne, bazy danych, systemy plików). |  |  |
|  | Poszczególne moduły systemu muszą pracować z najmniejszymi możliwymi uprawnieniami. Nie jest dozwolone wymaganie uprawnień administracyjnych do zwykłej pracy tych modułów. |  |  |
|  | Dostęp do danych (o ile to możliwe) powinien być ograniczony na poziomie bazy danych a nie aplikacji. Nie jest dopuszczalna praca aplikacji na bazie danych z uprawnieniami DBA. |  |  |
|  | Niedozwolone jest wykorzystywanie protokołów uważanych powszechnie za niebezpieczne (tftp, NFS, SMB) przy komunikacji pomiędzy sieciami o różnym poziomie zaufanie (szczególne przy przepływnie z sieci o mniejszym zaufaniu do sieci o większym zaufaniu).  W przypadku, gdy koniecznie użycie jest niebezpiecznych protokołów rekomendowane jest zastosowanie protokołu zabezpieczającego taką komunikację (np. TLS, IPSec). |  |  |
|  | Jeśli aplikacja oparta o technologie WWW używa protokołu https wymagane jest, aby ruch szyfrowany mógł być terminowany na brzegowych firewallach aplikacyjnych a następnie w zależności od potrzeby zestawiany drugi tunel z tych urządzeń bezpośrednio do aplikacji (lub przepuszczony ruch nieszyfrowany - http). Jest to wymaganie opcjonalne, zależne od decyzji ORLEN S.A. |  |  |

## Kontrola dostępu

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Mechanizmy kontroli dostępu zaimplementowane w aplikacji muszą egzekwować zasadę ograniczonego dostępu - „wszystko, co nie jest wyraźnie dozwolone, jest zabronione”. Tym samym, muszą skutecznie blokować dostęp użytkowników do wszystkich chronionych usług, funkcji, danych i obiektów (plików, katalogów, danych w bazie danych, URL[[1]](#endnote-1)) z wyłączeniem tych, do których posiadają przyznane uprawnienia. |  |  |
|  | Zalecanym mechanizmem kontroli dostępu jest Mandatory Access Control[[2]](#endnote-2). |  |  |
|  | Użytkownicy mogą mieć dostęp jedynie do tych adresów URL, do których posiadają konkretne uprawnienia. |  |  |
|  | Użytkownicy mogą mieć dostęp jedynie do tych plików, do których posiadają konkretne uprawnienia. |  |  |
|  | Jeżeli w ramach aplikacji funkcjonują użytkownicy o zróżnicowanych uprawnieniach, mechanizmy kontroli dostępu zaimplementowane w tej aplikacji muszą zapewniać możliwość stosowania zasady minimalnych uprawnień, czyli możliwość przydzielania użytkownikom wyłącznie tych uprawnień, które są im niezbędne do wykonywania niezbędnych działań. |  |  |
|  | Aplikacja powinna bazować co najmniej na mechanizmach opartych o role (Role Based Access Control[[3]](#endnote-3)). |  |  |
|  | Dostęp do części administracyjnej aplikacji i części zarządzania treścią merytoryczną aplikacji nie może być udostępniony w sieci Internet. Dostęp do tych części może być możliwy wyłącznie z sieci wewnętrznych. |  |  |
|  | Powinien istnieć scentralizowany mechanizm zabezpieczający dostęp do każdego typu chronionych zasobów. |  |  |
|  | Część administracyjna i część zarządzania treścią powinna być odseparowana od części biznesowej aplikacji poprzez uruchomienie tych części, jako niezależne procesy systemowe, w kontekście uprawnień różnych nieuprzywilejowanych użytkowników, nasłuchujące na odrębnych portach (lub jako odrębne instancje serwerów www i serwerów aplikacyjnych). |  |  |
|  | Jeżeli w ramach pojedynczej aplikacji lub wielu instancji tej samej aplikacji funkcjonują użytkownicy przetwarzający odrębne zestawy danych, niepowiązane ze sobą (np. aplikacja jest wykorzystywana przez niezależne podmioty), zaimplementowane mechanizmy kontroli dostępu muszą skutecznie separować dostęp do zestawów danych poszczególnych użytkowników aplikacji, niezależnie od posiadanych uprawnień. To znaczy, że zaimplementowana separacja musi funkcjonować zarówno na poziomie uprawnień zwykłych użytkowników, jak również na poziomie uprawnień użytkowników uprzywilejowanych zarządzających uprawnieniami lub danymi. |  |  |
|  | Aplikacja musi chronić bezpośrednie odniesienia do obiektów tak, aby tylko uprawnione obiekty lub dane były dostępne dla użytkownika (np. ochrona przed manipulacją bezpośrednimi odniesieniami do obiektów). |  |  |
|  | Wszystkie ustalone reguły kontroli dostępu do usług, funkcji, danych i obiektów muszą być wymuszane po stronie serwera. Ponadto, mechanizmy kontroli dostępu muszą skutecznie blokować dostęp do reguł i atrybutów kontroli dostępu, tak, aby nie była możliwa ich nieautoryzowana modyfikacja przez użytkowników aplikacji, chyba, że są oni do tego uprawnieni. |  |  |
|  | Uprawnienia nie mogą być zaszywane (ang. hardcoding) w kodzie aplikacji |  |  |
|  | Decyzje dotyczące przydzielania uprawnień dostępu nie mogą bazować wyłącznie na parametrach przekazywanych w polach formularzy, ukrytych polach lub innych parametrach przekazywanych do serwera. |  |  |
|  | Dostęp do bazy danych musi być filtrowany zgodnie z uprawnieniami użytkowników aplikacji. |  |  |
|  | W przypadku, gdy używany jest wyłącznie jeden użytkownik bazodanowy dalej dostęp do danych w aplikacji musi być filtrowany zgodnie z uprawnieniami użytkownika aplikacyjnego. Dodatkowo w każdym zapytaniu SQL musi być przesyłany identyfikator użytkownika aplikacyjnego. |  |  |
|  | Mechanizmy kontroli dostępu, które kończą pracę niepowodzeniem muszą to robić w sposób bezpieczny, tzn. nie może nastąpić nieautoryzowane dopuszczenie użytkownika do chronionych zasobów oraz wyjątek, który miał miejsce musi być prawidłowo obsłużony. Poprzez prawidłową obsługę przyjmuje się odnotowanie zdarzenia w logach i wyświetlenie użytkownikowi dostosowanego komunikatu zawierającego identyfikator błędu, bez ujawniania szczegółów technicznych funkcjonowania samej aplikacji lub jej otoczenia infrastrukturalnego. |  |  |
|  | Aplikacja musi logować zdarzenia dotyczące kontroli dostępu. |  |  |
|  | Aplikacja musi logować zmiany konfiguracji |  |  |
|  | Muszą być logowane decyzje o próbie dostępu zakończonej niepowodzeniem. |  |  |
|  | Powinna być możliwość logowania decyzji zakończonych sukcesem. |  |  |
|  | Mechanizmy kontroli dostępu zaimplementowane w aplikacji muszą utrzymywać aktualny stan uprawnień użytkowników i w przypadku zmiany, ich egzekwowanie powinno być realizowane w trybie natychmiastowym. |  |  |
|  | Wszystkie informacje i pliki konfiguracyjne związane z bezpieczeństwem aplikacji muszą być przechowywane w miejscach chronionych przed nieautoryzowanym dostępem. |  |  |
|  | Konfiguracja serwera WWW musi zapobiegać indeksowaniu katalogów. |  |  |
|  | Mechanizmy kontroli dostępu zaimplementowane w aplikacji muszą skutecznie wymuszać ustalone reguły przepływu pracy w aplikacji (przyjęta kolejność realizacji zadań) oraz nałożone limity (wielkość załącznika, ilość operacji, ilość transakcji, ilość zmian, itp.). |  |  |
|  | Aplikacja powinna wykorzystywać sprawdzone mechanizmy kontroli dostępu, które oferowane są przez system operacyjny, powiązane usługi i zastosowany framework[[4]](#endnote-4). |  |  |
|  | Aplikacja uruchamiana na serwerze WWW powinna być uruchamiana z minimalnymi uprawnieniami (np. Microsoft IIS[[5]](#endnote-5) - Low Privilege Application Pool Identity) |  |  |
|  | Jeśli to technicznie możliwe (i uzasadnione biznesowo) kontrola dostępu do aplikacji powinna być zintegrowana z systemem operacyjnym i korzystać z natywnych funkcji właściwego systemu (np. Single Sign-On [[6]](#endnote-6)- zintegrowane uwierzytelnianie w przypadku Windows). |  |  |
|  | Jeśli to możliwe, aplikacja powinna korzystać z mechanizmów typu sandbox[[7]](#endnote-7) lub izolacji procesów. |  |  |
|  | Aplikacja lub framework musi generować silne, losowe tokeny anty-CSRF unikalne dla użytkownika, jako część wszystkich istotnych operacji lub przy dostępie do wrażliwych danych. Aplikacja musi weryfikować obecność tego tokenu przy przetwarzaniu wskazanych zapytań. |  |  |
|  | Zdalne obiekty IFRAME oraz wymiana zasobów pomiędzy domenami nie mogą pozwalać na zawieranie dowolnych treści zewnętrznych.  Flash, Silverlight lub polityka wymiany zasobów pomiędzy domenami innej aplikacji internetowej musi być skonfigurowana tak by zapobiegać nieuwierzytelnionemu dostępowi zdalnemu. |  |  |

## Identyfikacja i uwierzytelnianie

| Lp. | Opis | SPEŁNIA / NIE SPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Wszyscy użytkownicy aplikacji lub procesy wykonywane w imieniu tych użytkowników muszą być jednoznacznie zidentyfikowani przed uzyskaniem dostępu. |  |  |
|  | Wszyscy użytkownicy aplikacji lub procesy wykonywane w imieniu tych użytkowników muszą przechodzić proces uwierzytelnienia. |  |  |
|  | Wymuszanie uwierzytelniania użytkowników musi być realizowane po stronie serwera. |  |  |
|  | Wszystkie mechanizmy uwierzytelniania (również biblioteki wywołujące zewnętrzne usługi uwierzytelniania) muszą być zaimplementowane centralnie. |  |  |
|  | Wykonanie wrażliwych operacji w aplikacji powinno być poprzedzone ponownym uwierzytelnieniem. |  |  |
|  | W przypadku uwierzytelniania użytkowników na bazie certyfikatów PKI[[8]](#endnote-8), mechanizm uwierzytelniania w aplikacji musi zapewniać: budowę i weryfikację pełnej ścieżki zaufania dla certyfikatu użytkownika uwzględniając wytyczne standardu X.509, weryfikację ważności certyfikatu, weryfikację braku unieważnienia certyfikatu z aktualną w danej chwili listą certyfikatów unieważnionych, weryfikację zgodności wystawcy z zaufanymi i autoryzowanymi wystawcami certyfikatów, istnienia powiązania certyfikatu z kontem w aplikacji oraz weryfikację podpisu cyfrowego użytkownika. |  |  |
|  | W przypadku uwierzytelniania użytkowników na bazie certyfikatów PKI, klucz prywatny z certyfikowanej pary kluczy musi być zabezpieczony hasłem dostępowym ustalanym przez użytkownika. Rekomendowaną polityką haseł jest polityka haseł dostępowych. |  |  |
|  | Aplikacja powinna wymagać, aby wszystkie funkcje operujące na czynnikach uwierzytelniania danej tożsamości (np. rejestracja, aktualizacja profilu, przypomnienie loginu/hasła), które mogą przywrócić dostęp do konta są co najmniej tak odporne na ataki jak główny mechanizm uwierzytelniający. |  |  |
|  | Wszystkie strony oraz zasoby muszą wymagać uwierzytelnienia za wyjątkiem tych specjalnie przeznaczonych na dostęp publiczny. |  |  |
|  | Muszą istnieć mechanizmy pozwalające na zarządzanie informacjami uwierzytelniającymi, tj. nadawaniem, zmianą, blokowaniem, czasem życia i przechowywaniem konta. |  |  |
|  | Aplikacja nie może pozwalać na odzyskanie utraconego hasła. |  |  |
|  | Aplikacja musi posiadać mechanizm pozwalający na zmianę utraconego hasła w sposób bezpieczny z wykorzystaniem dodatkowego kanału komunikacji (np. e-mail lub SMS). Mechanizm zmiany hasła powinien być odporny na ataki przynajmniej w takim stopniu jak podstawowy mechanizm uwierzytelniania. W przypadku wiadomości e-mail mechanizm zmiany hasła powinien wykorzystywać unikatowy adres URL wygenerowany dla danego użytkownika, przy czym część unikatowa powinna być nie krótsza niż 128 bitów. W przypadku wiadomości SMS, użytkownik powinien otrzymać kod SMS, nie krótszy niż 8 znaków alfanumerycznych. Czas ważności danych pozwalających na zmianę hasła powinien być nie dłuższy niż 15 minut. |  |  |
|  | Hasła do kont muszą być tworzone z użyciem ciągu zaburzającego (ang. salt) dla danego konta (np. wewnętrzny identyfikator użytkownika, czas utworzenia konta) i przekształcane przy użyciu mechanizmu hashowania[[9]](#endnote-9) przed ich zapisaniem. |  |  |
|  | Wszystkie dane uwierzytelniające służące do uzyskiwania dostępu do usług zewnętrznych względem aplikacji muszą być zaszyfrowane i przechowywane w zabezpieczonej lokalizacji. |  |  |
|  | W przypadku innego sposobu uwierzytelniania niż Active Directory muszą istnieć mechanizmy pozwalające na konfigurację złożoności haseł dla użytkowników, tj. budowy hasła (małe, wielkie litery, znaki specjalne, cyfry), długości hasła oraz czasu życia hasła. |  |  |
|  | Domyślna polityka haseł powinna wymuszać tworzenie haseł o długości minimum 8 znaków, zawierających co najmniej jedną małą i dużą literę, cyfrę oraz znak specjalny. |  |  |
|  | System powinien pamiętać co najmniej 6 ostatnio stosowanych haseł. |  |  |
|  | Domyślna polityka haseł powinna wymuszać minimalny oraz maksymalny czas życia haseł. Domyślny minimalny czas życia haseł powinien wynosić 1 dzień, a maksymalny 30-180 dni. |  |  |
|  | Aplikacja powinna pozwalać na ustawianie długich i skomplikowanych haseł oraz powinna wymagać minimalnej złożoności haseł w celu zapobiegania stosowaniu powszechnie używanych haseł. |  |  |
|  | Jakiekolwiek domyślne hasła używane inicjalnie w systemie muszą być przed wdrożeniem produkcyjnym zmienione na zgodne z polityka haseł (opisaną w tym dokumencie). |  |  |
|  | Muszą istnieć mechanizmy zapewniające kontrolę sesji uwierzytelnionego użytkownika (np. zamykanie nieaktywnych sesji po określonym czasie). |  |  |
|  | Identyfikator sesji nie może być krótszy niż 128 bitów. |  |  |
|  | Identyfikator sesji musi być losowy. |  |  |
|  | Identyfikator sesji musi być generowany z jak najszerszego zestawu znaków. |  |  |
|  | Identyfikator sesji musi być unikatowy dla użytkowników danej aplikacji. |  |  |
|  | Identyfikator sesji musi być zmieniany przy logowaniu użytkownika. |  |  |
|  | Identyfikator sesji musi być zmieniany przy ponownym uwierzytelnieniu. |  |  |
|  | Identyfikator sesji musi być zmieniany przy przejściu pomiędzy HTTP[[10]](#endnote-10) i HTTPS[[11]](#endnote-11). |  |  |
|  | Może być akceptowany za poprawny tylko ten identyfikator, który został wygenerowany przez aplikację. |  |  |
|  | Identyfikator sesji musi być unieważniany po określonym czasie bezczynności użytkownika. |  |  |
|  | Identyfikatory sesyjne nigdy nie mogą być przekazywane inaczej niż poprzez nagłówek cookie, w szczególności w adresie URL, wiadomościach błędu lub logach. Wlicza się w to wyłączenie wsparcia tzw. URL rewriting dla ciasteczek sesyjnych. |  |  |
|  | Identyfikator sesji nie może być ujawniany w komunikatach błędów i logach. |  |  |
|  | Identyfikator sesji musi być unieważniany i zmieniany lub usuwany przy wylogowaniu użytkownika. |  |  |
|  | Identyfikator sesji nie może być zapamiętywany w przeglądarce (brak funkcji zapamiętaj mnie). |  |  |
|  | Maksymalny okres bezczynności użytkownika powinien wynosić 15 minut. Upłynięcie maksymalnego czasu bezczynności musi skutkować unieważnieniem sesji po stronie serwera oraz klienta (wylogowaniem). |  |  |
|  | Powinno być możliwe zdefiniowanie bezwzględnego maksymalnego czasu życia sesji. |  |  |
|  | Wszystkie strony, do których dostęp wymaga uwierzytelnienia muszą zawierać linki umożliwiające wylogowanie. |  |  |
|  | Cookie[[12]](#endnote-12) zawierające uwierzytelnione tokeny[[13]](#endnote-13) / identyfikatory sesji muszą mieć ustawione atrybuty domain i path odpowiednio dla lokalizacji. |  |  |
|  | Mechanizmy uwierzytelniania użytkowników, które kończą pracę niepowodzeniem muszą to robić w sposób bezpieczny, tzn. nie może nastąpić dopuszczenie użytkownika do chronionych zasobów oraz wyjątek, który miał miejsce musi być prawidłowo obsłużony. Poprzez prawidłową obsługę przyjmuje się odnotowane zdarzenia w logach i wyświetlenie użytkownikowi dostosowanego komunikatu zawierającego identyfikator błędu, bez ujawniania szczegółów technicznych funkcjonowania samej aplikacji lub jej otoczenia infrastrukturalnego. |  |  |
|  | Aplikacja musi mieć możliwość uwierzytelniania użytkowników w oparciu o powszechnie używane i popularne metody uwierzytelniania. |  |  |
|  | Aplikacja powinna umożliwiać uwierzytelnianie Single Sign-On w oparciu o mechanizm zintegrowanego uwierzytelniania Windows. |  |  |
|  | Aplikacja powinna umożliwiać uwierzytelnianie Single Sign-On z wykorzystaniem kont domenowych systemu Windows. |  |  |
|  | Aplikacja powinna wykorzystywać domyślną implementację mechanizmu zarządzania sesją, dostarczaną przez stosowany framework. |  |  |
|  | W przypadku kont uprzywilejowanych (np. administratorów) powinno być zastosowane dwuskładnikowe uwierzytelnianie. |  |  |
|  | Dane uwierzytelniające nie mogą być zaszywane (ang. hardcoding) w kodzie aplikacji lub przekazywane w parametrach adresu URL. |  |  |
|  | W aplikacjach dostępnych w sieciach publicznych (np. sieci Internet), uwierzytelnianie w celu dostępu do zasobów związanych z czynnościami administracyjnymi powinno być możliwe jedynie z sieci wewnętrznej. |  |  |
|  | Aplikacja musi posiadać mechanizm ochrony przez atakami siłowymi (ang. brute-force)[[14]](#endnote-14) na dane uwierzytelniające, blokujący kolejne próby uwierzytelnienia na zdefiniowany okres. |  |  |
|  | Aplikacja powinna stosować domyślną politykę blokowania ataków siłowych. Blokowanie możliwości uwierzytelnienia powinno następować po 5 nieudanych próbach. Okres blokowania powinien trwać minimum 15 minut, a licznik blokowania konta powinien być zerowany po 5 minutach. Aplikacja powinna posiadać mechanizm pozwalający na bezwzględne blokowanie konta, po przekroczeniu ustalonej liczby nieudanych prób uwierzytelnienia. |  |  |
|  | Wszystkie pola służące do wprowadzania haseł nie mogą pokazywać haseł użytkowników w czasie ich wpisywania oraz muszą mieć wyłączoną funkcję automatycznego uzupełnienia. |  |  |
|  | Mechanizm autouzupełniania haseł oraz wrażliwych danych (np. PIN) musi być wyłączony. |  |  |
|  | Wszystkie decyzje dotyczące wyników uwierzytelniania muszą być logowane, zarówno te zakończone sukcesem jak i te zakończone porażką. |  |  |
|  | Odpowiedzi aplikacji powinny zawierać nagłówki HTTP zwiększające poziom bezpieczeństwa użytkowników aplikacji |  |  |
|  | Odpowiedzi aplikacji muszą zawierać następujące nagłówki HTTP definiujące podstawową politykę bezpieczeństwa adresów, z którymi aplikacja może nawiązywać połączenie: Content-Security-Policy (Google Chrome), X-Content-Security-Policy (Firefox), X-WebKit-CSP (przeglądarki oparte na WebKit, np. Safari) |  |  |
|  | Nagłówek Content Security Policy musi mieć zdefiniowany atrybut default-src oraz (jeżeli to konieczne) atrybuty script-src, img-src, frame-src, connect-src tak, aby zezwolone były połączenia jedynie z zaufanymi adresami, np. „Content-Security-Policy: default-src 'self'”. |  |  |
|  | Aplikacja musi wymuszać używanie filtrów Cross-Site Scripting zaimplementowanych w przeglądarkach poprzez ustawienie nagłówka odpowiedzi HTTP: „X-XSS-Protection: 1; mode=block;” |  |  |
|  | Odpowiedzi HTTPS aplikacji muszą zawierać nagłówek „Strict-Transport-Security: max-age=31536000; includeSubDomains” |  |  |
|  | Aplikacja powinna wykrywać oraz blokować próby zautomatyzowanego logowania się na konta użytkowników z jednym hasłem (tzw. bruteforce horyzontalny) |  |  |
|  | Aplikacja musi zapobiegać enumeracji użytkowników przez logowanie, resetowanie hasła lub funkcjonalność przypomnienia loginu. |  |  |
|  | Aplikacja nie może używać domyślnych haseł dla frameworka aplikacji lub jakichkolwiek komponentów użytych przez aplikację. |  |  |
|  | Wszystkie dane uwierzytelniające służące do uzyskania dostępu do zewnętrznych usług muszą być zaszyfrowane oraz przechowywane w bezpiecznym miejscu (nie w kodzie źródłowym). |  |  |
|  | Funkcja przypomnienia hasła nie może blokować lub w inny sposób wyłączać konta, dopóki użytkownik nie zmieni hasła z powodzeniem. Należy jednak zabezpieczać się przed uporczywymi próbami przypomnienia hasła. |  |  |
|  | Aplikacja nie może używać powszechnie znanych pytań/odpowiedzi (tzw. "tajne" pytania i odpowiedzi). |  |  |
|  | Aplikacja nie może zezwalać na równoczesne działanie wielu sesji tego samego użytkownika pochodzących z różnych maszyn. |  |  |
|  | Każdy użytkownik powinien mieć unikatowy identyfikator, który jest niezmienny przez cały czas istnienia tego użytkownika  w systemie. |  |  |
|  | Identyfikator użytkownika wyrejestrowanego z systemu nie może być przypisany innemu użytkownikowi. |  |  |
|  | Po zalogowaniu do systemu użytkownik powinien uzyskać informacje o błędnych próbach logowania do jego konta (a przynajmniej o ostatniej błędnej próbie) oraz ostatniej pozytywnej próbie logowania do systemu. Wymaganie opcjonalne zależne od decyzji ORLEN S.A.. |  |  |
|  | W przypadku systemów, gdzie nie ma zastosowania mechanizm SSO musi być możliwość automatycznego lub ręcznego blokowania kont użytkowników np. ze względu na wygaśnięcie ważności konta. |  |  |

## Kontrola integralności i ochrona przed złośliwym oprogramowaniem

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Muszą istnieć mechanizmy lub proces potwierdzający autentyczność instalowanej aplikacji (sprawdzanie sum kontrolnych). |  |  |
|  | Muszą istnieć mechanizmy potwierdzające spójność krytycznych plików w systemie. |  |  |
|  | Muszą istnieć mechanizmy potwierdzające integralność kodu, bibliotek i plików wykonywalnych (w tym kodu i plików wytworzonych przez zewnętrznych dostawców). |  |  |
|  | Muszą istnieć mechanizmy potwierdzające integralność plików konfiguracyjnych. |  |  |
|  | Muszą istnieć mechanizmy zapewniające kontrolę integralności sesji użytkownika. |  |  |
|  | Aplikacja nie może zawierać złośliwego kodu (w tym kodu, który mógłby zostać użyty do nawiązywania sesji z pominięciem mechanizmów uwierzytelniania – np. backdoor). |  |  |
|  | Muszą istnieć mechanizmy zapewniające kontrolę wprowadzanych danych (w przypadku wprowadzania ciągów znaków - ich format i składnię). |  |  |
|  | Wszystkie źródła wejścia w aplikacji muszą mieć zdefiniowane i zastosowane wzorce pozytywnej walidacji. |  |  |
|  | Walidacja wejścia w aplikacji zakończona niepowodzeniem musi odrzucać lub oczyszczać przyjmowane dane. |  |  |
|  | Wszystkie źródła wejścia muszą posiadać zdefiniowaną stronę kodową np. UTF-8. |  |  |
|  | Cała walidacja danych wejściowych lub mechanizmy enkodowania muszą być realizowane po stronie serwera. |  |  |
|  | Aplikacja musi wykorzystywać jeden mechanizm walidacji danych wejściowych dla każdego typu przyjmowanych danych. |  |  |
|  | Wszystkie walidacje wejścia zakończone niepowodzeniem muszą być logowane. |  |  |
|  | Przez rozpoczęciem walidacji wszystkie dekodery lub interpretery muszą otrzymywać dane wejściowe w postaci ustandaryzowanej. |  |  |
|  | Aplikacja musi filtrować i nadzorować manipulowanie parametrami HTTP, zwłaszcza, jeśli framework nie odróżnia źródła parametrów (metod GET, POST, ciasteczek, nagłówków). |  |  |
|  | Muszą istnieć mechanizmy kontroli rodzaju, zawartości, zakresu, nazwy, formatu i wielkości plików w przypadku funkcjonalności związanych z importem plików (upload) oraz pobieraniem plików (download). |  |  |
|  | Zawartość importowanych plików musi być sprawdzana pod kątem zawierania złośliwego kodu. |  |  |
|  | Muszą istnieć mechanizmy uniemożliwiające wykonanie kodu zawartego w importowanych plikach. |  |  |
|  | Nazwy plików oraz ścieżek otrzymane z niezaufanych źródeł muszą być znormalizowane w celu wyeliminowania ataków manipulacją ścieżką (path traversal). |  |  |
|  | Parametry uzyskane z niezaufanych źródeł nie mogą być użyte przy tworzeniu nazw plików, ścieżek lub jakichkolwiek obiektów systemu plików bez poprzedniego znormalizowania, walidowania danych wejściowych oraz enkodowania danych wyjściowych. |  |  |
|  | Pliki otrzymane z niezaufanych źródeł muszą być przetrzymywane poza katalogiem głównym (webroot). |  |  |
|  | Serwer aplikacyjny musi być skonfigurowany tak by domyślnie blokować dostęp do zdalnych zasobów lub systemów poza serwerem aplikacyjnym. |  |  |
|  | Aplikacja musi mieć zaimplementowane mechanizmy enkodowania/escapowania[[15]](#endnote-15) danych wyjściowych. |  |  |
|  | Wszystkie niezaufane dane, których wynikiem jest kod HTML[[16]](#endnote-16) (elementy HTML, atrybuty HTML, wartości danych javascript[[17]](#endnote-17), bloki CSS[[18]](#endnote-18) i atrybuty URI[[19]](#endnote-19)) muszą podlegać escapowaniu odpowiednio do kontekstu. |  |  |
|  | Wszystkie mechanizmy enkodowania / escapowania muszą być zaimplementowane po stronie serwera. |  |  |
|  | Mechanizmy enkodowania / escapowania danych wyjściowych muszą enkodować wszystkie znaki, które nie są uważane za bezpieczne dla danego interpretera. |  |  |
|  | Dla każdego typu enkodowania / escapowania wyjścia wykonywanego przez aplikację musi istnieć pojedynczy mechanizm bezpieczeństwa dla tego typu docelowych danych wyjściowych. |  |  |
|  | Wszystkie niezaufane dane wyjściowe trafiające do interpreterów SQL[[20]](#endnote-20) muszą używać parametryzowanych interfejsów (ang. prepared statements), przygotowanych instrukcji lub być właściwie escapowane. |  |  |
|  | Wszystkie niezaufane dane wyjściowe, których wynikiem jest XML [[21]](#endnote-21)muszą używać parametryzowanych interfejsów lub być właściwie escapowane. |  |  |
|  | Wszystkie niezaufane dane używane w zapytaniach LDAP[[22]](#endnote-22) muszą być właściwie escapowane. |  |  |
|  | Wszystkie niezaufane dane używane jako parametry poleceń systemu operacyjnego muszą być właściwie escapowane. |  |  |
|  | Wszystkie niezaufane dane przekazywane do interpreterów innych niż ww., również muszą być właściwie escapowane. |  |  |
|  | Wszystkie połączenia wykorzystywane do komunikacji z innymi elementami aplikacji lub systemami podmiotów zewnętrznych, przez które przesyłane są dane wrażliwe, muszą być zabezpieczone kryptograficznie oraz strony takich połączeń muszą być uwierzytelnione. |  |  |
|  | Aplikacja oraz otoczenie infrastrukturalne, w którym funkcjonuje aplikacja nie może pozwalać na realizację połączeń niezabezpieczonych w przypadku, gdy nie powiodła się próba nawiązania połączenia bezpiecznego. |  |  |
|  | Czas w aplikacji musi być synchronizowany z zaufanym źródłem czasu. |  |  |
|  | Naruszenia integralności muszą być logowane oraz objęte właściwą obsługą. |  |  |
|  | Aplikacja musi mieć możliwość współpracy z zastosowanymi wewnętrznie ustawieniami zabezpieczeń klienta WWW (używanej w Organizacji przeglądarki WWW). |  |  |
|  | Aplikacja nie może wysyłać żadnych informacji do serwerów w sieci Internet (w szczególności informacji o charakterze poufnym). |  |  |
|  | W aplikacji musi być możliwość pobierania uaktualnień z wewnętrznego serwera (zalecane, aby odbywało się to w sposób automatyczny). |  |  |
|  | Środowisko uruchomieniowe aplikacji musi wykorzystywać mechanizmy zapobiegające wystąpieniu błędów przepełnienia bufora pamięci. |  |  |
|  | W przypadku, gdy w aplikacji wykorzystywane są zewnętrzne biblioteki czy framework-i należy wykorzystywać wyłącznie aktualne wersje. |  |  |
|  | Aplikacja nie może polegać na skryptach znajdujących się na serwerach firm trzecich. Gdy jednak jest to konieczne, wykonawca musi o tym fakcie powiadomić ORLEN S.A. i uzyskać pisemną zgodę. |  |  |
|  | Dostawca powinien dostarczyć aplikację razem z kodem źródłowym. |  |  |
|  | Kod źródłowy aplikacji musi zostać poddany analizie bezpieczeństwa. |  |  |

## Kryptografia

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Transmisja informacji wrażliwych przekazywanych za pośrednictwem publicznych sieci musi odbywać się wyłącznie w postaci zaszyfrowanej i tylko z wykorzystaniem protokołów komunikacji zapewniających wysoki poziom poufności. |  |  |
|  | Wszystkie połączenia do zewnętrznych systemów, które dotyczą wrażliwych informacji lub funkcji muszą być uwierzytelnione. |  |  |
|  | Wszystkie połączenia do zewnętrznych systemów, które dotyczą wrażliwych informacji lub funkcji muszą używać konta o minimalnych przywilejach koniecznych do właściwej pracy aplikacji. |  |  |
|  | Składowanie i przetwarzanie wrażliwych informacji w systemach, aplikacjach i bazach danych powinno być możliwe jedynie w postaci zaszyfrowanej (np. z użyciem natywnych mechanizmów szyfrujących). |  |  |
|  | Wszystkie wrażliwe dane, w cache lub kopiach tymczasowych, przechowywane na serwerze muszą być chronione przed nieautoryzowanym dostępem lub czyszczone / unieważniane po uzyskaniu dostępu do danych wrażliwych przez upoważnionego użytkownika. |  |  |
|  | Każdy typ wrażliwych danych musi być usuwany z aplikacji wraz z końcem wymaganego terminu ich przetrzymywania. |  |  |
|  | Dane wrażliwe nie mogą być nigdzie przechowywane w postaci jawnej. System powinien zapewniać kryptograficzne środki ochrony przechowywanych danych wrażliwych. |  |  |
|  | Składowanie informacji uwierzytelniającej (hasła, tokeny oraz klucze prywatne) musi odbywać się jedynie w zabezpieczonej przestrzeni (np. zewnętrznym serwerze uwierzytelniającym, urządzeniu HSM[[23]](#endnote-23)). |  |  |
|  | Dostęp do wszystkich kluczy master secret [[24]](#endnote-24)musi być chroniony przed nieuprawnionym dostępem. |  |  |
|  | Stosowane informacje uwierzytelniające nie powinny być dostępne w kodzie źródłowym aplikacji (ang. hardcoding). |  |  |
|  | Przechowywanie informacji uwierzytelniającej w pamięci aplikacji musi odbywać się w sposób uniemożliwiający proste jej odczytanie (np. przechowywanie hasła w postaci jawnego tekstu jest niedopuszczalne). |  |  |
|  | Należy stosować mechanizmy zapobiegające przechowywaniu wrażliwych informacji po stronie użytkownika. |  |  |
|  | Wszystkie formularze zawierające dane wrażliwe nie powinny być cache’owane po stronie klienta, włączając w to funkcjonalność automatycznego wypełniania pól. |  |  |
|  | Wszystkie wrażliwe dane, w cache lub kopiach tymczasowych, przesyłane do klienta muszą być chronione przed nieautoryzowanym dostępem lub czyszczone / unieważniane po uzyskaniu dostępu do danych wrażliwych przez upoważnionego użytkownika (np. powinny być ustawiane odpowiednie nagłówki kontrolne Cache-Control: no-cache i no-store). |  |  |
|  | Aplikacja WWW udostępniona w sieci Internet (szczególnie służąca do komercyjnego udostępniania usług) musi stosować certyfikaty SSL wystawione przez zewnętrzne zaufane centrum certyfikacji. Dla szczególnie istotnych usług powinien to być certyfikat typu Extended Validation Certificate (EV SSL, ang. certyfikaty rozszerzonej walidacji). |  |  |
|  | Podejrzanie zwiększona ilość zapytań informacyjnych lub zapytań wywołujących krytyczne transakcje muszą być blokowane. |  |  |
|  | Obsługa protokołu HTTP/HTTPS musi następować w zgodnie z najlepszymi praktykami bezpieczeństwa (np. ustawianie wymaganych nagłówków HTTP oraz ich atrybutów, konfiguracja protokołu TLS[[25]](#endnote-25)). |  |  |
|  | Musi być możliwe zbudowanie ścieżki z zaufanego CA[[26]](#endnote-26) do każdego serwowanego certyfikatu TLS i każdy certyfikat serwera musi być ważny. |  |  |
|  | Nieudane połączenia TLS nie mogą być powtórnie nawiązywane jako połączenia niezabezpieczone. |  |  |
|  | TLS musi być używany dla wszystkich połączeń (zewnętrznych i backendowych), które są uwierzytelnione lub są związane z wrażliwymi danymi lub funkcjami. |  |  |
|  | Nieudane backendowe połączenia TLS muszą być logowane. |  |  |
|  | Ścieżki certyfikatów muszą być budowane i weryfikowane dla wszystkich certyfikatów klienckich przy użyciu skonfigurowanych powiązań zaufania i informacji nt. unieważniania certyfikatów. |  |  |
|  | Wykorzystywana konfiguracja protokołu SSL/TLS powinna być wolna od znanych błędów bezpieczeństwa np. podatności na ataki DoS[[27]](#endnote-27) wykorzystujący renegocjację połączeń inicjowaną przez klienta, atak BEAST[[28]](#endnote-28), atak CRIME[[29]](#endnote-29), atak MITM[[30]](#endnote-30)plaintext injection[[31]](#endnote-31) wykorzystujący niebezpieczną renegocjację połączeń itp. |  |  |
|  | Wykorzystywana konfiguracja protokołu SSL/TLS powinna stosować wyłącznie zestawy szyfrów powszechnie uznawane za silne. |  |  |
|  | Dla wszystkich połączeń powinno zostać zdefiniowane kodowanie znaków. |  |  |
|  | Przekierowania HTTP nie mogą zawierać danych pozbawionych walidacji. |  |  |
|  | Aplikacja musi akceptować wyłącznie zdefiniowany zestaw metod zapytań HTTP, takich jak GET i POST. |  |  |
|  | Każda odpowiedź HTTP musi zawierać nagłówek Content-type, określający zestaw bezpiecznych znaków (np. UTF-8). |  |  |
|  | Flaga HttpOnly musi być wykorzystywana dla wszystkich cookie, do których nie jest wymagany dostęp z JavaScript. |  |  |
|  | Flaga secure musi być wykorzystywana dla wszystkich cookie, które zawierają wrażliwe informacje, włączając w to cookie sesyjne. |  |  |
|  | Wszystkie strony zwracane od serwera powinny zawierać nagłówek X-Frame-Options, ustalony na wartość DENY lub SAMEORIGIN. Nagłówek ten aktywuje politykę po stronie przeglądarki, ograniczającą możliwość używania mechanizmu ramek IFRAME i ładowania danej strony do ramki (DENY) lub zezwalającą wyłącznie na ładowanie danej strony do ramki pochodzącej z tego samego serwera i numeru portu, co sama strona (SAMEORIGIN). Dodatkowo, aplikacja powinna posiadać mechanizmy wykrywające i blokujące ładowanie treści strony w ramkę pływającą IFRAME poprzez zastosowanie mechanizmu framebusting. |  |  |
|  | Nagłówki HTTP i/lub inne mechanizmy dla starszych przeglądarek muszą być załączone by zapobiegać atakom Clickjacking. |  |  |
|  | Nagłówki HTTP dodane przez frontend (np. X-Real-IP) wykorzystane w aplikacji muszą być chronione przed zmianą przez użytkownika końcowego. |  |  |
|  | Nagłówki HTTP nie mogą ujawniać szczegółowych informacji o wersji komponentów. |  |  |
|  | Nagłówki HTTP w zapytaniach i odpowiedziach muszą zawierać wyłącznie drukowalne znaki ASCII. |  |  |
|  | Aplikacja musi generować silne losowe tokeny będące częścią linków i formularzy powiązanych z transakcjami lub mających dostęp do wrażliwych danych, a także w trakcie przetwarzania takich zapytań aplikacja musi sprawdzać obecność tego tokena oraz poprawność jego wartości odpowiednio dla bieżącego użytkownika. |  |  |
|  | Wszystkie dane wrażliwe muszą być przesyłane do serwera wewnątrz wiadomości HTTP (tzn. parametry URL nie są używane do przesyłania danych wrażliwych). |  |  |
|  | Stosowane moduły kryptograficzne muszą pracować w trybie zapewniającym należyty stopień bezpieczeństwa (stosowanie odpowiedniego źródła losowości, należyta obsługa błędów, zgodność z normami, standardami i wewnętrznymi politykami, a także najlepszymi praktykami w dziedzinie kryptografii)~~.~~ |  |  |
|  | Wszystkie liczby losowe, losowe nazwy plików, losowe GUID’y[[32]](#endnote-32) oraz losowe ciągi znaków, których napastnik nie powinien odgadnąć muszą być generowane z wykorzystaniem zatwierdzonego generatora liczb losowych z modułu kryptograficznego. |  |  |
|  | Wszystkie moduły kryptograficzne, które kończą pracę niepowodzeniem muszą robić to w sposób bezpieczny. |  |  |
|  | Próby naruszenia ochrony kryptograficznej oraz jej błędne działanie muszą być logowane. |  |  |
|  | Moduły kryptograficzne wykorzystywane przez aplikację powinny być zatwierdzone w zakresie zgodności ze standardem FIPS 140-2[[33]](#endnote-33) lub jego odpowiednikiem, poziom (Level) certyfikacji może zostać ustalony w sposób ekspercki po analizie wrażliwości przetwarzanych danych. |  |  |
|  | Moduły kryptograficzne powinny pracować w trybie zgodnym z opublikowanymi politykami bezpieczeństwa dla danego modułu. |  |  |
|  | Wszystkie funkcje kryptograficzne służące do zabezpieczania danych poufnych przed użytkownikiem aplikacji muszą być zaimplementowane po stronie serwera. |  |  |
|  | Implementacja mechanizmów kryptograficznych w aplikacji musi wykorzystywać standardowe, powszechnie używane i akceptowane biblioteki kryptograficzne. Nie jest akceptowane implementowanie algorytmów, bibliotek i protokołów kryptograficznych we własnym zakresie. |  |  |
|  | Implementacja kryptograficznych mechanizmów bezpieczeństwa nie może wykorzystywać powszechnie uznanych za podatne lub o słabej mocy kryptograficznej protokołów (takich, jak np. SSL v3, TLS 1.1). Użycie takich protokołów powinno być wyłączone po stronie serwera. |  |  |
|  | Aplikacja powinna mieć możliwość implementacji polityk określających zarządzanie kluczami kryptograficznymi (np. generowanie, rozprowadzanie, unieważnianie, w jaki sposób wygasają). |  |  |
|  | Aplikacja musi pozwalać na implementacje funkcji (dla zidentyfikowanych wrażliwych danych) przetwarzania, przechowywana i przesyłana tych danych zgodnie z określoną polityką, które może regulować kwestie kontroli dostępu do tych danych oraz kiedy te dane muszą podlegać szyfrowaniu. |  |  |
|  | Musi istnieć metoda usunięcia z aplikacji każdego typu wrażliwych danych na końcu ich wymaganego okresu ważności. |  |  |
|  | Wrażliwe dane muszą być usuwane z pamięci tak szybko jak tylko nie będą potrzebne i używane przez funkcje oraz techniki wspierane przez framework/bibliotekę/system operacyjny. |  |  |
|  | Jeżeli dane transmitowane są w tunelu SSL VPN, to musi być on zgodny ze specyfikacją transmisji danych protokołem HTTPS z użyciem obustronnego uwierzytelniania za pomocą certyfikatów PKI (niekwalifikowanych). |  |  |
|  | Jeżeli dane transmitowane są w tunelu IPSec VPN[[34]](#endnote-34), to musi on spełniać następujące wymagania:   * Tryb pracy IPSec: ESP w trybie tunelowym; * Protokół negocjacji parametrów: IKE; * Metoda uwierzytelniania stron:   + Użytkownicy:     - Certyfikat PKI (niekwalifikowany);   + Systemy:     - Certyfikat PKI (niekwalifikowany); * Symetryczne algorytmy szyfrowania:   + AES;   + Minimalna długość klucza: 128 bitów. * Funkcje skrótu: HMAC-SHA-256 lub dłuższa; * Grupy Diffie-Hellman:   + Modular Exponential (MODP), grupy 5, lub 14-18.   + Elliptic Curve Group over GF[2^N] (EC2N), grupa 4. * Tryb negocjacji w fazie I:   + Main mode,   + Aggressive mode (zabroniony); * Czas ważności kluczy: 3600 sekund. |  |  |
|  | Rekomendowane jest składowanie danych na dyskach twardych w postaci zaszyfrowanej algorytmem zapewniającym należyty stopień poufności np. AES-128. |  |  |
|  | Parametry wykorzystywanych certyfikatów PKI powinny spełniać poniższe wytyczne:   * Algorytmy asymetryczne: RSA * Długość kluczy: 2048 bitów * Algorytm podpisu: SHA2-with-RSA * Maksymalny okres ważności certyfikatów: 1 rok * Regeneracja kluczy w trakcie odnowienia certyfikatów |  |  |
|  | Parametry haseł użytkowników aplikacji webowych powinny spełniać poniższe wytyczne:   * Hasła tworzone i przechowywane z użyciem soli * Długość soli: co najmniej 32 bajty * Sól[[35]](#endnote-35) generowana losowo * Inna sól dla każdego konta * Dopuszczalne algorytmy dla zabezpieczania haseł:   + Funkcje skrótu: SHA256, SHA512   + Scrypt, PBKDF2 (liczba iteracji: 64 000). |  |  |
|  | W przypadku budowy aplikacji z wykorzystaniem usług serwisowych (web services) należy stosować mechanizmy zapewniające poufność i integralność przesyłanych danych. |  |  |
|  | W celu zapewnienia poufności komunikacji należy stosować mechanizmy szyfrowania z wykorzystaniem SSL/TLS. |  |  |
|  | Należy poprawnie stosować mechanizmy uwierzytelniania serwera (certyfikat serwera musi być wydany przez zaufane CA) oraz jeśli jest to wymagane – uwierzytelniania użytkowników (również z wykorzystaniem certyfikatów). |  |  |
|  | W celu zapewnienia integralności przesyłanych informacji należy stosować SSL/TLS. |  |  |
|  | W przypadku, gdy przesyłane wiadomości są szczególnie wrażliwe, należy je szyfrować za pomocą silnych mechanizmów szyfrujących (należy wtedy wykorzystywać szyfrowanie na poziomie przesyłanych wiadomości a nie wyłącznie warstwie transportowej SSL/TLS). |  |  |
|  | Należy stosować mechanizmy chroniące przez atakami XML Denial of Service. |  |  |

## Logowanie zdarzeń i kontrola rozliczalności

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Aplikacja musi rejestrować wszystkie zdarzenia, które mogą być pomocne w monitorowaniu poziomu bezpieczeństwa, analizie, ewentualnym dochodzeniu lub zgłoszeniu objawów naruszenia bezpieczeństwa, w tym jako min.:   * udane i nieudane próby uwierzytelnienia, * założenie i usunięcie konta użytkownika * zablokowanie i odblokowanie konta użytkownika * aktywacja i dezaktywacja kont użytkowników, * zmiany ról i uprawnień użytkowników, * zmiany parametrów konfiguracyjnych, * próby przekraczania uprawnień. |  |  |
|  | Wszystkie mechanizmy logowania muszą być zaimplementowane na serwerze. |  |  |
|  | Zarejestrowane zdarzenia muszą mieć określoną strukturę, zakres oraz miejsce składowania. |  |  |
|  | Logowane zdarzenia na serwerze WWW muszą być zgodne ze znanym standardem (np. W3C) i jako minimum muszą zawierać informacje o:   * żądanym adresie URL * zastosowanej metodzie protokołu HTTP, * adresie IP użytkownika, * dacie i czasie wykonania żądania, * zwróconym kodzie serwera HTTP, * ilości przesłanych danych, * nagłówku user-agent oprogramowania użytkownika, * poziom ważności zdarzenia. |  |  |
|  | Logowane zdarzenia z aplikacji binarnych muszą korzystać z natywnych funkcji systemu operacyjnego (np. Event Log w systemach Windows). |  |  |
|  | Struktura logów aplikacji musi pozwalać na korelację zdarzeń z innymi systemami, w tym z systemami służącymi do analizy logów np. SIEM[[36]](#endnote-36). |  |  |
|  | Logi powinny być spójne dla całego systemu, a jeśli są one generowane osobno dla każdego z modułów systemu, powinna istnieć korelacja pomiędzy nimi, aby można było odtworzyć na ich podstawie spójny ciąg zdarzeń. W systemach o wysokim stopniu komplikacji i budowie modułowej wymagane jest, aby moduł odpowiedzialny za logowanie/raportowanie był odrębnym, dedykowanym modułem. |  |  |
|  | Logi aplikacji przetwarzających wrażliwe informacje muszą zapewniać możliwość odtworzenia całego przebiegu przetwarzania tych informacji wraz z możliwością odtworzenia każdej wprowadzonej zmiany przez poszczególnych użytkowników. |  |  |
|  | Pola logów z zaufanych oraz niezaufanych źródeł muszą być rozróżniane w dziennikach zdarzeń. |  |  |
|  | Zarejestrowane zdarzenia muszą być przechowywane w bezpieczny sposób i chronione przed nieautoryzowanym ujawnieniem lub modyfikacją przez zdefiniowany okres. |  |  |
|  | Informacje o zarejestrowanych zdarzeniach z krytycznych elementów aplikacji powinny być przesyłane do systemu, który pełni rolę centralnego serwera logów. W przypadku zastosowania centralnego serwera logów, rejestrowane zdarzenia muszą dodatkowo posiadać informację o urządzeniu/systemie/aplikacji, która jest ich źródłem. |  |  |
|  | Rejestrowane zdarzenia muszą być chronione przed nieautoryzowanym ujawnieniem, zmianą i zniszczeniem. Wszystkie dzienniki zdarzeń muszą mieć ustalone restrykcyjne prawa dostępu. Dla szczególnie wrażliwych komponentów aplikacji, rejestrowane zdarzenia muszą być przechowywane z zapewnieniem zachowania ich integralności poprzez zastosowanie mechanizmów kryptograficznej ochrony informacji (np. znakowanie czasem lub podpisy cyfrowe gromadzonych zdarzeń). |  |  |
|  | Informacje o zarejestrowanych zdarzeniach muszą być przechowywane w trybie dostępu przez minimum 90 dni. Po upływie tego czasu informacje o zdarzeniach mogą być archiwizowane lub usuwane, z zachowaniem zgodności z przepisami prawa |  |  |
|  | Zdarzenia dla wszystkich komponentów aplikacji powinny być rejestrowane z uwzględnieniem strefy czasowej: CET (UTC+1). |  |  |
|  | Znaczniki czasu uwzględniane w rejestrowanych informacjach o zdarzeniach powinny pochodzić z wiarygodnego źródła czasu (wymagana synchronizacja czasu aplikacji z wiarygodnym źródłem czasu). |  |  |
|  | Wszyscy użytkownicy aplikacji muszą być jednoznacznie zidentyfikowani (poprzez przydzielanie unikatowych identyfikatorów). |  |  |
|  | Wszystkie operacje wykonywane w aplikacji muszą być jednoznacznie powiązane z odpowiedzialnymi za ich wykonanie użytkownikami. |  |  |
|  | Komunikaty błędów widoczne dla użytkowników nie mogą zawierać danych potencjalnie wrażliwych (np. dane osobowe, dane nt. konfiguracji serwera, identyfikatory sesyjne użytkowników, kod aplikacji, ślady stosu). |  |  |
|  | Logowane zdarzenia nie mogą zawierać danych wrażliwych (np. dane osobowe, identyfikatory sesyjne, dane konfiguracyjne serwera, hasła). |  |  |
|  | Moduł służący do przeglądania logów musi posiadać mechanizmy ochrony przez wykonaniem (potraktowaniem jako kod) niezaufanych danych będących treścią logów aplikacji. |  |  |
|  | W przypadku uwierzytelniania w oparciu o identyfikator i hasło, aplikacja musi wyświetlać ogólny komunikat, z którego nie będzie można wywnioskować czy błędnie został podany identyfikator czy hasło dostępowe. |  |  |
|  | Oprogramowanie musi używać jednej implementacji logowania zdarzeń aplikacyjnych. |  |  |
|  | Wszystkie niedrukowalne znaki oraz separatory pól muszą być poprawnie enkodowane w dziennikach zdarzeń. |  |  |
|  | Logowanie zdarzenia musi wystąpić przed wykonaniem akcji. Jeżeli logowanie się nie powiodło (np. przez brak miejsca na dysku, brakujące uprawnienia) aplikacja powinna kończyć pracę bezpiecznie. |  |  |
|  | Logi techniczne aplikacji (np. debugowe) powinny być oddzielone od logów audytowych. |  |  |
|  | System powinien zapewniać rozliczalność działań administracyjnych. Między innymi w tym celu powinny istnieć indywidualne konta administracyjne (niedopuszczalne jest tworzenie administracyjnych kont grupowych). |  |  |

## Wytyczne aplikacji mobilnych

Poniższe wymagania stanowią wytyczne wymagań bezpieczeństwa określone jako minimum dla przygotowania i dostarczenia aplikacji działającej na urządzeniach mobilnych.

Urządzenia mobilne pracują pod systemem iOS i Android nie starszym niż 5 wersji bazowych, po weryfikacji wymagań bezpieczeństwa, jest potrzebna akceptacja wersji systemu.

Wytyczne bazują na otwartym projekcie OWASP **dla aplikacji mobilnych** oraz na dobrych praktykach wykorzystywanych w spółkach grupy ORLEN S.A.

Aplikacja mobilna posiadająca backend dodatkowo powinna spełniać wymagani opisane w pkt. I-VI.

Aplikacja mobilna dodatkowo powinna spełniać wymagani opisane w pkt. VII-X

### Aplikacje Mobilne - Architektura, projektowanie i modelowanie zagrożeń

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Wszystkie elementy aplikacji są zidentyfikowane, są znane i niezbędne do realizacji zakładanych funkcji, nie są wykorzystywane nieznane elementy kodu, bibliotek, innych aplikacji lub ich fragmenty, |  |  |
|  | Kontrole i mechanizmy bezpieczeństwa nigdy nie są wymuszane tylko po stronie klienta, ale także na po stronie backendu. |  |  |
|  | Jest zdefiniowana wysokopoziomowa architektura i obejmuje obszar bezpieczeństwa dla aplikacji i wszystkich podłączonych usług. |  |  |
|  | Dane uznane za wrażliwe są jasno określone i wskazane w aplikacji. |  |  |
|  | Wszystkie elementy aplikacji mają zdefiniowane funkcje biznesowe oraz funkcje bezpieczeństwa, które dostarczają. |  |  |
|  | Aplikacja mobilna i związane z nią usługi mają opracowany model zagrożeń, który je identyfikuje i określa środki zapobiegawcze. |  |  |
|  | Mechanizmy bezpieczeństwa są wdrażane w sposób scentralizowany. |  |  |
|  | Zarządzanie kluczami kryptograficznymi (jeśli istnieją) spełnia wymagania standardu zarządzania kluczami zgodnie z NIST SP 800-57. |  |  |
|  | Aplikacja posiada mechanizm, który wymusza aktualizację aplikacji mobilnej. |  |  |
|  | Na wszystkich etapach cyklu rozwoju oprogramowania są uwzględnione niezbędne elementy bezpieczeństwa. |  |  |
|  | Aplikacja lub jej elementy posiada możliwość ujawnienia informacji z urządzeń mobilnych lub logów, gdzie dane systemowe są odseparowane od danych użytkownika/biznesowych. |  |  |
|  | Aplikacja spełnia wymagania zgodności z prawem i przepisami dotyczącymi ochrony prywatności. |  |  |

### Aplikacje Mobilne - Przechowywanie danych i ochrona prywatności

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Dane wrażliwe takie jak dane osobowe, dane uwierzytelniające użytkownika lub klucze kryptograficzne, są przechowywane w zabezpieczonym magazynie. |  |  |
|  | Dane wrażliwe nie powinny być przechowywane poza kontenerem aplikacji lub poza rozwiązaniem, które służy do przechowywania poświadczeń. |  |  |
|  | Logi aplikacji nie zawierają wrażliwych danych. |  |  |
|  | Dane wrażliwe nie mogą być udostępniane stronom trzecim, chyba że jest to niezbędny element architektury. |  |  |
|  | Podczas wprowadzania poufnych danych, pamięć podręczna klawiatury jest wyłączona. |  |  |
|  | Żadne dane wrażliwe nie są ujawniane za pośrednictwem mechanizmów przekazywania danych między procesami wykorzystanymi do działania aplikacji. |  |  |
|  | W interfejsie użytkownika nie są widoczne wrażliwe dane, takie jak hasła lub PIN. |  |  |
|  | Kopie zapasowe wygenerowane przez system mobilny nie zawierają wrażliwych danych. |  |  |
|  | Aplikacja usuwa wrażliwe dane z widoku po przeniesieniu jej do warstwy tła. |  |  |
|  | Aplikacja nie przechowuje wrażliwych danych w pamięci dłużej niż to konieczne, a pamięć jest czyszczona zaraz po użyciu. |  |  |
|  | Aplikacja wymusza politykę bezpieczeństwa dostępu do urządzenia, np. wymagając od użytkownika ustawienia hasła do urządzenia mobilnego. |  |  |
|  | Aplikacja edukuje użytkownika w zakresie przetwarzanych danych wrażliwych, jak również najlepszych praktyk bezpieczeństwa, które użytkownik powinien stosować podczas korzystania z aplikacji |  |  |
|  | Wrażliwe dane nie powinny być przechowywane lokalnie na urządzeniu. Powinny być pobierane ze zdalnych źródeł w razie potrzeby i przechowywane tylko w pamięci podręcznej. |  |  |
|  | Jeśli dane wrażliwe muszą być przechowywane lokalnie lub przekazywane poza aplikację, powinny być zaszyfrowane przy użyciu klucza, który wymaga uwierzytelnienia. |  |  |
|  | Miejsce przechowywania danych lokalnie dla aplikacji powinno zostać wyczyszczone po określonej liczbie nieudanych prób logowania. |  |  |

### Aplikacje Mobilne – Kryptografia

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Aplikacja może korzystać z różnych metod szyfrowania, nie powinna bazować jedynie na kryptografii symetrycznej bazującej na kluczu przechowywanym na urządzeniu. Należy zabezpieczyć dostęp do klucza, jeżeli jest przechowywany na urządzeniu. |  |  |
|  | Aplikacja wykorzystuje bezpieczne mechanizmy kryptograficzne (bez znanych podatności). |  |  |
|  | Aplikacja wykorzystuje kryptografię, odpowiednio do danego zastosowania, skonfigurowaną zgodnie z najlepszymi praktykami. |  |  |
|  | Aplikacja nie wykorzystuje protokołów kryptograficznych ani algorytmów, które są powszechnie uważane za przestarzałe. |  |  |
|  | Aplikacja nie używa ponownie tego samego klucza kryptograficznego do wielu celów. |  |  |
|  | Wszystkie wartości losowe są generowane za pomocą zatwierdzonego generatora liczb losowych z modułu kryptograficznego. |  |  |
|  | W wersji aplikacji produkcyjnej, wszystkie hasła klucze i elementy kryptograficzne używane podczas testów, muszą zostać na nowo wygenerowane. |  |  |

### Aplikacje Mobilne - Uwierzytelnianie i zarządzanie sesjami

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Jeśli aplikacja zapewnia użytkownikom dostęp do zdalnej usługi, to na urządzeniu mobilnym musi dojść do formy uwierzytelnienia, np. za pomocą nazwy użytkownika/hasła. |  |  |
|  | W przypadku statycznego zarządzania sesją, urządzenie mobilne używa losowo wygenerowanych identyfikatorów sesji do uwierzytelniania klienta bez wysyłania danych uwierzytelniających użytkownika. |  |  |
|  | Jeśli wykorzystywane jest uwierzytelnianie oparte na tokenach to serwer dostarcza token, który został podpisany przy użyciu bezpiecznego algorytmu. |  |  |
|  | Urządzenie mobilne kończy istniejącą sesję, w momencie wylogowania się użytkownika. |  |  |
|  | W aplikacji jest zaimplementowana polityka haseł, która jest wymuszona na urządzeniu końcowym. W znaczeniu zarówno backendu jak i front endu w zależności o przyjęty metody uwierzytelnienia. |  |  |
|  | Na urządzeniu końcowym jest zaimplementowany mechanizm zabezpieczający przed wielokrotnym wprowadzeniem błędnych poświadczeń. W znaczeniu zarówno backendu jak i front endu w zależności o przyjęty metody uwierzytelnienia. |  |  |
|  | Po upływie uprzednio zdefiniowanego okresu braku aktywności, sesja na urządzeniu zdalnym jest przerywana i wygasa token dostępowy. |  |  |
|  | Uwierzytelnianie biometryczne, jeśli istnieje, nie jest związane ze zdarzeniem (tzn. używa API, które po prostu zwraca "prawda" lub "fałsz"). Zamiast tego, opiera się na odblokowaniu miejsca, w którym przechowywane są dane biometryczne. |  |  |
|  | Na urządzeniu mobilnym powinien być wykorzystywany więcej niż jeden czynnik uwierzytelniający do aplikacji. |  |  |
|  | Wrażliwe transakcje wymagają dodatkowego uwierzytelnienia. |  |  |
|  | Aplikacja informuje użytkownika o wszystkich aktywnościach wykonywanych na jego koncie. Użytkownicy mają możliwość przeglądania listy urządzeń i dodatkowych informacji jak adres IP, lokalizacja, itp., oraz blokowania określonych urządzeń. |  |  |
|  | Autoryzacja powinna być zdefiniowana, wymuszona i walidowana po stronie backendu. |  |  |

### Aplikacje Mobilne – Komunikacja sieciowa

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Dane są szyfrowane w sieci za pomocą protokołu TLS. Kanał bezpieczny jest wykorzystywany w całej aplikacji. |  |  |
|  | Ustawienia TLS są zgodne z aktualnymi najlepszymi praktykami lub tak bliskie, jak to tylko możliwe, jeśli mobilny system operacyjny nie obsługuje zalecanych standardów. |  |  |
|  | Aplikacja weryfikuje certyfikat X.509 na urządzeniu mobilnym po ustanowieniu bezpiecznego kanału. Akceptowane są jedynie certyfikaty podpisane przez zaufany urząd certyfikacji oraz są ważne, lub nie został unieważniony. |  |  |
|  | Aplikacja albo korzysta z własnego magazynu certyfikatów, albo podpina certyfikat lub klucz publiczny back-endu (zdalna usługa), i nie nawiązuje połączeń z back-endem (zdalną usługą) oferującymi inny certyfikat lub klucz, nawet jeżeli są one podpisane przez zaufany urząd certyfikacji. |  |  |
|  | Aplikacja nie może opierać się na jednym niepewnym kanale komunikacyjnym (e-mail lub SMS) dla krytycznych operacji, takich jak rejestracja i odzyskiwanie konta. |  |  |
|  | Aplikacja korzysta i bazuje na aktualnych połączeniach i aktualnych bibliotekach. |  |  |

### Aplikacje Mobilne – Interakcja między platformami

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Aplikacja żąda tylko minimalnego zestawu niezbędnych uprawnień. |  |  |
|  | Wszystkie dane pochodzące ze źródeł zewnętrznych oraz od użytkownika są walidowane i w razie potrzeby filtrowane. Obejmuje to dane otrzymywane za pośrednictwem UI, mechanizmu IPC, niestandardowych adresów URL i źródeł sieciowych. |  |  |
|  | Aplikacja nie eksportuje wrażliwych funkcji poprzez niestandardowe adresy URL, chyba że mechanizmy te są odpowiednio chronione. |  |  |
|  | Aplikacja nie eksportuje wrażliwych funkcji za pośrednictwem IPC, chyba że mechanizmy te są odpowiednio chronione. |  |  |
|  | JavaScript jest wyłączony w WebView, chyba że jest to wyraźnie wymagane. |  |  |
|  | WebView jest tak skonfigurowany, aby pozwalał tylko na minimalny wymagany zestaw obsługiwanych protokołów (idealnie tylko https). Potencjalnie niebezpieczne protokoły pliki, tel i app-id, są wyłączone. |  |  |
|  | Jeśli natywne metody aplikacji są narażone na działanie WebView, sprawdź, czy WebView renderuje tylko JavaScript zawarty w aplikacji. |  |  |
|  | Deserializacja obiektów, jeśli istnieje, jest realizowana przy użyciu bezpiecznej serializacji za pomocą API. |  |  |
|  | Aplikacja chroni się przed atakami nakładek ekranowych. (tylko Android) |  |  |
|  | Przed zamknięciem WebView należy wyczyścić pamięć podręczną, pamięć masową i załadowane zasoby (JavaScript, itp.). |  |  |
|  | Upewnij się, że aplikacja uniemożliwia korzystanie z niestandardowych klawiatur innych firm w przypadku wprowadzania poufnych danych. |  |  |

### Aplikacje Mobilne - Jakość kodu i ustawienia tworzenia

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Aplikacja jest podpisana i zaopatrzona w ważny certyfikat, którego klucz prywatny jest właściwie chroniony. |  |  |
|  | Aplikacja została zbudowana w trybie wersjonowania, z ustawieniami odpowiednimi dla danego wydania (np. non-debuggable). |  |  |
|  | Symbole debuggowania zostały usunięte z rodzimych binariów. |  |  |
|  | Usunięto kod do debugowania oraz kod pomocniczy (np. kod testowy, backdoor, ukryte ustawienia). Aplikacja nie rejestruje błędów ani wiadomości o debugowaniu. |  |  |
|  | Wszystkie komponenty zewnętrznych dostawców używane przez aplikację mobilną, takie jak biblioteki i frameworki, są aktualne, znane i sprawdzane pod kątem znanych podatności. |  |  |
|  | Aplikacja wyłapuje i obsługuje możliwe wyjątki. |  |  |
|  | Logika zarządzania błędami w mechanizmach, bezpieczeństwa domyślnie zabrania do nich dostępu. Tylko osoby posiadające uprawnienia do przeglądania logów mogą to robić, domyślnie to uprawnienie jest ograniczone tylko do administratora. Logi przekazywane do usług typu Crashlitics nie powinny przekazywać poufnych danych. |  |  |
|  | W kodzie niezarządzanym, pamięć jest alokowana, zwalniana i używana w bezpieczny sposób. (W kodzie niezarządzanym (C/C++), memory leak) |  |  |
|  | Darmowe funkcje bezpieczeństwa, takie jak minifikacja kodu, ochrona stosów, obsługa PIE i automatyczne zliczanie referencji, są aktywowane. |  |  |

### Aplikacje Mobilne – Odporność na działanie inżynierii odwrotnej

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Aplikacja rozpoznaje złamanie zabezpieczeń systemu pozwalające na podniesienie uprawnień (jailbreak/rooted), ostrzegając użytkownika lub zamykając aplikację. |  |  |
|  | Aplikacja zapobiega debugowaniu i/lub wykrywa i reaguje na dołączenie debuggera. Wszystkie dostępne protokoły do debuggowania muszą być uwzględnione. |  |  |
|  | Aplikacja wykrywa i reaguje próby manipulacji plikami wykonywalnymi i krytycznymi danymi np. plikami konfiguracyjnymi, bibliotekami, certyfikatami (we własnej piaskownicy). Powinna zostać zapewniona weryfikacja integralności tych plików oraz podpisów cyfrowych plików wykonywalnych. |  |  |
|  | Aplikacja wykrywa i reaguje na obecność na urządzeniu narzędzi inżynierii wstecznej. |  |  |
|  | Aplikacja wykrywa i reaguje na uruchomienie w emulatorze. |  |  |
|  | Aplikacja wykrywa i reaguje na manipulację kodem i danymi w swojej własnej przestrzeni pamięci. |  |  |
|  | Aplikacja wdraża wiele mechanizmów zabezpieczeń (8.1 do 8.6). Należy zwrócić uwagę, że siła odporności rośnie wraz ze wzrostem liczby zastosowanych mechanizmów i ich różnorodnością oraz oryginalnością. |  |  |
|  | Mechanizmy wykrywania wyzwalają różnego rodzaju reakcje, w tym opóźnione i ukryte. |  |  |
|  | Kod źródłowy aplikacji jest zaciemniony i nie jesteśmy w łatwy sposób go odczytać. |  |  |
|  | Aplikacja implementuje funkcję "parowania urządzenia" za pomocą unikalnych danych urządzenia, który wynika (jest generowana) na podstawie określonego zbioru cech tego urządzenia. |  |  |
|  | Wszystkie pliki wykonywalne i biblioteki należące do aplikacji są albo zaszyfrowane na poziomie pliku i/lub fragmentu kodu. Analiza statyczna nie jest w stanie ujawnić kodu lub danych. |  |  |
|  | W przypadku przetwarzania poufnych danych zaciemnienie kodu powinno być na poziomie uniemożliwiającym odtworzenie kodu metodami automatycznymi i manualnymi. Każdorazowo powinny zostać wykonane testy odporności na możliwość odtworzenia zaciemnionego kodu. |  |  |
|  | W przypadku potrzeby zwiększenia zabezpieczenia poufności komunikacji, należy stosować metody szyfrowania komunikatów na poziomie warstwy aplikacji. |  |  |

## Dokumentacja i szkolenia

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Dostarczenie dokumentacji projektu aplikacji. |  |  |
|  | Dostarczenie dokumentacji instalacji aplikacji. |  |  |
|  | Dostarczenie dokumentacji konfiguracji aplikacji i oprogramowania zawierającej opis zastosowanych zabezpieczeń, które wynikały z przedstawionych wymagań oraz schemat architektury rozwiązania (z wymienionymi portami i protokołami używanymi do komunikacji i transmisji informacji). |  |  |
|  | Dostarczenie dokumentacji użytkowej (eksploatacyjnej) aplikacji z podziałem dla odbiorców (użytkowników biznesowych i technicznych). |  |  |
|  | Dostarczenie dokumentacji związanej z usystematyzowaniem obsługi błędów i incydentów bezpieczeństwa. |  |  |
|  | Dostarczenie dokumentacji związanej z innymi elementami, np. procedurami pracy (monitorowanie poprawności działania, tworzenia kopii zapasowych i odtwarzania ich po awarii), zgodności z regulacjami (np. Ustawą o Ochronie Danych Osobowych), zastosowanymi interfejsami do innych aplikacji i systemów. |  |  |
|  | Dostarczenie szkolenia lub szkoleń związanych z zapewnianiem bezpieczeństwa aplikacji (integralności, poufności i dostępności). |  |  |

## Ciągłość działania

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Każdy element dostarczanej aplikacji (np. front-end, repozytorium danych) powinien umożliwiać zastosowanie dodatkowych elementów realizujących redundancje. |  |  |
|  | Dla wszystkich elementów krytycznych dla zapewnienia ciągłości działania aplikacji powinna zostać zastosowana nadmiarowość (w postaci zapewnienia redundancji tych zasobów). |  |  |
|  | W przypadku aplikacji intranetowych aplikacja musi być zgodna z przeglądarką z aktualnego Korporacyjnego Standardu Informatycznego (KSI). Zgodność aplikacji internetowej z wiodącymi na rynku przeglądarkami internetowymi (Internet Explorer w wersji 7,8,9,10,11 oraz najaktualniejszymi wersjami: Google Chrome, Mozilla Firefox, Safari, Opera) powinna być zapewniona przez ustalony okres. |  |  |
|  | Zgodność aplikacji z wiodącymi na rynku systemami operacyjnymi powinna być zapewniona przez ustalony okres. |  |  |
|  | Wszystkie wykryte w trakcie testów bezpieczeństwa, podatności krytyczne, wysokie i średnie powinny być wyeliminowane przed fazą produkcyjną. |  |  |
|  | Środowisko produkcyjne musi być pozbawione wszelkich zbędnych dla działania produkcyjnego komponentów, pakietów a zwłaszcza narzędzi developerskich, kompilatorów, itp. |  |  |
|  | Musi być zapewniona możliwość logowania się administratora do systemu operacyjnego na którym uruchomiona jest aplikacja. |  |  |

## Warunki wsparcia

| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
| --- | --- | --- | --- |
|  | Firma dostarczająca aplikację powinna zapewnić wsparcie. |  |  |
|  | W umowie serwisowej uwzględnione powinny być elementy takie jak czas reakcji, czas naprawy, RPO, RTO[[37]](#endnote-37). |  |  |
|  | Zgłoszenia powinny być przyjmowane przynajmniej jedną z dróg: telefoniczną, za pomocą e-mail czy dedykowanego serwisu internetowego. |  |  |
|  | Powinna zostać ustalona priorytetyzacja zgłoszeń oraz czas reakcji dotyczący zgłoszeń o danym priorytecie. |  |  |
|  | Wszystkie podatności muszą być naprawione przez dostawcę. Podatność rozumiana jest jako opisana i wciągnięta do ewidencji luka w zabezpieczeniach opublikowana w powszechnie znanych, publicznych i uznawanych za wiarygodne katalogach takich, jak CVE, CCE, biuletyn bezpieczeństwa Microsoft i innych (np. dostępnych poprzez http://web.nvd.nist.gov/view/vuln/search) |  |  |

## Zgodność ze standardem korporacyjnym

|  |  |  |  |
| --- | --- | --- | --- |
| Lp. | Opis | SPEŁNIA / NIESPEŁNIA | Uwagi/Uzasadnienie warunkowego odstąpienia |
|  | Aplikacja powinna być pisana w sposób kompatybilny z aktualnie obowiązującym Korporacyjnym Standardem Informatycznym w ORLEN S.A. |  |  |

## Przypisy

1. **URL (ang. Uniform Resource Locator)** – oznacza ujednolicony format adresowania zasobów (informacji, danych, usług) stosowany w Internecie i w sieciach lokalnych. [↑](#endnote-ref-1)
2. **MAC (ang. Mandatory Access Control - obowiązkowa kontrola dostępu)** – sposób kontroli dostępu w systemie komputerowym. W MAC wyróżnia się podmioty (ang. subject) oraz obiekty (ang. object). Podmiot to użytkownik lub proces. Obiektami mogą być przykładowo: pliki, katalogi, urządzenia, klucze rejestru. Zarówno podmioty jak i obiekty posiadają określone atrybuty bezpieczeństwa. System posiada również zestaw reguł (ang. rule) określających zasady dostępu, zwany polityką (ang. policy). System operacyjny na podstawie atrybutów bezpieczeństwa i polityki udziela bądź odmawia podmiotowi dostępu do obiektu. Zarówno atrybuty bezpieczeństwa jak i polityka są ustalane wyłącznie przez administratora systemu. W odróżnieniu od DAC, użytkownik nie ma wpływu na działanie mechanizmów kontroli dostępu. [↑](#endnote-ref-2)
3. **RBAC (ang. Role Based Access Control)** – sposób kontroli dostępu w systemach komputerowych. W RBAC głównym elementem są role pełnione przez użytkowników w systemie. Dostęp określany jest na podstawie roli jaką pełni użytkownik. [↑](#endnote-ref-3)
4. **Framework (platforma programistyczna)** - jest szkieletem do budowy aplikacji. Definiuje on strukturę aplikacji oraz ogólny mechanizm jej działania, a także dostarcza zestaw komponentów i bibliotek ogólnego przeznaczenia do wykonywania określonych zadań. Programista tworzy aplikację, rozbudowując i dostosowując poszczególne komponenty do wymagań realizowanego projektu, tworząc w ten sposób gotową aplikację. [↑](#endnote-ref-4)
5. **Internet Information Services** (**IIS**) – jest zbiorem usług internetowych dla systemów rodziny Microsoft Windows. [↑](#endnote-ref-5)
6. **Single Sign-on (logowanie pojedyncze)** - możliwość jednorazowego zalogowania się do usługi sieciowej i uzyskania dostępu do wszystkich autoryzowanych zasobów zgodnych z tą usługą. [↑](#endnote-ref-6)
7. **Sandbox (piaskownica)** - mechanizm izolacji uruchomionych procesów mający na celu separację uruchomionego kodu od pozostałych procesów działających w systemie. [↑](#endnote-ref-7)
8. **Infrastruktura klucza publicznego (ang. Public Key Infrastructure (PKI))** – w ogólności jest to zespół urządzeń, oprogramowania, ludzi, polityk oraz procedur umożliwiający tworzenie, przechowywanie, zarządzanie i rozprowadzanie cyfrowych certyfikatów klucza publicznego. W szczególności jest to szeroko pojęty kryptosystem, w skład, którego wchodzą urzędy certyfikacyjne (CA), urzędy rejestracyjne (RA), subskrybenci certyfikatów klucza publicznego (użytkownicy), oprogramowanie oraz sprzęt. Infrastruktura klucza publicznego tworzy hierarchiczną strukturę zaufania, której podstawowym dokumentem jest certyfikat klucza publicznego. Najpopularniejszym standardem certyfikatów PKI jest X.509 w wersji trzeciej. [↑](#endnote-ref-8)
9. **Hash** - liczba obliczona przez funkcję hashującą (jednokierunkową funkcję skrótu). [↑](#endnote-ref-9)
10. **HTTP (ang. Hypertext Transfer Protocol – protokół przesyłania dokumentów hipertekstowych**) - to protokół sieci WWW (ang. World Wide Web). Obecną definicję HTTP stanowi RFC 2616. Za pomocą protokołu HTTP przesyła się żądania udostępnienia dokumentów WWW i informacje o kliknięciu odnośnika oraz informacje z formularzy. Zadaniem stron WWW jest publikowanie informacji – natomiast protokół HTTP właśnie to umożliwia. [↑](#endnote-ref-10)
11. **HTTPS (ang. HyperText Transfer Protocol Secure)** - szyfrowana wersja protokołu HTTP. W przeciwieństwie do komunikacji niezaszyfrowanego tekstu w HTTP klient-serwer, szyfruje go za pomocą protokołu SSL. Zapobiega to przechwytywaniu i zmienianiu przesyłanych danych. [↑](#endnote-ref-11)
12. **Cookie (ciasteczko)** - niewielkie informacje tekstowe, wysyłane przez serwer WWW i zapisywane po stronie użytkownika (zazwyczaj na twardym dysku). Domyślne parametry ciasteczek pozwalają na odczytanie informacji w nich zawartych jedynie serwerowi, który je utworzył. Ciasteczka są stosowane najczęściej w przypadku liczników, sond, sklepów internetowych, stron wymagających logowania, reklam i do monitorowania aktywności odwiedzających. [↑](#endnote-ref-12)
13. **Token** – losowa liczba, która jednoznacznie identyfikuje sesję użytkownika. [↑](#endnote-ref-13)
14. **Atak siłowy (ang. brute-force)** - określenie algorytmu, który opiera się na sukcesywnym sprawdzeniu wszystkich możliwych kombinacji hasła dostępowego. [↑](#endnote-ref-14)
15. **Enkodowanie/Escapowanie** – mechanizm polegający na usuwaniu i/lub kodowaniu znaków specjalnych w sposób, który uniemożliwi systemowi ich interpretację w sposób inny niż przewidział to programista. [↑](#endnote-ref-15)
16. **HTML (ang. HyperText Markup Language)** – hipertekstowy język znaczników, obecnie szeroko wykorzystywany do tworzenia stron internetowych. [↑](#endnote-ref-16)
17. **JavaScript, JS** – skryptowy język programowania, stworzony przez firmę Netscape, najczęściej stosowany na stronach internetowych. Pod koniec lat 90. XX wieku organizacja ECMA wydała na podstawie JavaScriptu standard języka skryptowego o nazwie ECMAScript. [↑](#endnote-ref-17)
18. **CSS** - Kaskadowe arkusze stylów (ang. Cascading Style Sheets; w skrócie CSS) to język służący do opisu formy prezentacji (wyświetlania) stron WWW. CSS został opracowany przez organizację W3C w 1996 r. jako potomek języka DSSSL przeznaczony do używania w połączeniu z SGML-em. [↑](#endnote-ref-18)
19. **URI** - Uniform Resource Identifier (ang. Uniform Resource Identifier) jest standardem internetowym umożliwiającym łatwą identyfikację zasobów w sieci. Zdefiniowany jest w dokumencie RFC 2396. URI jest zazwyczaj łańcuchem znaków, zapisanym zgodnie ze składnią określoną w standardzie. Łańcuch ten określa nazwę (URN) lub adres (URL) zasobu, identyfikowanego przez dany URI. URI może zostać sklasyfikowane jako URL (ang. Uniform Resource Locator) lub URN (ang. Uniform Resource Name). Szczególnym przypadkiem URI jest URL, który oprócz identyfikacji zasobu wskazuje również sposób dostępu do niego. [↑](#endnote-ref-19)
20. **SQL (ang. Structured Query Language)** – strukturalny język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych. [↑](#endnote-ref-20)
21. **XML (ang. Extensible Markup Language)** – uniwersalny język formalny przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. [↑](#endnote-ref-21)
22. **LDAP (ang. Lightweight Directory Access Protocol)** – protokół przeznaczony do korzystania z usług katalogowych, bazujący na standardzie X.500. Jest to również nazwa usługi katalogowej pozwalającej na wymianę informacji za pośrednictwem TCP/IP. [↑](#endnote-ref-22)
23. **HSM (ang. Hardware Security Module)** – Sprzętowy moduł kryptograficzny. System służący między innymi do bezpiecznego przechowywania kluczy poufnych. [↑](#endnote-ref-23)
24. **Master secret –** jest to kluczy tajny (kluczy asymetryczny) powiązany z użytkownikiem. [↑](#endnote-ref-24)
25. **TLS (ang. Transport Layer Security)** – przyjęte jako standard w Internecie rozwinięcie protokołu **SSL (ang. Secure Socket Layer)**, zaprojektowanego pierwotnie przez Netscape Communications. TLS zapewnia poufność i integralność transmisji danych, a także uwierzytelnienie serwera, a niekiedy również klienta. Opiera się na szyfrowaniu asymetrycznym oraz certyfikatach X.509. [↑](#endnote-ref-25)
26. **CA** - Urząd certyfikacji, centrum certyfikacji (ang. certificate authority lub certification authority, w skrócie CA) – w kryptografii jest to podmiot, który wystawia certyfikaty cyfrowe. Certyfikat potwierdza własność klucza publicznego poprzez wskazanie podmiotu certyfikatu. Pozwala to innym powołującym się stronom polegać na podpisach lub zapewnieniach złożonych przez klucz prywatny odpowiadającemu kluczowi publicznemu, który jest certyfikowany. W tym modelu relacji zaufania CA jest zaufaną stroną trzecią, której zawierzają zarówno podmiot (właściciel) certyfikatu oraz strona polegająca. Urzędy certyfikacji są charakterystyczne dla wielu systemów infrastruktury klucza publicznego (PKI). [↑](#endnote-ref-26)
27. **DoS (ang. Denial of Service – odmowa usługi)** – atak na system komputerowy lub usługę sieciową w celu uniemożliwienia działania. [↑](#endnote-ref-27)
28. **Atak BEAST (Browser Exploit Against SSL/TLS)** - atak na protokół SSL/TLS mający na celu deszyfrację komunikacji. [↑](#endnote-ref-28)
29. **Atak CRIME (Compression Ratio Info-leak Made Easy)** - atak na protokół SSL/TLS mający na celu deszyfrację komunikacji. [↑](#endnote-ref-29)
30. **MITM (ang. Man in the middle) –** atak kryptologiczny polegający na podsłuchu i modyfikacji wiadomości przesyłanych pomiędzy dwiema stronami bez ich wiedzy. [↑](#endnote-ref-30)
31. **MITM Plaintext Injection (ang. SSL/TLS Renegotiation Handshakes MITM Plaintext Data Injection)** – rodzaj ataku polegający na wymuszeniu renegocjacji komunikacji SSL/TLS. [↑](#endnote-ref-31)
32. **GUID (ang. Globally Unique Identifier) -** identyfikator globalnie unikatowy – jest identyfikatorem obiektów w systemie Windows. GUID jest generowany między innymi na podstawie czasu wygenerowania oraz liczb pseudolosowych. Ma on postać 32-znakowego ciągu heksadecymalnego. [↑](#endnote-ref-32)
33. **FIPS 140-2** – standard opracowany i wymagany przez rząd amerykański. Standard dotyczy akredytacji modułów kryptograficznych. [↑](#endnote-ref-33)
34. **VPN/IPsec (Virtual Private Network)** - wirtualna sieć prywatna. Można ją opisać jako "tunel", przez który płynie ruch w ramach sieci prywatnej pomiędzy klientami końcowymi za pośrednictwem publicznej sieci (takiej, jak Internet) w taki sposób, że węzły tej sieci są przezroczyste dla przesyłanych w ten sposób pakietów. Taki kanał może opcjonalnie kompresować lub szyfrować w celu zapewnienia lepszej jakości lub większego poziomu bezpieczeństwa przesyłanych danych. [↑](#endnote-ref-34)
35. **Sól (ang. Salt)** – są to dane dodawane podczas szyfrowania lub przygotowywania skrótu wiadomości (w tym skrótu hasła). [↑](#endnote-ref-35)
36. **SIEM (ang. Security Information and Event Management)** – system do archiwizacji oraz korelacji zdarzeń. Dodatkową funkcją jest możliwość wykrywania incydentów bezpieczeństwa w oparciu o zdarzenia pochodzące z systemów teleinformatycznych. [↑](#endnote-ref-36)
37. Recovery Point Objective – określa moment w przeszłości, w którym po raz ostatni została wykonana kopia danych i od którego to momentu będzie można odzyskać dane z backupu. Recovery Time Objective – określa maksymalny czas po awarii potrzebny do przywrócenia działania aplikacji systemów i procesów biznesowych. [↑](#endnote-ref-37)