Bezpieczeństwo przeglądarek internetowych

Ochrona użytkownika przed atakami na SSL/TLS

Zespół Bezpieczeństwa PCSS

security@man.poznan.pl

Wachlarz serwisów internetowych dostępnych dla użytkownika masowego rozszerza się nieustannie. Usługi, które związane są z transferem danych osobowych czy środków pieniężnych, są udostępniane internautom za pośrednictwem szyfrowanego protokołu HTTPS. Kryptograficzną metodą ochrony naszych danych są w tym przypadku protokoły SSL/TLS. Jak inne rozwiązania – także one mają pewne słabości, mogą zostać źle skonfigurowane, a ich funkcjonalność może zostać nadużyta.

Niniejszy dokument stanowi próbę porównania, w jaki sposób 5 najbardziej popularnych przeglądarek internetowych radzi sobie z niestandardowymi sytuacjami, które agresorzy mogliby sprowokować w celu zaatakowania – często niewystarczająco świadomych zagrożeń – użytkowników, także niezainteresowanych kwestiami technicznymi. Oprócz aspektów bezpieczeństwa, poruszono także częściowo kwestie wydajności poszczególnych aplikacji.

Wersja 1.0 – październik 2010

Zespół Bezpieczeństwa PCSS zajmuje się analizami, badaniami oraz konsultacjami w zakresie bezpieczeństwa teleinformatycznego. Zabezpiecza infrastrukturę PCSS, bierze udział w wielu krajowych i europejskich projektach badawczych oraz świadczy usługi m.in. w zakresie testów penetracyjnych dla Klientów zewnętrznych. Więcej informacji na stronach zespołu (http://security.psnc.pl).

Skrót wyników

Niniejszy raport jest zbiorem badań zespołu nad podstawowymi funkcjami przeglądarek internetowych w obszarze budowy i obsługi tuneli SSL/TLS. Wykonane testy pozwalają na ogólną ocenę mechanizmów do walki z atakami na tunele SSL/TLS. Pokazano tu między innymi problemy związane z implementacją (i ustawieniami) poszczególnych tuneli w kontekście zarówno protokołów, jak i algorytmów szyfrowania, ale też ważną, a często pomijaną kwestię, którą jest interakcja z użytkownikiem. W raporcie pokazano także czasowe zależności przy przesyłaniu danych w tunelach w różnych konfiguracjach (wiele lub jeden tunel w obu kierunkach).

Z założenia, dokument ten nie był pisany, aby wskazać użytkownikowi „najlepszą" przeglądarkę. Także w ogólności nie znajdziemy tu całościowego stwierdzenia, iż przeglądarka X jest lepsza od przeglądarki Y. Autorzy raportu zakładają, iż Czytelnik po zapoznaniu się z tekstem raportu i, być może, wykonaniu samemu niektórych testów, będzie mógł stwierdzić, która z przeglądarek spełnia lepiej jego oczekiwania. Na podstawie swego doświadczenia autorzy raportu postawić mogą tezę, iż wykorzystanie tylko jednej przeglądarki (w ujęciu ogólnym, a nie tylko w kontekście tuneli SSL/TLS) jest rozwiązaniem bardzo ograniczającym możliwość postrzegania problemów, a także niepopularnym w środowisku osób technicznych, zajmujących się różnymi aspektami sieci WWW.

Poniższa tabela prezentuje skrót uzyskanych wyników w formie zbioru stwierdzeń „za i przeciw". Wyraźnie wynika z niej, że bardzo trudno o stawianie w analizowanym zakresie jednoznacznych ocen w odniesieniu do poszczególnych przeglądarek.

Przeglądarka	Plusy	Minusy
Mozilla Firefox	• Wyłączona obsługa SSLv2 • Informacje o połączeniu SSL/TLS • Szybka obsługa pojedynczych tuneli przy przesyłaniu danych klient → serwer • czytelność komunikatów o błędach • czytelność prezentacji stanu połączenia	• Standardowe ustawienie zapamiętywania wyjątków • Zła obsługa niektórych błędów
Internet Explorer	• Wyłączona obsługa SSLv2 • Szybka obsługa pojedynczych tuneli przy przesyłaniu danych klient → serwer • Szybka obsługa pojedynczych tuneli przy przesyłaniu danych serwer → klient • Szybka obsługa zwielokrotnionych tuneli przy przesyłaniu danych serwer → klient	• Brak informacji o tunelu SSL/TLS • Domyślna obsługa słabych algorytmów szyfrowania (LOW/EXP) • Niedostateczna informacja przy wystąpieniu błędów • Domyślna interpretacja kodu w plikach bez rozszerzeń

Przeglądarka	Plusy	Minusy
	• Informacja na stronie głównej o błędzie w tunelu mimo zatwierdzenia wyjątku	
Google Chrome	• Wyłączona obsługa SSLv2 • Informacje o połączeniu SSL/TLS • czytelność prezentacji stanu połączenia • Informacja na stronie głównej o błędzie w tunelu mimo zatwierdzenia wyjątku	• Brak informacji po zatwierdzeniu błędnego tunelu
Opera	• Wyłączona obsługa SSLv2 • Informacje o połączeniu SSL/TLS • Szybka obsługa pojedynczych tuneli przy przesyłaniu danych serwer → klient • Czytelność prezentacji stanu połączenia • Bardzo zaawansowana obsługa komunikacji z użytkownikiem, • Szerokie opisy problemów dla zaawansowanego użytkownika • Wzorowe ustawienia domyślne • Informacja o przerwaniu ściągania pliku	• Brak wskazania problematycznych elementów przy ukrytym tunelu SSL/TLS • Zła obsługa niektórych błędów
Safari	• Wyłączona obsługa SSLv2 • Szybka obsługa pojedynczych tuneli przy przesyłaniu danych serwer → klient	• Brak informacji o tunelu SSL/TLS • Brak dostatecznych informacji o błędach • Braki w poprawnej implementacji wyjątków • Błędy zatwierdzania pojedynczym kliknięciem (bezmyślnym) • Brak wskazania problematycznych elementów przy ukrytym tunelu SSL/TLS • Zła obsługa niektórych błędów

SPIS TREŚCI

1. Wstęp

1.1. Wprowadzenie

Przełom wieków XX i XXI z pewnością można określić jako czas pełnego rozkwitu Internetu. Jak łatwo sobie wyobrazić, dla większości jego użytkowników, kojarzy się on w większości z usługami dostępnymi za pośrednictwem przeglądarki internetowej. Dzięki przeglądarce i wbudowanym w nią mechanizmom możliwe jest obecnie wykorzystanie bardzo wielu technologii, które wcześniej nie były powszechnie dostępne. Mowa tu o obsłudze poczty przy pomocy klientów webowych, o transakcjach finansowych, zakupach internetowych czy nawet transmisjach multimedialnych (czego doskonałym przykładem są transmisje internetowe oraz możliwość późniejszego odtwarzania spotkań piłkarskich Mistrzostw Świata RPA 2010 na stronach Telewizji Polskiej). Możliwe jest aktualnie także obsługiwanie całych serwerowych systemów operacyjnych (a także urządzeń sieciowych, drukarek, punktów dostępowych sieci bezprzewodowych) przy wykorzystaniu jedynie przeglądarki.

Fenomen przeglądarek jest bardzo prosty i oczywisty – dzięki zastosowaniu aplikacji zgodnej z pewnymi ogólnie uznanymi standardami (np. specyfikacje HTTP, kaskadowych arkuszy styli CSS, XHTML i wiele innych, które można wyszukać np. pod adresem [23]) możliwa jest obsługa bardzo wielu technologii z dowolnego miejsca w Sieci – np. w kawiarenkach internetowych po przeciwnej stronie globu.

Chcąc korzystać świadomie z dobrodziejstw tych rozwiązań należy oczywiście pamiętać o bezpieczeństwie zarówno klienckiej, jak i serwerowej strony połączenia. Bezpieczeństwo, o którym tu mowa, należy rozpatrywać w jego wszystkich aspektach, czyli brać pod uwagę na równi poufność przesyłanych informacji, jak i jej dostępność oraz integralność. Ataków, które agresor może tu wykonać, jest prawdopodobnie tyle, co możliwości wykorzystania samej przeglądarki, czyli bardzo dużo.

Szyfrowane usługi w Internecie – łagodne wprowadzenie

Żeby zabezpieczyć dane przed podejrzeniem czy podsłuchaniem (tzn. zapewnić ich poufność), potrzebny jest tzw. **klucz szyfrujący**, zamieniający zabezpieczany tekst na ciąg zakodowanych znaków. Rozróżniamy dwa rodzaje szyfrowania: symetryczne i asymetryczne, mówimy nawet o kryptografii symetrycznej i asymetrycznej.

Kryptografia symetryczna opiera się na jednym kluczu, który jest współdzielony przez obie strony transmisji. Algorytmy szyfrujące i deszyfrujące są tak skonstruowane, że tylko tym samym kluczem można zaszyfrować wiadomość i następnie przywrócić jej postać pierwotną. Oczywiście klucz ten musi być tajny dla wszystkich innych poza stronami transmisji. Kryptografia symetryczna jest bardzo szybka i wymaga mniejszych długości klucza szyfrującego. Problemem pozostaje,

w jaki sposób strony transmisji miałyby wybrać i uzgodnić klucze, nie ujawniając ich nikomu innemu.

Kryptografia asymetryczna (zwana także kryptografią z **kluczem publicznym**) wykorzystuje specyficznie skonstruowaną parę kluczy: tajny (**prywatny**) i jawny (**publiczny**). Każda ze stron transmisji posiada swoją własną parę kluczy, przy czym klucze jawne są powszechnie dostępne. Klucze mają bardzo specyficzną właściwość – to, co zaszyfrowano jednym, można odszyfrować wyłącznie drugim. Klucze w kryptografii asymetrycznej są co najmniej kilkakrotnie dłuższe niż w symetrycznej, inne są również ich podstawy i założenia – co powoduje, że ten rodzaj kryptografii jest znacznie wolniejszy od kryptografii symetrycznej. Za to ogromną zaletą jest brak konieczności bezpośredniego uzgadniania czy przesyłania tajnego klucza.

Kryptografii asymetrycznej można też użyć do złożenia **podpisu cyfrowego**, czyli poświadczenia treści dokumentu (zapewnienia integralności danych). W tym miejscu trzeba wprowadzić kolejne pojęcie – **funkcji skrótu**. Istnieją pewne algorytmy kryptograficzne, które są funkcjami przetwarzającymi dowolny tekst we względnie krótki – zawsze tej samej długości – ciąg znaków (np. 20 lub 32 znaki). Funkcje te mają bardzo interesujące właściwości – łatwo zastosować je do policzenia skrótów, ale niezwykle trudno odwrócić, tzn. wywnioskować treść źródłową z funkcji skrótu. Ponadto nawet minimalna zmiana w tekście źródłowym powinna zaowocować zupełnie inną postacią skrótu. Wreszcie bardzo trudne musi być znalezienie dwóch różnych tekstów źródłowych, które dadzą identyczny skrót – taką sytuację określa się jako **kolizję**. Kolizje są nieuniknione (bo możliwych skrótów jest drastycznie mniej niż możliwych tekstów źródłowych), ale w zależności od użytego algorytmu (funkcji skrótu) można je odnaleźć łatwiej lub trudniej. Możliwych wartości skrótów jest jednak tyle, że przypadkowe natrafienie na kolizję jest wyjątkowo mało prawdopodobne.

Załóżmy, że Ala chce zaszyfrować dokument i przesłać go Bartkowi (chce zapewnić tylko poufność danych). Szyfruje wtedy treść dokumentu kluczem publicznym Bartka (który zna) i przesyła dokument. Jedyną osobą, która zna klucz prywatny Bartka powinien być właśnie adresat dokumentu – dzięki temu Bartek może bez problemów odszyfrować treść. Nie wie jednak na pewno, kto go wysłał – każdy mógł skorzystać z klucza publicznego Bartka i zaszyfrować nim dokument.

Jeżeli Ala chce podpisać dokument dla Bartka, wylicza skrót z dokumentu i szyfruje go swoim kluczem prywatnym. Przesyła Bartkowi dokument i wartość skrótu. Ponieważ wiedza, jakiego algorytmu używa się do obliczania skrótu, jest jawna, Bartek również może obliczyć skrót otrzymanego dokumentu i porównać dwie wartości skrótu: uzyskaną samodzielnie i otrzymaną od Ali. Tę drugą wartość otrzymuje, odszyfrowując przysłany skrót kluczem publicznym Ali (który jest – jak pamiętamy – jawny). Jeżeli wartości są równe – to po pierwsze, przesłanego dokumentu nikt nie zmodyfikował, a po drugie – jego autorem mogła być tylko Ala (skrót zaszyfrowany kluczem prywatnym kogoś innego, a odszyfrowany kluczem publicznym Ali miałby inną wartość).

Istnieje jednak pewien problem. Co się stanie, jeśli np. Ewa prześle Bartkowi swój klucz publiczny i powie, że jest Alą, po czym – jako Ala – przekaże mu swój podpisany dokument? Bartek nie będzie w stanie stwierdzić, czy osoba podająca się za Alę rzeczywiście nią jest. W tym miejscu pomagają nam Centra Autoryzacji (ang. *Certification Authority* – CA). Prawdziwa Ala może uzyskać od zaufanego na całym świecie CA specjalny certyfikat, poświadczający jej tożsamość (do tego celu Ala będzie musiała osobiście udać się do centrum lub jego oddziału), zawierający jej klucz publiczny i podpisany przy pomocy klucza prywatnego CA. Na zasadzie opisanej powyżej każdy, kto otrzyma od Ali podpisany dokument i zweryfikuje jej zaufany certyfikat, będzie mógł się upewnić co do jej tożsamości.

Podobnymi certyfikatami legitymują się (a przynajmniej powinny – jeśli tak nie jest, nie należy im ufać) serwisy internetowe oferujące usługi przez Internet przy pomocy szyfrowanego protokołu HTTPS, np. serwisy pocztowe czy bankowości elektronicznej. Przeglądarki internetowe dokonują zaś automatycznego sprawdzenia, czy certyfikat poświadczający tożsamość ma odpowiednie cechy: czy wystawiony został przez zaufane CA, czy jest ważny, czy zawarte w nim klucze kryptograficzne nie są zbyt słabe, czy certyfikat jest wystawiony dla strony internetowej, z którą łączy się użytkownik itd.

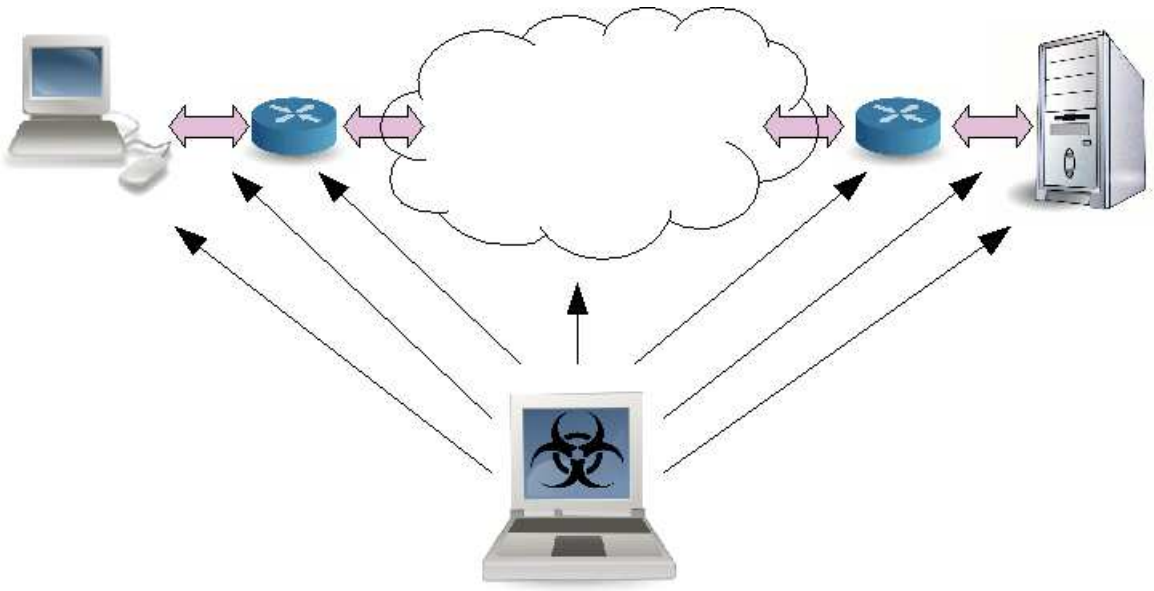
Dla zilustrowania tego faktu niech posłuży kilka prostych przykładów:

- dane potrzebne do przeprowadzenia transakcji finansowej zostają zmienione w trakcie ich przesyłania ich między przeglądarką a serwerem obsługującym transakcje. Podmieniony zostaje adresat przelewu, przez co uprawnione strony transakcji narażone są na straty finansowe. Nie zadziałały tu mechanizmy zapewniające poufność i integralność przesyłanych danych,
- serwer nękany dużą liczbą zapytań nie jest w stanie odpowiadać na wszystkie, więc część z nowo przychodzących zostaje odrzucona. Prawowity użytkownik nie ma możliwości otrzymania potrzebnych mu informacji, więc zaburzona została tu zasada dostępności informacji, co jest oczywiście problemem bezpieczeństwa,
- dane na temat połączenia dużych przedsiębiorstw zostają przechwycone w wyniku podsłuchania poczty elektronicznej przesyłanej między ich dyrektorami. Osoby znające wpływ tej informacji na rynek finansowy mają możliwość przewidzenia zachowań akcji owych przedsiębiorstw, przez co mogą uzyskać znaczące profity. Zawiodły tu mechanizmy zapewniające poufność.

W powyższych przykładach zauważyć można, jak niewiele potrzeba, aby utracić nie tylko informacje, ale i dalsze (już bardziej materialne) dobra, które są od nich silnie zależne.

Zapewnienie bezpieczeństwa na odpowiednim poziomie nie jest zadaniem trywialnym w tak złożonej strukturze, jaką jest Internet i wymaga znaczących nakładów pracy w bardzo różnych elementach środowiska. Przy analizie nawet najprostszych zależności, na przykład w architekturze

klient-serwer, zidentyfikować można wiele miejsc, gdzie należy zadbać o bezpieczeństwo (obrazuje je Rysunek 1).

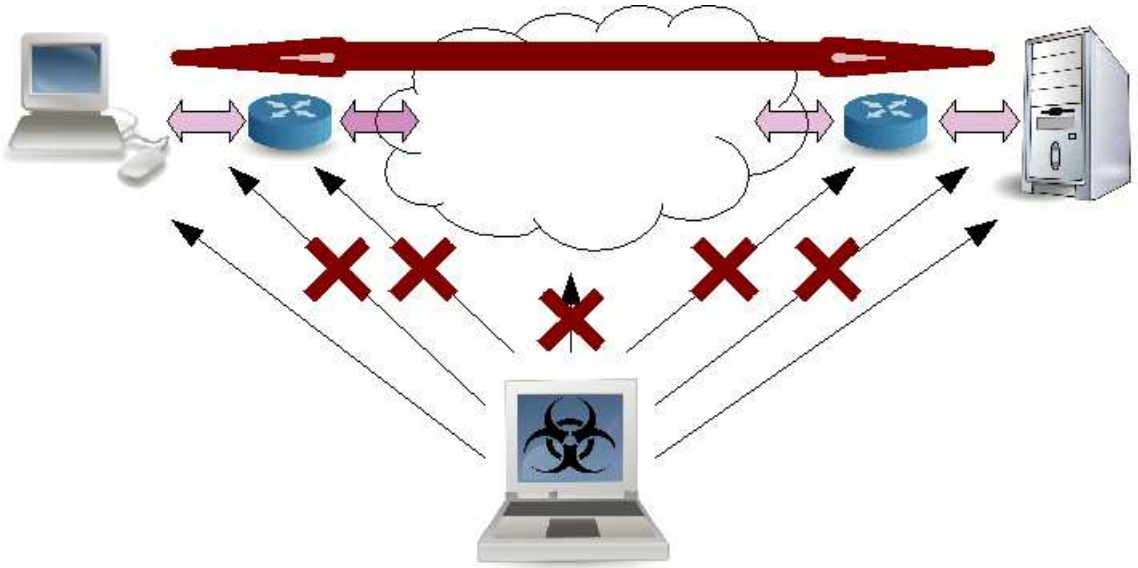


Rysunek 1. Miejsca możliwego zagrożenia atakiem informatycznym

Łatwo można tu wyróżnić takie miejsca ataku jak:

a) komputer kliencki,
b) lokalne połączenie klienta,
c) brama dostępowa łącząca klienta z Internetem,
d) połączenia realizowane w Internecie oraz urządzenia je obsługujące,
e) brama dostępowa sieci serwera,
f) sieć lokalna, w której umieszczone są serwery usługowe,
g) system serwerowy.

Mimo tak wielu możliwości ataku, istnieją mechanizmy potrafiące zapewnić bardzo wysoki poziom bezpieczeństwa oraz znaczącą redukcję wektora ataku. Jednym z takich mechanizmów jest szyfrowanie i tunelowanie ruchu za pomocą protokołów SSL/TLS, co przedstawia Rysunek 2.



Rysunek 2. Redukcja wektorów ataku przez wprowadzenie tuneli SSL/TLS

Protokoły TLS (ang. Transport Layer Security) oraz SSL (ang. Secure Socket Layer) [5] mają na celu zapewnienie poufności i integralności transmisji danych, a także zapewnienie bezpieczeństwa procesowi uwierzytelniania. Opierają się one na szyfrach asymetrycznych oraz certyfikatach standardu X.509 [6]. Należy w tym miejscu zaznaczyć, że w niniejszym raporcie nie są poruszane problemy zapewnienia bezpieczeństwa Centrów Autoryzacji (CA), co samo w sobie może być tematem na kolejny raport).

Podstawową cechą obu protokołów jest fakt, iż działają one w warstwie transportowej modelu ISO/OSI, więc można je zastosować do zabezpieczenia takich protokołów jak na przykład: Telnet, HTTP, POP3 czy IMAP.

Historia protokołu SSL/TLS

Protokół SSL (ang. *Secure Socket Layer*), opracowany pierwotnie przez firmę Netscape, zapewnia poufność i integralność transmisji danych przesyłanych przez Internet. Umożliwia również uwierzytelnianie stron połączenia (oparte na szyfrowaniu asymetrycznym i certyfikatach X509, obowiązkowe dla serwera i opcjonalne dla klienta). Intencją twórców było zaprojektowanie protokołu uniwersalnego, tak aby mogły z niego korzystać protokoły aplikacyjne (HTTP, FTP, Telnet itd.).

Pierwsza wersja protokołu nie doczekała się szerszej publikacji. Wydana w 1994 roku wersja 2.0 była pierwszą publicznie dostępną. Podobnie jak wersja 1, nie była ona projektowana przez profesjonalnych kryptografów (lecz inżynierów Netscape) i twórcom umknęło kilka istotnych aspektów. SSLv2 miał sporo słabości, między innymi pojawiła się poważna luka koncepcyjna w protokole negocjacji szyfru. Napastnik mógł w łatwy sposób wymusić na serwerze zastosowanie najsłabszego z szyfrów obsługiwanych przez obie strony. Szyfr taki był względnie łatwy do złamania. Więcej o słabościach SSL w wersji 2.0 można dowiedzieć się w raporcie [9] oraz w publikacji [1].

Odpowiedzią na wspomniane luki była wersja 3.0 protokołu SSL opublikowana w 1995 roku. Została ona opracowana przez kryptografa Paula Kochera i jej poziom bezpieczeństwa jest dużo wyższy (do dziś uważana jest za bezpieczną). SSL w wersji 3 jest kompatybilne wstecz z wersją poprzednią. Od 1999 roku wersja jest uznawana za powszechnie używaną i obowiązującą w odniesieniu do protokołu SSL. Wersja 2.0 miała być wspierana jedynie przez jakiś czas z powodów wstecznej kompatybilności, jednak obsługiwana była do niedawna – a niekiedy jest do dziś – w wielu popularnych aplikacjach (także w przeglądarkach internetowych). Wsparcie takie (jak również samo używanie SSL w wersji 2.0) jest jednak odradzane.

W 1999 roku został opublikowany dokument RFC 2246 [16], opisujący standard TLS (ang. *Transport Layer Security*) w wersji 1.0. Jest to zaadaptowany standard SSLv3. Różnice są niewielkie, funkcjonalnie praktycznie żadne – istotnym wyróżnikiem jest natomiast fakt, że protokół TLS w założeniu funkcjonuje jako niezależny, a nie tworzony przez określonego producenta (TLS rozwijany jest przez ciało standaryzujące Internet Engineering Task Force [15]). Aplikacje wspierające TLSv1 zazwyczaj wspierają również SSL w wersji 3.0 (protokół TLS 1.0 jest, przy założeniu zgodności specyfikacji z implementacją, wstecz kompatybilny z SSLv3 – zatem pośrednio również z SSLv2).

W kwietniu 2006 roku opublikowano oficjalną wersję TLS 1.1 (dokument RFC 4346 [17]). Wersja precyzowała pewne niejednoznaczności i dodawała nowe zalecenia wynikające z praktyki użycia.

W sierpniu 2008 IETF wydał dokument RFC 5246 [18], definiujący wersję 1.2 protokołu TLS. Jest to wersja obecnie rozwijana i zalecana przez IETF jako standard. W odniesieniu do wersji poprzedniej m.in. poszerzono zestaw wspieranych algorytmów kryptograficznych, wymuszono domyślne użycie silniejszych algorytmów w pewnych sytuacjach, a także rozwinięto możliwości negocjowania poszczególnych algorytmów.

W 2009 roku wykryto błąd w mechanizmie renegocjacji protokołu TLS, umożliwiający niezauważalne wstrzyknięcie danych napastnika do szyfrowanej komunikacji między klientem a serwerem [2]. Zaproponowano poprawkę do specyfikacji protokołu w postaci jego rozszerzenia, opisaną w dokumencie RFC 5746 [19].

Trwają obecnie prace nad wersją 1.3 protokołu TLS.

Zabezpieczeniom omawianych tu technologii poświęcone zostały już dwa raporty przygotowane przez Zespół Bezpieczeństwa PCSS („«Bezpieczna» e-bankowość" – raport nt. bezpieczeństwa szyfrowanej transmisji w połączeniach z polskimi portalami bankowości elektronicznej [9] oraz „Bezpieczeństwo sklepów internetowych – sesje i ciasteczka" – raport nt. bezpieczeństwa implementacji sesji HTTP w sklepach internetowych [10]), jednak żaden z nich nie dotykał testów ich implementacji w przeglądarkach internetowych, który to temat jest przedmiotem niniejszego opracowania. Badacze dokonali tu porównania podstawowych przeglądarek dostępnych na rynku pod kątem ich działania w specyficznych sytuacjach, w których mogą zostać postawione ich mechanizmy obsługi tuneli SSL/TLS. Dokładniej, początkowo planowano wykonać porównania:

a) ustawień standardowych przeglądarki internetowej w przypadku prób wymuszenia protokołów/algorytmów na określonym poziomie,
b) kosztu czasowego przeprowadzania operacji przesyłania dużej ilości informacji między klientem a serwerem,
c) działania tuneli dla poszczególnych implementacji w przypadku problemów związanych z zasobami systemu (głównie chodzi tu o obciążenie pamięci, procesora oraz konkretnej jakości połączenia sieciowego),
d) reakcji przy niestandardowym zachowaniu implementacji protokołu,
e) interakcji z użytkownikiem,
f) działań wykonywanych w tle (bez interakcji z użytkownikiem).

W związku z trudnymi do rozwiązania problemem rozróżnienia wpływu na rezultaty testów, wywieranego przez implementację samych tuneli od czynników będących pochodną obsługi innych elementów (funkcji) przeglądarki pod kątem wykorzystywania zasobów, ostatecznie autorzy raportu zrezygnowali z publikacji wyników dla podpunktu c).

Jak działa protokół SSL/TLS?

Jak wspomniano powyżej, kryptografia asymetryczna jest „droższa" od symetrycznej, ale nie wymaga uzgadniania tajnego klucza szyfrującego między stronami transmisji. Dlatego protokół SSL wykorzystuje do wstępnej fazy połączenia kryptografię asymetryczną (z kluczem publicznym), przy pomocy której uzgadniane są klucze symetryczne, stosowane później do kodowania danych. W ten sposób przeważająca część komunikacji odbywa się przy pomocy szybkich procedur szyfrujących kluczami tajnymi, które jednakże bezpiecznie przekazano przez sieć. Nawiązanie połączenia przy wykorzystaniu protokołu SSL rozpoczyna się trójfazową procedurą uzgadniania (tzw. *handshake*). Procedura ta rozpoczynana jest przez klienta, który zgłasza swoje żądanie do serwera.

Serwer uwierzytelnia się klientowi, przedstawiając swój certyfikat. Jest to krok obowiązkowy. Można również zażądać, aby dodatkowo klient wystawił swój certyfikat serwerowi. W skład certyfikatu wchodzi między innymi klucz publiczny serwera oraz jego pełna nazwa kwalifikowana (Full Qualified Domain Name – FQDN). Zawartość certyfikatu jest podpisana cyfrowo przy pomocy klucza prywatnego serwera. Oznacza to, że każdy, kto posiada klucz publiczny serwera (a więc także klient, który otrzyma go w certyfikacie), będzie mógł sprawdzić, czy certyfikat jest rzeczywiście okazany przez tę stronę transmisji, która go wysłała do klienta. Klient liczy funkcję skrótu z zawartości certyfikatu (informacje, jakiego algorytmu do obliczenia analogicznego skrótu użył serwer, także znajdują się w certyfikacie), przy pomocy klucza publicznego serwera odszyfrowuje skrót uzyskany po stronie serwera i porównuje te dwie wartości. Jeśli są one sobie równe, weryfikacja się powiodła.

Po weryfikacji certyfikatu strony transmisji uzgadniają klucze symetryczne i mogą już wymieniać dane.

1.2. Przyjęta metodyka testów

Niniejsze opracowanie ma na celu porównanie działania poszczególnych przeglądarek oraz zaimplementowanych w nich mechanizmów w kontekście standardowego ich użytkowania przez tzw. użytkownika masowego. Wobec tego podczas testów konieczny był wysoki stopień interakcji ze strony testerów – nie we wszystkich przypadkach zasadne i możliwe było symulowanie zachowania użytkownika poprzez automatyczne skrypty. Niemniej jednak podczas testów przyjęto kilka podstawowych zasad, które krótko scharakteryzowano poniżej:

a) wykonywanie testów przy pomocy przeglądarek zainstalowanych z możliwie standardowymi ustawieniami i bez jakichkolwiek ich modyfikacji (tzw. przeciętny użytkownik najczęściej nie przeprowadza takich działań – zwłaszcza w zakresie tak poważnych zmian ustawień, jak dotyczące wykorzystywanych algorytmów szyfrujących),

b) przeprowadzanie badań na możliwe aktualnych i uznanych za stabilne wersjach oprogramowania (zarówno w odniesieniu do przeglądarek, jak i serwerów WWW oraz innych aplikacji pomocniczych),

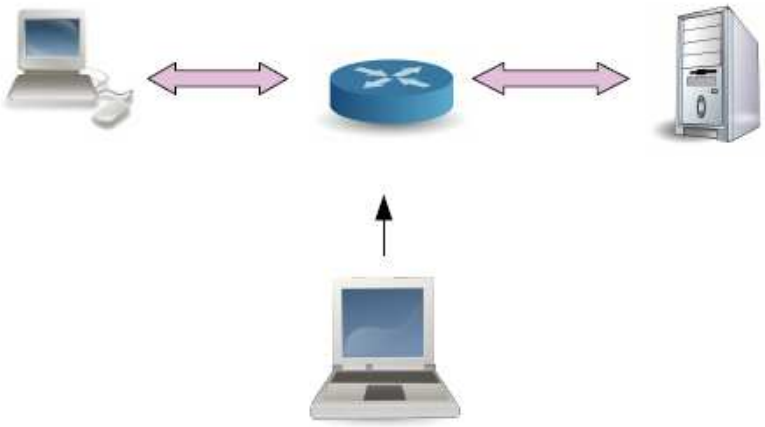
c) zapewnienie identycznego środowiska działania (parametry serwera, sieci i systemu klienckiego). Nie jest to w pełni poprawne, jeżeli celem jest uzyskanie całościowego obrazu problemu, gdyż część przeglądarek dedykowana była początkowo na zupełnie inne systemy i z innych implementacji się wywodzi. Kompletne wyniki wymagałyby przeprowadzenia testów wszystkich przeglądarek na kilku systemach operacyjnych, co niestety – z powodu braku stabilnej implementacji Internet Explorera na inne systemy niż z rodziny Windows – nie było możliwe,

d) przeprowadzanie testów z pominięciem dodatkowych możliwości systemów operacyjnych lub samych przeglądarek, a niemających istotnego związku z przedmiotem testów (mechanizmy typu *cache* itp.).

1.3. Środowisko testowe

W skład środowiska testowego wchodzą elementy przedstawione na Rysunku nr 3, a dokładniej:

a) kliencki komputer, na którym zainstalowana jest przeglądarka internetowa,
b) maszyna serwerowa, wyposażona w serwer WWW wraz z obsługą tuneli szyfrowanych oraz bibliotekami kryptograficznymi do obsługi certyfikatów,
c) system testowy, wykorzystywany do weryfikacji poprawności działania zdefiniowanych tuneli, mierzący czasy i rzeczywisty przepływ ruchu, wprowadzający zmiany w strumieniu pakietów oraz powodujący anomalie testowane w ramach niektórych punktów raportu,
d) urządzenie sieciowe, spinające całość w prostą i jednolitą sieć.



Rysunek 3. Środowisko testowe

1.4. Testowane przeglądarki

W trakcie testów wykorzystano następujące oprogramowanie:

a) przeglądarka internetowa Mozilla Firefox w wersji 3.6.8 [13],
b) przeglądarka internetowa Microsoft Internet Explorer w wersji 8.0.6001.18928 [20],
c) przeglądarka internetowa Opera w wersji 10.60 – kompilacja 3445 [22],
d) przeglądarka internetowa Google Chrome w wersji 5.0.375.55 [14],
e) przeglądarka internetowa Apple Safari w wersji 4.05 [12].

W przypadku ukazania się nowszej stabilnej wersji którejś z przeglądarek każdorazowo przeprowadzano powtórne testy z wykorzystaniem najnowszej implementacji, co w praktyce zdarzyło się w trakcie trwania prac kilkukrotnie. Testy zakończone zostały na początku września, możliwe zatem, że w chwili publikacji niniejszego raportu dostępne będą nowsze wersje niektórych aplikacji. Producenci aplikacji starają się nieustannie je ulepszać – autorzy raportu uznali zatem, że nie jest zasadne dalsze powtarzanie testów i odkładanie przygotowania dokumentu.

Należy także pamiętać, iż niektóre z przeglądarek posiadały już publicznie dostępne kolejne wersje beta. Szczególnej uwadze Czytelników polecana jest wersja beta przeglądarki Google Chrome (6.0.472.36 beta) dostępna dla systemów Linux, dla której w porównaniu do aktualnych wyników zostały wprowadzone bardzo pozytywne poprawki (interakcja z użytkownikiem oraz szybkość działania). W związku jednak z faktem, iż przeglądarka nie była jeszcze udostępniona w wersji finalnej (zwłaszcza dla systemu operacyjnego, na którym prowadzono opisane testy) i nie było pewności, czy zmiany zostaną wprowadzone – nie została wzięta pod uwagę w raporcie.

Warto również wspomnieć, iż do testów wykorzystywane były również serwer WWW Apache z modułem mod_ssl oraz biblioteka kryptograficzna OpenSSL.

2. Wyniki badań

2.1. Wymuszenie protokołu po stronie serwerowej

2.1.1 Opis testów i ich cel

Pierwszy z serii testów badających zaimplementowane funkcje obsługi tuneli SSL/TLS w przeglądarkach ma na celu wstępne rozpoznanie, czy i jakie tunele są w ogóle dla nich dostępne.

Wyniki tego testu w ogólności są bardzo łatwe do przewidzenia, jak pokazała jednak praktyka – czasami nie są zgodne z ogólnie przyjętym stanem wiedzy i spodziewanym stanem.

Głównym celem testów było wskazanie, czy którakolwiek z obecnie wypuszczanych przeglądarek przy standardowych ustawieniach i w praktyce (testy przez rzeczywiste nawiązywanie połączeń, a nie przez weryfikację ustawień) wykorzystuje protokoły w określonych wersjach.

Podatności możliwe do zidentyfikowania tylko i wyłącznie na podstawie tych testów, to weryfikacja obecności protokołu SSL w wersji 2 (SSLv2), który z powodu błędów projektowych i implementacyjnych podatny jest na szereg ataków. Najgroźniejsza z podatności skutkuje możliwością wykonania pełnego ataku bezwarunkowego *Man in the Middle* [1].

Warto w tym miejscu dodać, iż na tak zwany „warunkowy *Man in the Middle*" podatne są wszystkie wersje protokołów, jednak w miejscu, na które same protokoły nie mają żadnego wpływu – a konkretnie na etapie interakcji z użytkownikiem. Jeśli, mimo informacji o błędach ze strony przeglądarki, użytkownik zatwierdzi błędny i niezgodny certyfikat – będzie podatny na atak (!). W związku z tak dużym znaczeniem tej właśnie interakcji, poświęcono jej osobny zestaw testów, a co za tym idzie – także i rozdział w niniejszym opracowaniu (2.4).

Z atakiem **Man-in-the-Middle** (dosłownie z jęz. ang. „człowiek w środku") mamy do czynienia, gdy napastnik jest w stanie umiejscowić się pomiędzy ofiarą (klientem) a serwerem, z którym ona się kontaktuje. Przechwytuje komunikację w obu kierunkach i w każdym przypadku udaje drugą stronę transmisji. Zarówno klient, jak i serwer są przeświadczeni, że kontaktują się z właściwym odbiorcą – jednak przesyłają swoje dane napastnikowi, który może je zmodyfikować niepostrzeżenie dla komunikujących się stron.

Odpowiednie wykorzystanie implementacji infrastruktury klucza publicznego (np. certyfikatów X.509 poświadczanych przez zaufane centra certyfikacji) w idealnym przypadku zabezpieczy przed tym rodzajem ataku. Niestety, zarówno protokoły, jak ich implementacje mogą zawierać błędy. Także użytkownik, nie czytając komunikatu przeglądarki z ostrzeżeniem o nieprawidłowym certyfikacie i klikając „Zatwierdź" lub „Zezwól" może nieświadomie narazić się na opisywane niebezpieczeństwo.

Innym aspektem w przypadku „dobrych" (w rozumieniu bezpieczeństwa) protokołów, czyli SSL w wersji 3 (SSLv3) i TLS w wersji 1 (TLSv1), jest fakt zaimplementowania ich obsługi w konkretnych przeglądarkach. Jak można zobaczyć poniżej – fakt ich pełnej implementacji nie jest tak oczywisty, jak mogłoby się to wydawać! z uzyskanych rezultatów wynika, iż niektóre z przeglądarek ewidentnie preferują połączenia przy użyciu SSLv3, a inne przeciwnie, wolą dużo bardziej wykorzystywać szyfrowane połączenia przy wykorzystaniu protokołu TLSv1. Jak wspomniano już we wstępie – protokoły te są bardzo do siebie podobne (choć nie w pełni kompatybilne), więc fakt rozbijania ich obsługi na poszczególne implementacje wydaje się dość zaskakujący.

W ogólności oczywiście nie jest to problemem, jednak zespół testowy przygotowujący niniejszy raport doszedł do wniosku, że możliwe jest zaklasyfikowanie i tego faktu do elementów bezpieczeństwa. Łączy się to oczywiście z aspektem dostępności. W tym celu przeprowadzone zostały dodatkowe testy, mające na celu zweryfikowanie, czy w Internecie także i administratorzy serwerów webowych mają swoje szczególne preferencje co do konfiguracji bezpiecznych tuneli w kontekście wyboru między SSLv3 oraz TLSv1. Pobocznie przeprowadzono także dodatkową weryfikację dostępności na serwerach protokołu SSLv2, co jednak nie jest przedmiotem niniejszego raportu, w związku z czym nie będzie w ramach tego dokumentu szerzej analizowane. Metodyka tychże testów dodatkowych jest bardzo prosta i sprowadza się do testów rzeczywistych połączeń SSL/TLS dla wybranej grupy serwisów. Dobrym wyznacznikiem popularności stron w Internecie są portale prowadzące rankingi. Jednym z nich jest portal Alexa [11], który udostępnia *online* bogatą listę jednego miliona najpopularniejszych stron internetowych na świecie [7].

Dla stron, których adresy pobrano z podanego wyżej portalu, dokonano sprawdzenia, czy mają one swoje odpowiedniki zabezpieczone przez protokoły SSL/TLS, oraz – dla rozpoznanych stron tego rodzaju – przeprowadzono weryfikację wersji tych protokołów (w kontekście możliwości ich wynegocjowania przez klienta). Badania takie przeprowadzono również osobno dla listy polskich witryn zawartych w powyższym podzbiorze – ich liczba była równa 445.

2.1.2 Wyniki oraz wnioski

Poniższa tabela przedstawia ogólny rozkład standardowo udostępnionych protokołów dla poszczególnych przeglądarek. Całość testu pokazała empirycznie, iż nie wszystkie przeglądarki zestawiały połączenia przy zadanym na serwerze protokole (w przypadku Safari wynik testu okazał się niezgodny z konfiguracją przeglądarki!). Bardzo pozytywnym (choć w kontekście dzisiejszego stanu wiedzy na temat zagrożeń bezpieczeństwa IT – spodziewanym) rezultatem jest fakt, iż przy standardowych ustawieniach wszystkich analizowanych przeglądarek zablokowana jest możliwość zestawiania połączeń za pomocą protokołu SSLv2. Co więcej, w osobnych testach stwierdzono, iż w większości także poprzednie wersje testowanych przeglądarek miały obsługę tego protokołu

wyłączoną, co dobrze świadczy o świadomości osób odpowiedzialnych za standardową konfigurację.

Dokładny rozkład protokołów udostępnionych domyślnie dla poszczególnych przeglądarek widoczny jest w Tabeli nr 1 (X = nie udostępniono domyślnie, V = udostępniono domyślnie).

Tabela 1. Standardowo udostępnione wersje protokołów SSL/TLS w przeglądarkach

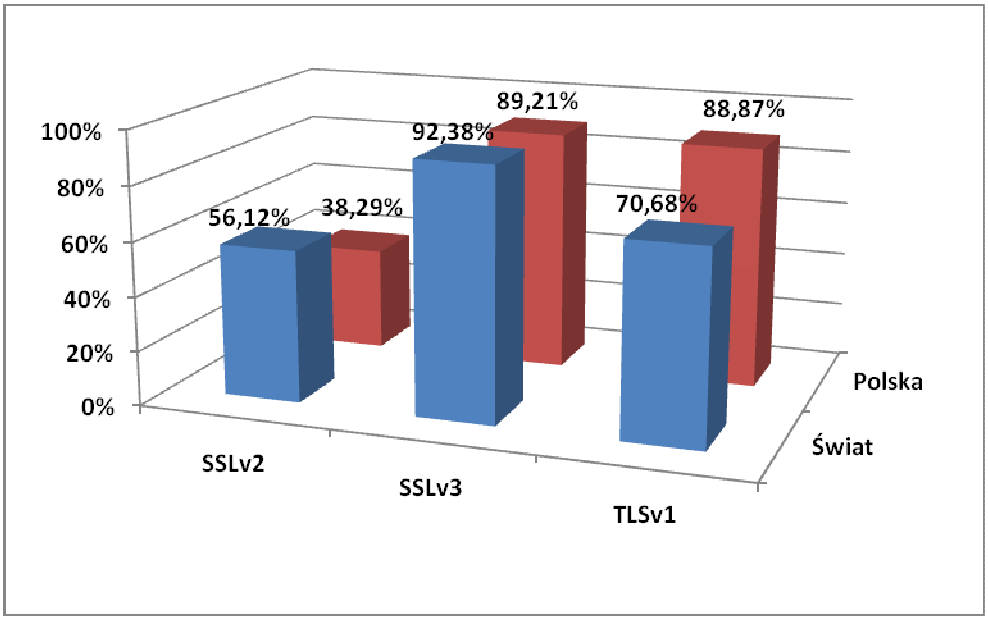
	SSLv2	SSLv3	TLSv1
Firefox	X	V	X
Internet Explorer	X	V	V
Opera	X	V	X
Google Chrome	X	V	V
Safari	X	X	V

Poniższa tabela (Tabela nr 2) przedstawia rozkład występowania możliwości połączenia szyfrowanego do tych spośród miliona najpopularniejszych stron internetowych na świecie według portalu Alexa [11], z którymi udało się nawiązać poprawne połączenie szyfrowane. Wyniki przedstawiono zarówno w ujęciu ogólnym (światowym), jak i tylko dla domen zarejestrowanych w naszym kraju (*.pl). W kolumnie „SSL/TLS" przedstawiono liczbę stron, z którymi udało nawiązać się szyfrowane połączenie, zakończone pobraniem certyfikatu. od których udało uzyskać się. W kolejnych kolumnach pokazano także ilościowy i procentowy (względem liczby stron wystawiających certyfikat oraz wszystkich stron) rozkład protokołów, przy których pomocy udało się nawiązać poprawne połączenie.

Tabela 2. Możliwości połączeń szyfrowanych do stron WWW

	Stron	**SSL/TLS**	**SSLv2**	**SSLv3**	**TLSv1**
Strony światowe	1 000 000	231 437 23,14% 100,00%	129 893 56,12% 12,99%	213 802 92,38% 21,38%	140 662 60,78% 14,07%
Strony polskie	9 087	4 670 51,39% 100,00%	1 788 38,29% 19,68%	4 166 89,21% 45,85%	4 150 88,87% 45,67%

Z powyższej tabeli dokładnie widać, iż tylko nieco ponad 23% z miliona najpopularniejszych stron internetowych na świecie oraz ponad 51% najpopularniejszych polskich witryn daje możliwość poprawnego połączenia przy użyciu szyfrowanego tunelu oraz pobrania certyfikatu – należy naturalnie pamiętać, że nie wszystkie popularne serwisy internetowe wymagają takich zabezpieczeń (np. serwisy informacyjne) – dlatego też autorzy raportu nie chcieliby wyciągać w tym zakresie wiążących wniosków, choć naturalnie cieszy rezultat uzyskany dla stron w domenie .pl. Biorąc pod uwagę jedynie serwisy, od których udało się uzyskać certyfikat tożsamości, rozkład wyników jest jakościowo podobny, ale dość znacząco różni się ilościowo (por. Wykres 1).



Wykres 1. Możliwości nawiązania szyfrowanych połączeń z popularnymi witrynami WWW

Wykres obrazuje, że zarówno w odniesieniu do stron polskich, jak i ogółu analizowanych witryn, najmniej serwerów oferuje możliwość połączenia przy pomocy nierekomendowanego dłużej protokołu SSL w wersji 2, a najwięcej – przy użyciu protokołu SSL w wersji 3. Wyraźnie jednak widać, że polskie witryny oferują szyfrowane tunele SSLv2 rzadziej niż wynosi przeciętna światowa. Także w przypadku polskich serwisów stopień wykorzystania najnowszego protokołu – TLS w wersji 1.x – zbliżył się już niemal do prezentowanego dla SSLv3. Pozwala to wysnuć wniosek, że twórcy polskich serwisów, oferujących połączenia szyfrowane, zdają się charakteryzować nieco lepszą wiedzą w tym konkretnym zakresie niż wynosi przeciętna światowa.

UWAGA: w trakcie testów nie weryfikowano, czy treść na stronie dla HTTP i HTTPS jest identyczna. Może tu zatem nastąpić pewnego rodzaju przekłamanie w wynikach, które jednak z uwagi na charakterystykę testów (badających samą możliwość połączenia przy pomocy konkretnych wersji protokołów kryptograficznych) można z całą pewnością pominąć.

2.2. Wymuszenie algorytmu szyfrowania po stronie serwera

2.2.1 Opis testów i ich cel

Naturalną wręcz kontynuacją testów z poprzedniego rozdziału stało się dla osób przygotowujących raport przeprowadzenie weryfikacji, które zestawy kryptograficzne i w których protokołach były najbardziej popularne (domyślne) dla testowanych implementacji w przeglądarkach internetowych.

Przyjęto tu podział algorytmów dla poszczególnych protokołów, znany z pakietu OpenSSL [21]:

a) High,
b) Medium,

c) Low,
d) Exp.

Nazwa pierwszych trzech kategorii dość dobrze charakteryzuje poziom bezpieczeństwa, jaki może być przypisany do poszczególnych zestawów. Ostatnia z grup oznacza tzw. szyfry eksportowe, wykorzystywane wcześniej ze względu na obowiązujące w Stanach Zjednoczonych ograniczenia dotyczące eksportu technologii kryptograficznych. Większość z tych ograniczeń obecnie zniesiono, a samych zestawów eksportowych – znacznie słabszych (charakteryzujących się mniejszymi długościami klucza stosowanego w odniesieniu do konkretnego algorytmu) – wykorzystywać nie należy.

Dla wyjaśnienia i dokładniejszej informacji, w poniższej tabelce pokazano algorytmy kryptograficzne, które wykorzystywane są w każdym przypadku dla poszczególnych protokołów.

Tabela 3. Algorytmy kryptograficzne składające się na poszczególne zestawy

	SSLv2	**SSLv3**	**TLSv1**
High	DES-CBC3-MD5 RC2-CBC-MD5 RC4-MD5 DES-CBC-MD5 EXP-RC2-CBC-MD5 EXP-RC4-MD5	DHE-RSA-AES256-SHA AES256-SHA EDH-RSA-DES-CBC3-SHA DES-CBC3-SHA DHE-RSA-AES128-SHA AES128-SHA	DHE-RSA-AES256-SHA AES256-SHA EDH-RSA-DES-CBC3-SHA DES-CBC3-SHA DHE-RSA-AES128-SHA AES128-SHA
Medium		RC4-SHA RC4-MD5	RC4-SHA RC4-MD5
Low		EDH-RSA-DES-CBC-SHA DES-CBC-SHA	EDH-RSA-DES-CBC-SHA DES-CBC-SHA
Exp		EXP-EDH-RSA-DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC2-CBC-MD5 EXP-RC4-MD5	EXP-EDH-RSA-DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC2-CBC-MD5 EXP-RC4-MD5

Jak widać z powyższej tabelki, w protokołach dostępne są na różnych poziomach algorytmy, które ogólnie zostały już uznane za podatne lub osłabione na tyle, aby mogły (przynajmniej teoretycznie) zostać złamane w rozsądnym czasie – np. algorytm RC4 [4].

Dodać tu jednak należy, iż wszystkie podatności poszczególnych systemów kryptograficznych nie mogą stanowić podstawy do ich usunięcia z zestawów implementowanych przy protokołach SSL/TLS. Omawiany wyżej algorytm RC4, mimo swoich podatności, posiada jedną bardzo istotną zaletę – jest bardzo szybkim algorytmem, dzięki czemu nawet przy wykorzystaniu kluczy 40-bitowych i wiedząc, że jego złamanie może zająć określony czas, nadal można wykorzystywać go do przesyłania dużej ilości danych o niskim poziomie poufności, dla których wiadomo, że czas

wymaganej dla nich poufności (w ogólności czas ich życia) jest krótszy od minimalnego czasu złamania klucza.

Testy były wykonywane półautomatycznie przez wymuszenie po stronie serwerowej jedynie zadanego protokołu oraz zadanej klasy systemów kryptograficznych.

Celem testów było ukazanie klas algorytmów kryptograficznych dostępnych dla każdej z testowanych przeglądarek oraz sprawdzenie, który z szyfrów jest preferowany dla każdej z nich, aby osoby czytające raport miały świadomość swojego poziomu bezpieczeństwa przy wykorzystaniu każdej z przeglądarek.

2.2.2 Wyniki oraz wnioski

Poniższa tabelka przestawia zestawy kryptograficzne wraz z konkretnymi algorytmami, które zostały wynegocjowane przez poszczególne przeglądarki internetowe. Oczywistym i pozytywnym wynikiem, który można było przewidzieć na podstawie rezultatów poprzedniego testu, jest brak możliwości jakiegokolwiek połączenia z serwerem, gdy ten wykorzystuje protokół SSLv2. Jeśli chodzi o protokoły SSLv3 i TLSv1, kwestia wygląda już na dużo bardziej skomplikowaną. Wielokrotne testy wykazały, iż producenci przeglądarek internetowych także wykorzystują przyjętą do testów przez autorów systematykę i wykonują standaryzację domyślnych ustawień względem niej. Wyróżniającymi się przeglądarkami są Safari, Google Chrome i Mozilla Firefox, których producenci założyli zgodnie, iż algorytmy będące w grupach LOW i EXP nie są na tyle zaufane pod względem bezpieczeństwa, aby zezwalać na połączenia przy ich pomocy do serwera. Zachowanie to zasługuje tu na znaczącą pochwałę z racji faktu, iż tego typu konfiguracja jest aktualnie zaleceniem na poziomie „dobrych praktyk" w zarządzaniu tunelami SSL/TLS.

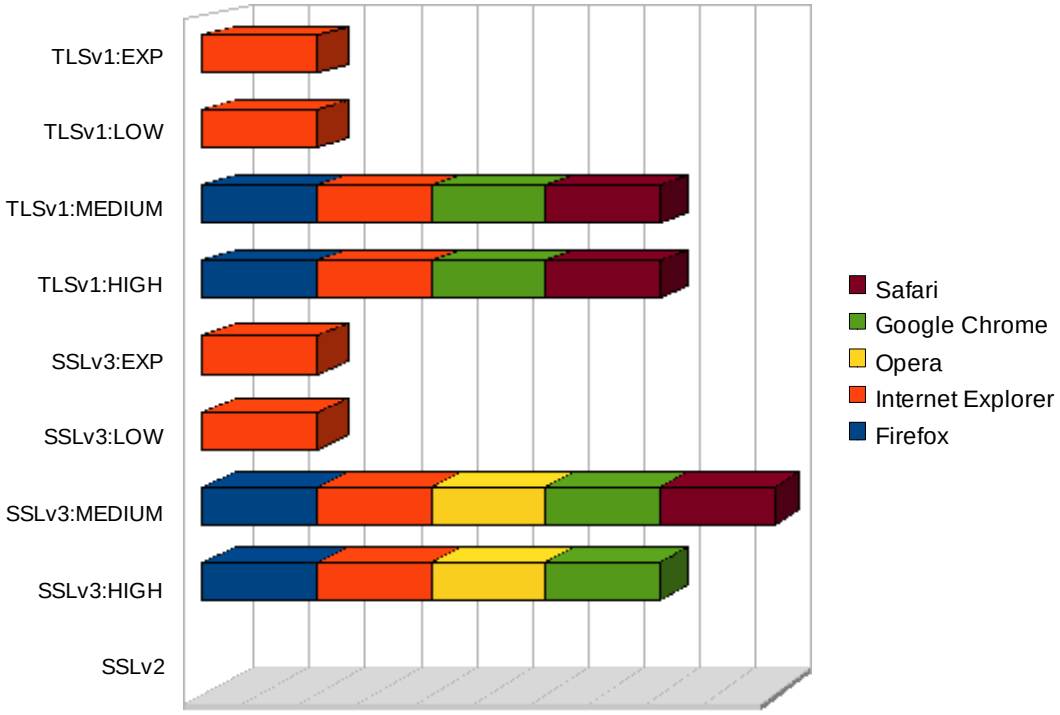
W poniższej tabeli X oznacza brak możliwości połączenia, V – możliwość połączenia bez wskazania konkretnego algorytmu.

Tabela 4. Możliwości negocjacji algorytmów kryptograficznych

	Firefox	Internet Explorer	Opera	Google Chrome	Safari
SSLv2	X	X	X	X	X
SSLv3:HIGH	AES-256	V	AES-256	EDH-RSA-DES-CBC3-SHA-168	X
SSLv3:MEDIUM	RC4-128	V	RC4-128	RC4-128	V
SSLv3:LOW	X	V	X	X	X
SSLv3:EXP	X	V	X	X	X
TLSv1:HIGH	AES-256	V	X	DHE-RSA-AES128-SHA	V
TLSv1:MEDIUM	RC4-128	V	X	RC4-128	V
TLSv1:LOW	X	V	X	X	X
TLSv1:EXP	X	V	X	X	X

Inną ciekawostką, którą można tu zauważyć, jest fakt zgody producenta przeglądarki Internet Firefox na połączenie niezależnie od poziomu grupy algorytmów kryptograficznych, pozostawiając tu niejako decyzję w gestii administratora przygotowującego konfigurację serwera względem konkretnej aplikacji webowej.

Dla lepszego zobrazowania wyników, przedstawiono je także na wykresie poniżej.



Wykres 2. Możliwości połączeń szyfrowanych do stron WWW

2.3. Testy czasowe obsługi protokołu SSL/TLS przy wybranych algorytmach szyfrowania

2.3.1 Opis testów i ich cel

Poniższy zestaw testów pokazuje różnice w implementacjach mechanizmów wykorzystywanych przez przeglądarki do przesyłania danych w tunelach SSL/TLS. Wykonane testy są porównaniem przypadków wykorzystania tuneli w różnej konfiguracji (jeden lub kilka tuneli z dużymi lub rozproszonymi porcjami danych) oraz w różnych kierunkach – testowano zarówno pobieranie (*download*), jak i wysyłanie (*upload*) danych.

Porównanie ma też za zadanie pokazać, jak przeglądarki zareagują w przypadku bardzo dużej ilości przesyłanych danych (zrywane tunele SSL), a także przy dużej liczbie jednocześnie obsługiwanych tuneli – i jaki ma to wpływ na jej ogólne działanie. Można w tym przypadku mówić

o ataku na dostępność przeglądarki, ale i na integralność danych (brak informacji o niepełnym przesłaniu pliku przy zerwaniu tunelu).

Metodologie testów zostały dostosowane do poszczególnych funkcji, które były przy testach wykorzystywane. I tak – do testowania możliwości pobierania przez przeglądarkę dużych plików stworzony został specjalny skrypt, udostępniający po interfejsie webowym zadaną ilość danych, tak aby były one w całości przesłane przez testowany tunel SSL/TLS. W przypadku przesyłania danych wieloma tunelami wykorzystano mechanizmy wirtualizacji (na różnych poziomach), aby wygenerować stronę, na której znajdywało się dużo elementów udostępnianych z zewnętrznych domen (i jednocześnie z zewnętrznym certyfikatem), a to w celu utworzenia grupy niezależnych od siebie tuneli o znanych z góry parametrach.

W przypadku trybu *upload* zbudowano dwa skrypty w języku PHP do wysyłania dużego pliku o zadanej wielkości przez tunel o z góry znanych parametrach, a także skrypty napisane w językach PHP i JavaScript, wykorzystywane do generowania jednoczesnego przesyłu danych kilkoma tunelami w stronę serwera.

Badania parametrów czasowych wykonywane były możliwie dokładnie (czas badany na podstawie dokładnego momentu przesłania pakietów względem siebie), co pozwala stosunkowo dokładnie ustalić różnice w prędkości działania przeglądarek względem siebie, jednak nie determinuje, iż różnice te wywołane są jedynie przez różne implementacje tuneli SSL/TLS.

Nadmienić tu także należy, iż – aby nie zaciemniać obrazu wyników – nie były wykorzystywane żadne mechanizmy przyspieszania lub *cache'owania* danych, które mogłyby spowodować niepożądane zaburzenia rezultatów.

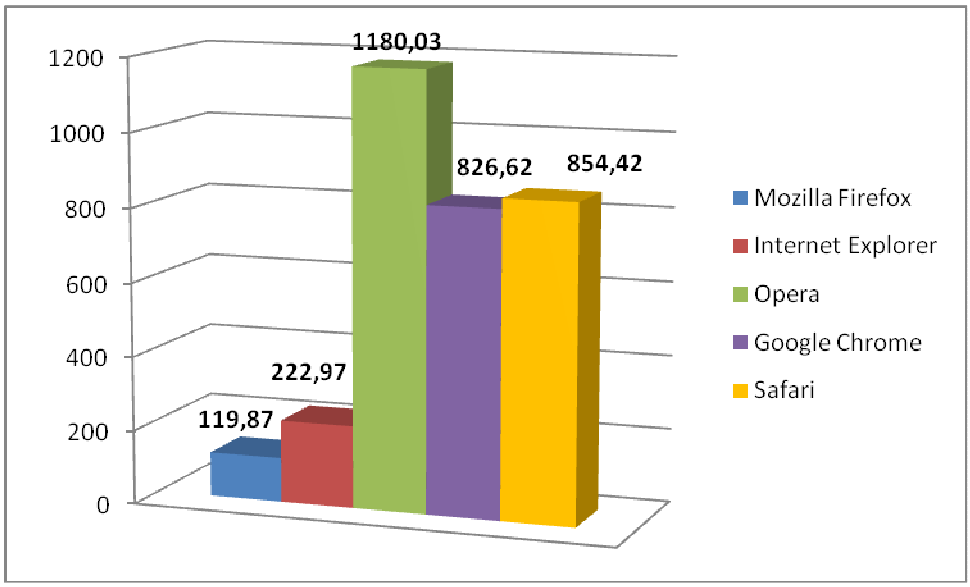
2.3.2 Wyniki oraz wnioski

2.3.2.1 Przesyłanie dużej ilości danych do serwera przy użyciu małej liczby połączeń

Poniżej przedstawiono zestawienie rezultatów osiągniętych dla poszczególnych przeglądarek podczas przesyłania dużej ilości danych (pojedynczy plik o losowej zawartości i wielkości 256 MB) od klienta do serwera przy użyciu pojedynczego szyfrowanego tunelu.

Tabela 5. Porównanie czasu przesyłania dużej ilości danych pojedynczym tunelem w kierunku klient → serwer

	Mozilla Firefox	Internet Explorer	Opera	Google Chrome	Safari
Czas (s)	119,87	222,97	1180,03	826,62	854,42



Wykres 3. Porównanie czasu przesyłania dużej ilości danych pojedynczym tunelem w kierunku klient → serwer

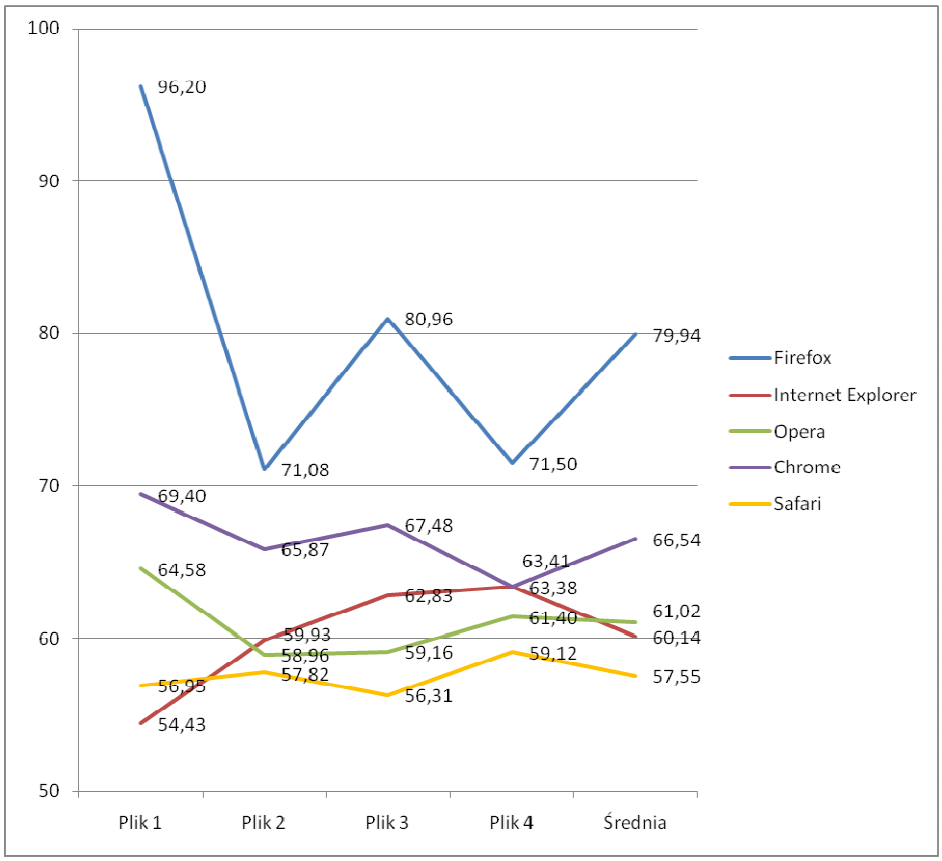
Z uzyskanych rezultatów wynika, że dwie przeglądarki (Mozilla Firefox, a w drugiej kolejności Internet Explorer) były w stanie obsłużyć operację transferu danych najszybciej. O klasę gorzej zaprezentowały się pozostałe aplikacje – w szczególności Opera. Różnica między najszybszą a najwolniejszą przeglądarką osiągnęła w tym wypadku rząd wielkości, co wydaje się już znaczącym błędem jakościowym.

2.3.2.2 Przesyłanie dużej ilości danych od serwera przy użyciu małej liczby połączeń

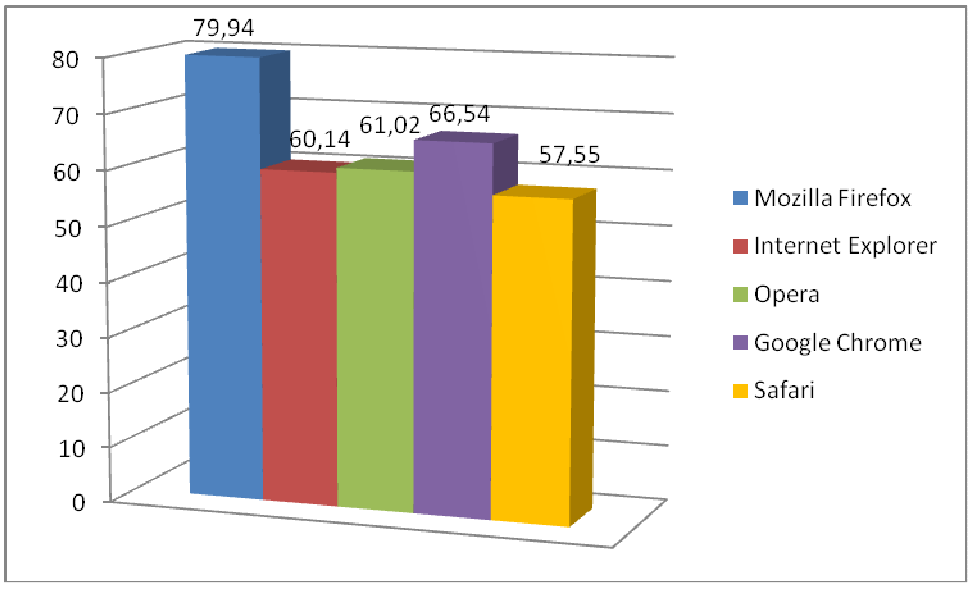
Przedmiotem kolejnego testu było również wysyłanie dużej ilości danych przy pomocy pojedynczego szyfrowanego tunelu, ale w kierunku od serwera do klienta. W tym przypadku – z uwagi na specyfikę procesu pobierania danych (który użytkownicy uruchamiają znacznie częściej niż wysyłanie, szczególnie w odniesieniu do dużych plików) – zdecydowano się na wykorzystanie 4 plików tej samej wielkości (512 MB), które pobierano sekwencyjnie (pojedynczo), i zaprezentowanie wyników szczegółowych oraz uśrednionych. Poniższa tabela i wykresy przedstawiają zestawienie osiągniętych rezultatów dla poszczególnych przeglądarek.

Tabela 6. Porównanie czasu przesyłania dużej ilości danych pojedynczym tunelem w kierunku serwer → klient (wyniki pojedyncze i uśrednione)

	Firefox	Internet Explorer	Opera	Chrome	Safari
Plik 1	96,20	54,43	64,58	69,40	56,95
Plik 2	71,08	59,93	58,96	65,87	57,82
Plik 3	80,96	62,83	59,16	67,48	56,31
Plik 4	71,50	63,38	61,40	63,41	59,12
Średnia	79,94	60,14	61,02	66,54	57,55



Wykres 4. Porównanie czasu przesyłania dużej ilości danych pojedynczym tunelem w kierunku serwer → klient



Wykres 5. Porównanie czasu przesyłania dużej ilości danych pojedynczym tunelem w kierunku serwer → klient (tylko wyniki uśrednione)

Tym razem, co ciekawe, osiągnięte wyniki okazały się zupełnie odmienne od poprzednich. Dla kierunku od serwera do klienta (a więc – jak się wydaje – znacznie częściej wykorzystywanym) wyniki okazały się dużo bardziej zbliżone (różnica między najlepszą a najgorszą przeglądarką wynosiła tylko ok. 40%). Największą wydajnością obsługi dużych wolumenów danych charakteryzowała się przeglądarka Safari, ale tylko bardzo niewiele ustąpiły jej Internet Explorer oraz Opera. Wyraźnie najwolniejszą przeglądarką okazał się tym razem Firefox.

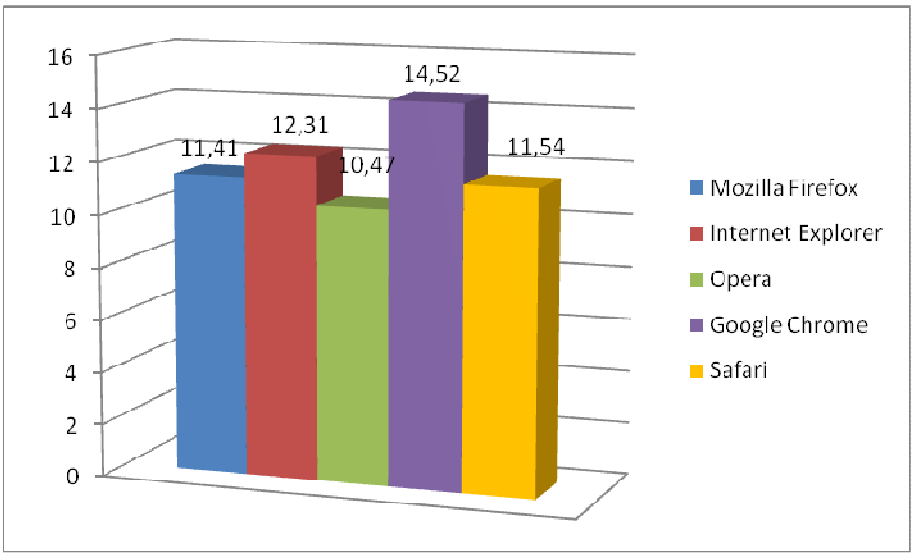
Wyniki tego testu mogą sugerować, że producenci przeglądarek starają się zwracać znacznie bardziej wytężoną uwagę na aspekty, które są ważne dla przeciętnego użytkownika (tj. jak największej liczby użytkowników) – co jest całkowicie uzasadnione z marketingowego punktu widzenia. Mogą oczywiście wystąpić sytuacje, w których poszczególni pojedynczy użytkownicy wykazują specyficzne preferencje w kierunku innego konkretnego kryterium i na jego podstawie wykazane różnice okażą się znaczące.

2.3.2.3 Przesyłanie małej ilości danych do serwera przy użyciu dużej liczby połączeń

Następne testy koncentrowały się na przesyłaniu wielu niedużych porcji danych w kierunku od klienta do serwera przy wykorzystaniu dużej liczby równolegle nawiązywanych połączeń w szyfrowanych tunelach. Przesyłanych było tu dziesięć plików graficznych o wielkości około 2MB (ponieważ pliki były rzeczywistymi grafikami, rozmiar plików nie był dokładną potęgą liczby 10 czy też 2). Rezultaty zamieszczono w poniższej tabeli oraz na wykresie.

Tabela 7. Porównanie czasu przesyłania małej ilości danych wieloma równoległymi tunelami w kierunku klient → serwer

	Mozilla Firefox	Internet Explorer	Opera	Google Chrome	Safari
Czas (s)	11,41	12,31	10,47	14,52	11,54



Wykres 6. Porównanie czasu przesyłania małej ilości danych wieloma równoległymi tunelami w kierunku klient → serwer

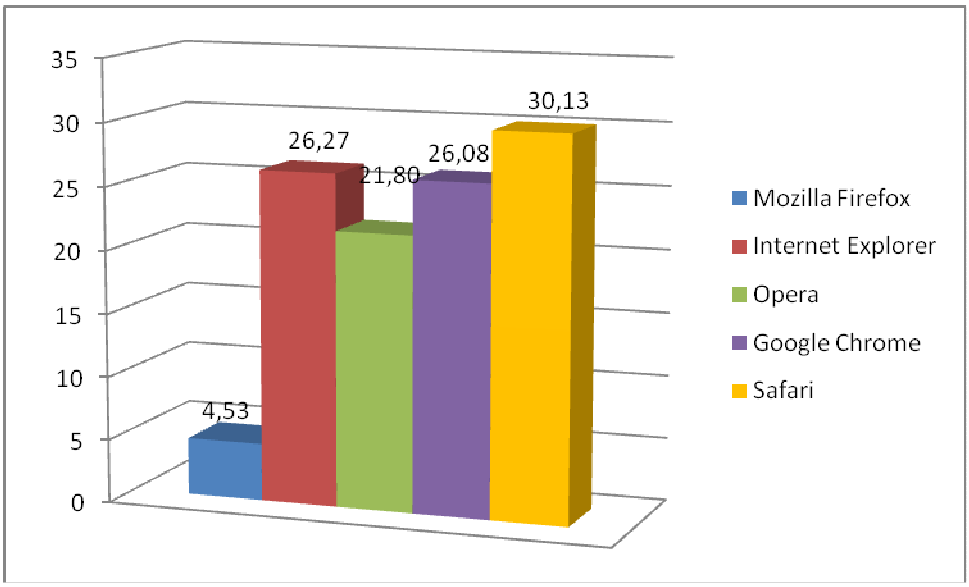
Wyniki nie były tym razem tak zróżnicowane, jak w przypadku przesyłania na serwer dużych ilości danych. Tym razem najszybciej z zadaniem poradziła sobie Opera (która, najgorzej wypadła w teście dla pojedynczego dużego pliku), niewiele wolniej Mozilla Firefox, a następnie Safari. Najwolniejszą przeglądarką w tym teście okazała się aplikacja Google Chrome. Rezultaty są jednak w tym przypadku na tyle podobne, że trudno mówić o jakichkolwiek wskazaniach jakościowych.

2.3.2.4 Przesyłanie małej ilości danych od serwera przy użyciu dużej liczby połączeń

Kolejna grupa testów – analogicznie do poprzedniej – badała zachowanie przeglądarek dla przesyłania wielu niedużych porcji danych przy wykorzystaniu dużej liczby równolegle nawiązywanych połączeń w szyfrowanych tunelach, ale tym razem w kierunku od serwera do klienta (dla celów testowych obrano 10 plików graficznych wielkości 2 MB każdy). Poniższe: tabela i wykres obrazują osiągnięte rezultaty.

Tabela 8. Porównanie czasu przesyłania małej ilości danych wieloma równoległymi tunelami w kierunku serwer → klient

	Mozilla Firefox	Internet Explorer	Opera	Google Chrome	Safari
Czas (s)	4,53	26,27	21,80	26,08	30,13



Wykres 7. Porównanie czasu przesyłania małej ilości danych wieloma równoległymi tunelami w kierunku serwer → klient

Ponownie wyniki dla przeciwnego kierunku transmisji są zupełnie odmienne (por. testy przeprowadzone w podrozdziałach 2.3.2.1 oraz 2.3.2.2). Wyraźnie najszybszą przeglądarką okazał się Firefox. Pozostałe przeglądarki były wyraźnie wolniejsze (niemal o rząd wielkości) – spośród nich względnie najszybsza była Opera, następnie – niemal na tym samym poziomie – Google Chrome oraz Microsoft Internet Explorer, wreszcie najwolniejsze Safari).

Podsumowując powyższe 4 grupy testów, należy stwierdzić, że optymalizacja obsługi transferu danych w poszczególnych przeglądarkach jest bardzo zróżnicowana – przykładowo Mozilla Firefox wygrywa 2 testy, a przegrywa 1. Trudno zatem wskazać jednoznacznego lidera w tym obszarze. Z punktu widzenia użytkownika często pobierającego spore wolumeny danych z zabezpieczonych witryn, względnie dobrym wyborem mógłby być Firefox (zwycięzca testu dla wielu mniejszych plików przy wprawdzie najgorszym wyniku testu dla pojedynczego dużego pliku – ale z małym zróżnicowaniem rezultatów).

2.4. Reakcja w przypadku niestandardowego zachowania oraz interakcja z użytkownikiem

2.4.1 Opis testów oraz ich cel

Jednym z podstawowych testów przeprowadzanych dla przeglądarek internetowych w kontekście ich bezpieczeństwa jest sposób interakcji z użytkownikiem. Kwestia ta jest szczególnie istotna, jeśli zważyć, że z usług internetowych korzystają dziś nie tylko specjaliści IT. Dlatego tak ważne są intuicyjne komunikaty, konfiguracja odpowiednich wartości domyślnych, wręcz zdolność do prowadzenia użytkownika w pewnym stopniu „za rękę" w kierunku decyzji pozwalającej uchronić się przed atakiem. Z drugiej strony – mechanizmy tego rodzaju nie mogą użytkownika irytować, ponieważ prawdopodobnie będzie próbował je wyłączyć.

W większości przeprowadzonych testów badano, w jaki sposób przeglądarka sygnalizuje problematyczne elementy, które są przez nią wyświetlane lub choćby interpretowane. W przypadku wykorzystywania tuneli SSL/TLS można sobie wyobrazić kilka problemów, jednak najdotkliwszymi są: przesyłanie danych w słabych tunelach (SSLv2, brak szyfrowania), w tunelach, które mają niepoprawny certyfikat, przesyłanie tylko części danych przez kanał zabezpieczony certyfikatem lub błędy samych połączeń.

Bardzo ważnym aspektem w przypadku mechanizmów bezpieczeństwa opartych na certyfikatach jest analiza poprawności certyfikatu (np. jego aktualności, siły klucza, renomy centrum wystawiającego certyfikat, faktu ewentualnego unieważnienia, a przede wszystkim – czy strona, która go wykorzystuje, jest faktycznie tą, z którą użytkownik zamierzał się połączyć).

W związku z faktem, iż z powodu błędu użytkownika (zwłaszcza niemającego wiedzy dotyczącej zagrożeń internetowych), np. automatycznego kliknięcia i zatwierdzenia w ten sposób komunikatu z ostrzeżeniem, bardzo prosto przeprowadzić można atak typu Man In The Middle (MitM, [3]), poziom dbałości o interakcję między użytkownikiem a przeglądarką był przedmiotem szczególnie wielu testów. W niniejszym rozdziale przedstawione zostaną podstawowe aspekty ww. interakcji, czyli:

- fakt, czy błędy (komunikaty, informacje) są przesyłane do użytkownika, czy obsługiwane „w tle",

- sposoby, na jakie informacje o błędach prezentowane są przez przeglądarkę,
- standardowe ustawienia, pozwalające na zapamiętywanie „niebezpiecznych" ustawień,
- możliwości (intencjonalne lub nie) zmiany krytycznych ustawień dla stron/domen,

Bardzo ważna w tym miejscu jest również możliwość łatwej weryfikacji możliwych ustawień i komunikatów wykorzystywanej przez czytelnika przeglądarki. Poniżej przedstawiono kolejne kroki dla standardowych interfejsów poszczególnych przeglądarek wraz z wyświetlanymi przez nie komunikatami dla zobrazowania sposobu obsługi opisanej problematyki.

2.4.2 Komunikaty wyświetlane przez przeglądarki

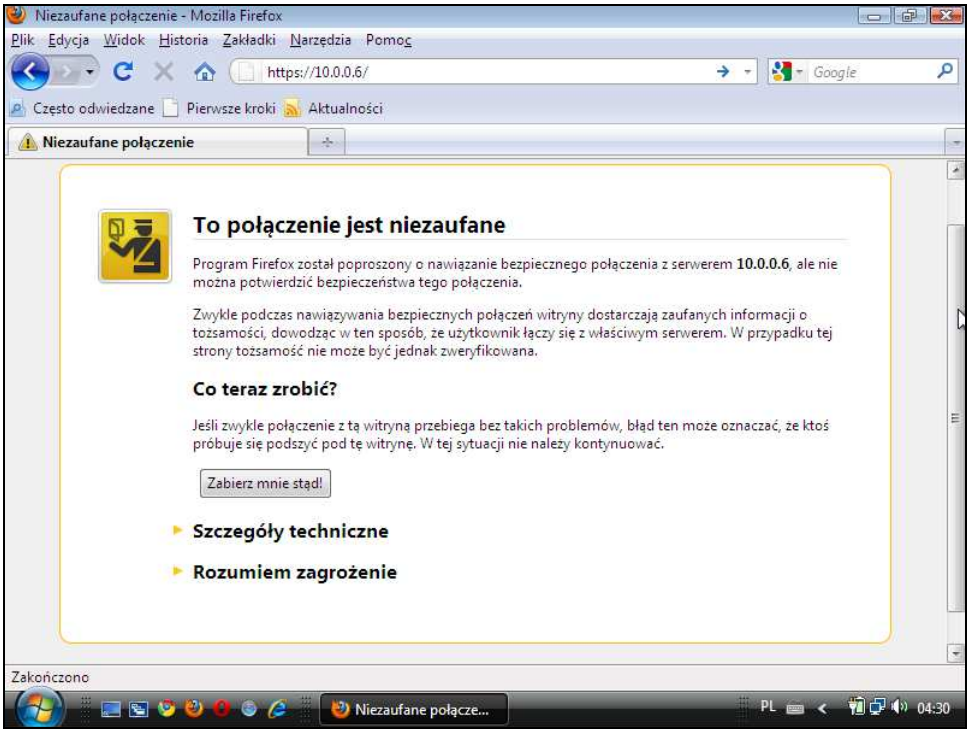
Podstawowym elementem, na który audytorzy chcieli zwrócić uwagę, jest podejście producentów przeglądarek do prezentowania zagrożeń użytkownikowi. Abstrahując od typów błędów, które zostaną opisane w dalszej części opracowania, bardzo ważnym jest czytelne i jasne zasygnalizowanie zagrożenia, płynącego z napotkanych sytuacji czy też poszczególnych operacji wykonywanych przez użytkownika. Pierwszą istotną kwestią, jaką można postawić w przypadku tego typu testów, jest fakt, iż producenci nie powinni pozwalać sobie w tak krytycznych miejscach jak informacje o błędnych certyfikatach na akceptację przez użytkownika sytuacji, która spowodowała błąd (i tym samym – narażenie się na atak) przy pomocy pojedynczego kliknięcia.

Mimo, że świadomość zagrożeń internetowych nieustannie rośnie, niedoświadczeni użytkownicy nadal mogą dać się „nabrać" na podstawione (fałszywe) strony logowania, „przyjazne" linki przesyłane przy pomocy poczty elektronicznej czy informacje o wygranych nagrodach. Ci właśnie użytkownicy, widząc komunikat wyświetlający się bezpośrednio przed treścią przez nich pożądaną, często starają się – bez czytania – możliwie szybko go kliknąć, aby przejść dalej. Według autorów raportu jest to spowodowane błędnymi próbami ułatwienia użytkownikowi pracy przy pomocy systemów informatycznych (np. proces instalacji przez klikanie jedynie przycisków: „Dalej", „Dalej", „Zgadzam się", „Dalej", „Tak", „OK") oraz czasem automatycznych potwierdzeń wymaganych dla kwestii mało istotnych [24], które – mimo, iż z założenia i w pojedynczych przypadkach są pozytywne i bardzo skuteczne – w praktyce przyzwyczajają użytkownika do konkretnej sekwencji potwierdzeń po każdej wykonanej operacji. Stałe (niekiedy uciążliwe) wykonywanie takich sekwencji prowadzi w końcu albo do wykształcenia u użytkowników automatyzmu w klikaniu (potwierdzaniu) komunikatów ostrzeżeń, albo wręcz – o ile zezwala na to aplikacja – do wyłączania funkcjonalności odpowiedzialnej za generowanie ostrzeżeń.

Jest oczywistym faktem, że w przypadku tuneli SSL-owych nie należy obciążać całą winą producentów przeglądarek. Niestety, bardzo często w Internecie spotkać się można z błędnymi certyfikatami, które także przyzwyczajają użytkowników do komunikatów błędów. Jednak nie jest to kwestia będąca podstawowym obiektem testów, więc nie będzie ona tu podnoszona.

Poniżej przedstawione zostały konkretne sekwencje operacji wykonywanych przez użytkownika w celu potwierdzenia (lub zanegowania) poprawności certyfikatu. W przypadku poniższych analiz autorzy zalecają zwrócenie uwagi na następujące kryteria oceny przeglądarek:

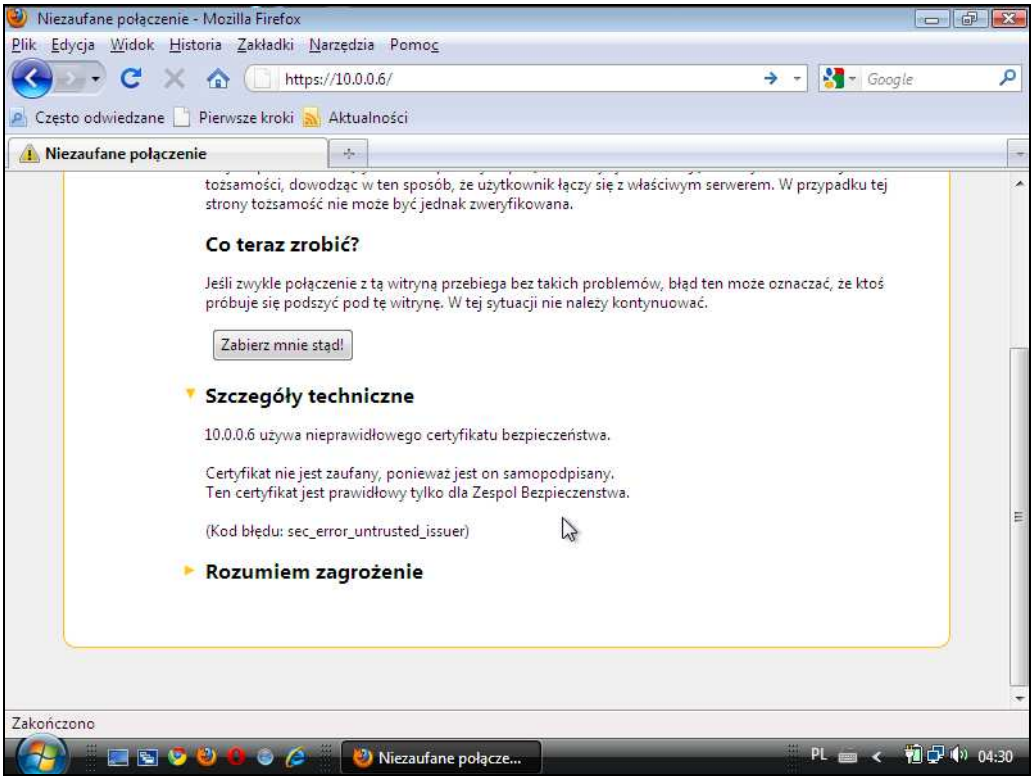
- Liczba kroków prowadzących do zatwierdzenia błędnego certyfikatu,
- Czytelność komunikatu o błędzie,
- Szczegółowość informacji o błędzie,
- Standardowe ustawienia (w tym także dotyczące zapamiętywania decyzji użytkownika),
- Jasność i czytelność ikon sygnalizujących obecność połączenia SSL, ewentualnie także jego parametry (np. „kłódeczka”),
- Inne mechanizmy blokowania „nieprzemyślanych decyzji”.



Rysunek 4. Mozilla Firefox: Pierwsza informacja po rozpoznaniu błędnego certyfikatu/połączenia – krok 1

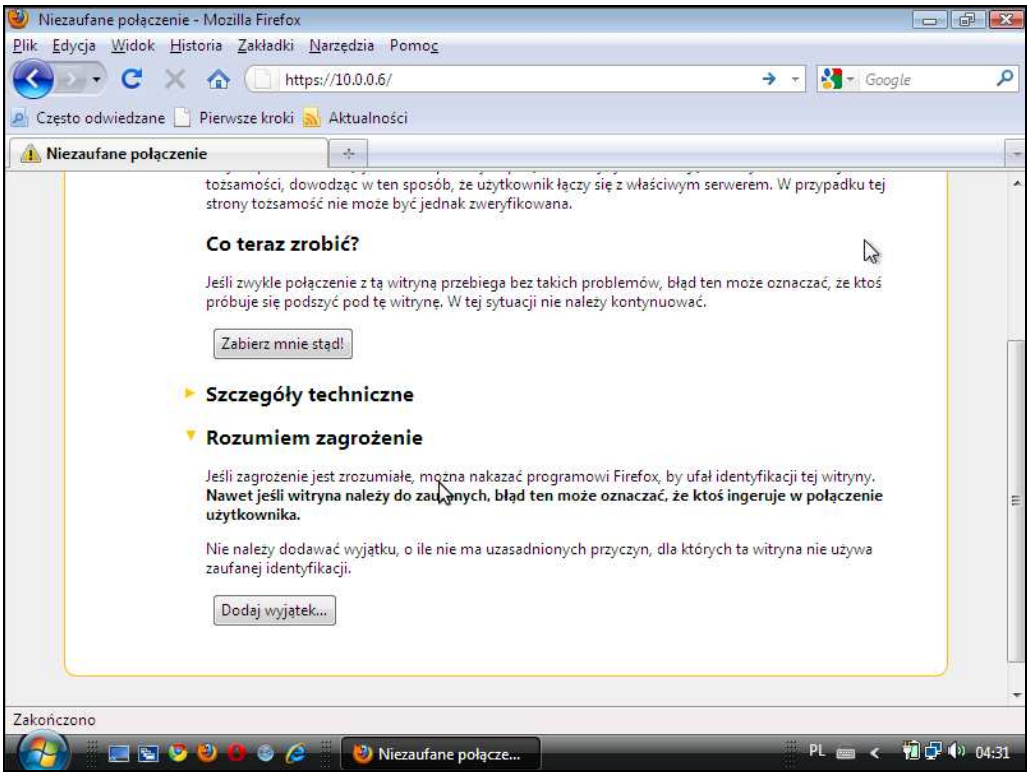
Rysunek 4 przedstawia pierwszy krok – informację wstępną, pokazywaną w przeglądarce Mozilla Firefox w przypadku rozpoznania błędów w połączeniu lub certyfikacie. Wielu użytkowników może nie zechcieć poświęcić czasu na studiowanie komunikatów, które zawierają więcej niż kilka linijek tekstu. Na szczęście, mimo stosunkowo dużej ilości wyświetlonej informacji, projektantom przeglądarki Mozilla Firefox udało się skutecznie zwrócić uwagę użytkownika na podstawowe elementy przekazu. Już po pierwszym spojrzeniu użytkownik widzi rzucające się w oczy ostrzeżenie, a dalej informacje o tym, jakie kroki może w związku z tym poczynić. Co ciekawe – poniżej znajdują się linki (przyciski) z których jeden wygląda podobnie do przycisków spotykanych

w standardowych aplikacjach desktopowych. To właśnie ten przycisk jest naciskany odruchowo przez użytkowników, którzy w ogóle nie zapoznają się z informacją z wygenerowanej przez przeglądarkę strony. Po wciśnięciu tego przycisku użytkownikowi nie stanie się żadna krzywda, a przeniesienie (na stronę startową), które bezpośrednio po nim nastąpi, spowoduje jedynie zdziwienie użytkownika i – być może – nakłoni go do próby analizy sytuacji i zastanowienia się, dlaczego zawartość strony, którą chciał odwiedzić, spowodowała przekierowanie.



Rysunek 5. Mozilla Firefox: Szczegóły techniczne błędnego połączenia – krok 2

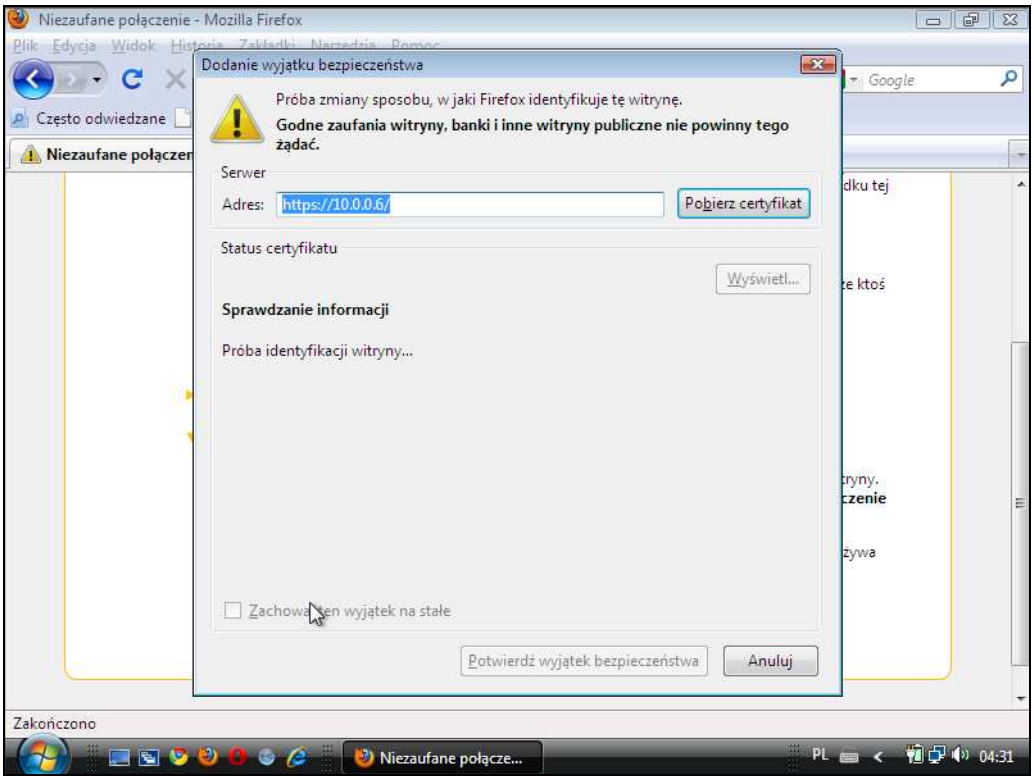
Opis, jaki prezentuje Rysunek 5, jest szczególnym ukłonem w stronę użytkowników trochę bardziej zaawansowanych technicznie. Znaleźć tam można dodatkowy opis zaistniałego błędu, który z pewnością pomoże w faktycznej analizie zagrożenia, na jakie użytkownika może narazić błędne połączenie SSL/TLS.



Rysunek 6. Mozilla Firefox: przeniesienie odpowiedzialności i dalsze uświadamianie użytkownika – krok 3

Bezpośrednio pod „Szczegółami technicznymi" umiejscowiono link „Rozumiem zagrożenie", którego nazwa przenosi niejako na użytkownika odpowiedzialność za otwarcie danego połączenia, co potwierdza opis, który pojawia się bezpośrednio po kliknięciu linku. Przedstawia to Rysunek 6.

Zauważyć tu również można, iż projektanci przeglądarki Mozilla Firefox przyjęli interesujące założenie dla operacji na certyfikatach. Otóż każde wykorzystanie połączenia z błędami jest wyjątkiem (niezależnie od faktu, czy jest to wyjątek pojedynczy, czy zapamiętany) i tak właśnie – a nie jako reguła – powinno być traktowane przez użytkownika! Autorzy raportu często spotykają się z problemem braku świadomości wśród administratorów, którzy w skrajnych przypadkach potrafią wręcz dodać instrukcję dodawania błędnego (tj. niepodpisanego przez ogólnie uznane Centrum Autoryzacji) certyfikatu do wyjątków w przeglądarkach lub innych programach działających na tunelach tylko po to, zmniejszyć nakład organizacyjny niezbędny do uzyskania certyfikatu podpisanego przez zewnętrzną, zaufaną stronę trzecią (Centrum Autoryzacji).



Rysunek 7. Mozilla Firefox: dodanie wyjątku w regułach tuneli SSL/TLS – krok 4

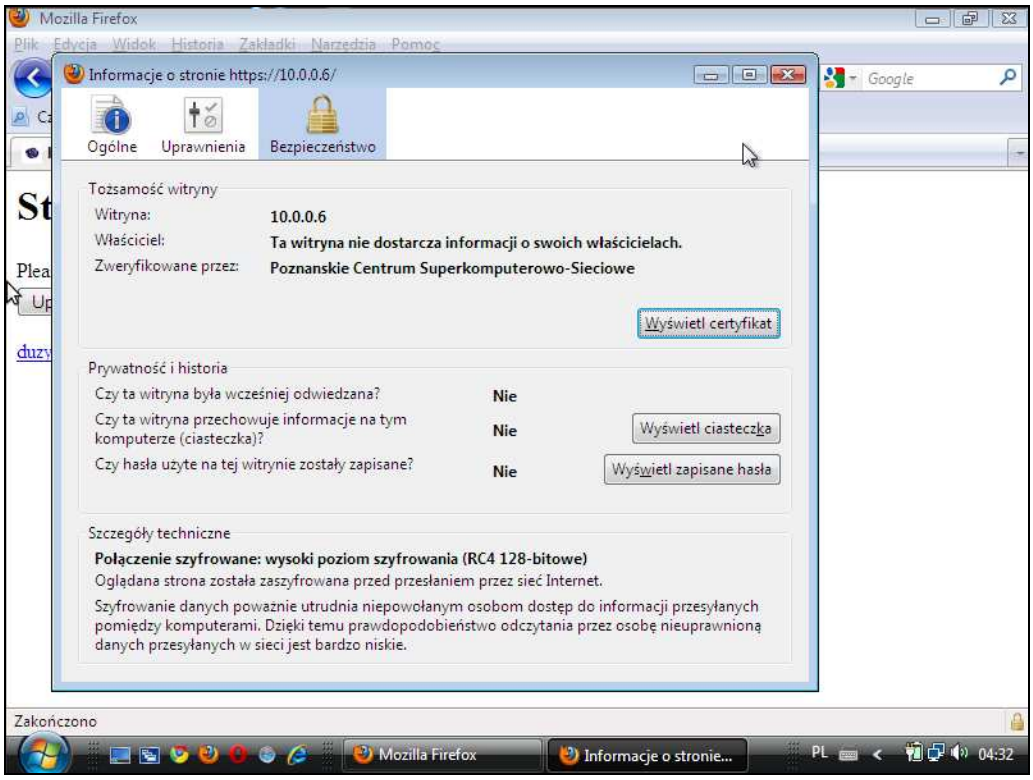
W następnym kroku użytkownik musi potwierdzić wyjątek, zapoznając się ze szczegółami problemu, w związku z czym pobierany jest ponownie (ręcznie lub automatycznie) certyfikat danego serwera, co zajmuje pewien – widoczny dla użytkownika – interwał czasowy. W opinii autorów, jest to działanie celowe ze strony projektantów, gdyż każde tego typu działanie skupia uwagę użytkownika na mechanizmach zabezpieczenia strony, a nie na samej stronie, jak z pewnością byłoby w przypadku potwierdzenia wyjątku przy pomocy pojedynczego kliknięcia.

Jedynym niedociągnięciem ze strony producentów przeglądarki Mozilla Firefox jest tu w większości przypadków standardowe ustawienie pola wyboru opcji „Zachowaj ten wyjątek na stałe”, który sugerować może przyjęcie przez użytkownika stanu aktualnego jako normalnego (co, choć istotnie wygodniejsze dla użytkownika, z punktu widzenia bezpieczeństwa jest oczywiście nie do przyjęcia). W przypadku jednak tak długiego procesu weryfikacji certyfikatu, pojedyncze dodatkowe kliknięcie nie ma tu już aż tak wielkiego znaczenia. Uwaga użytkownika została już bowiem zwrócona na problem.

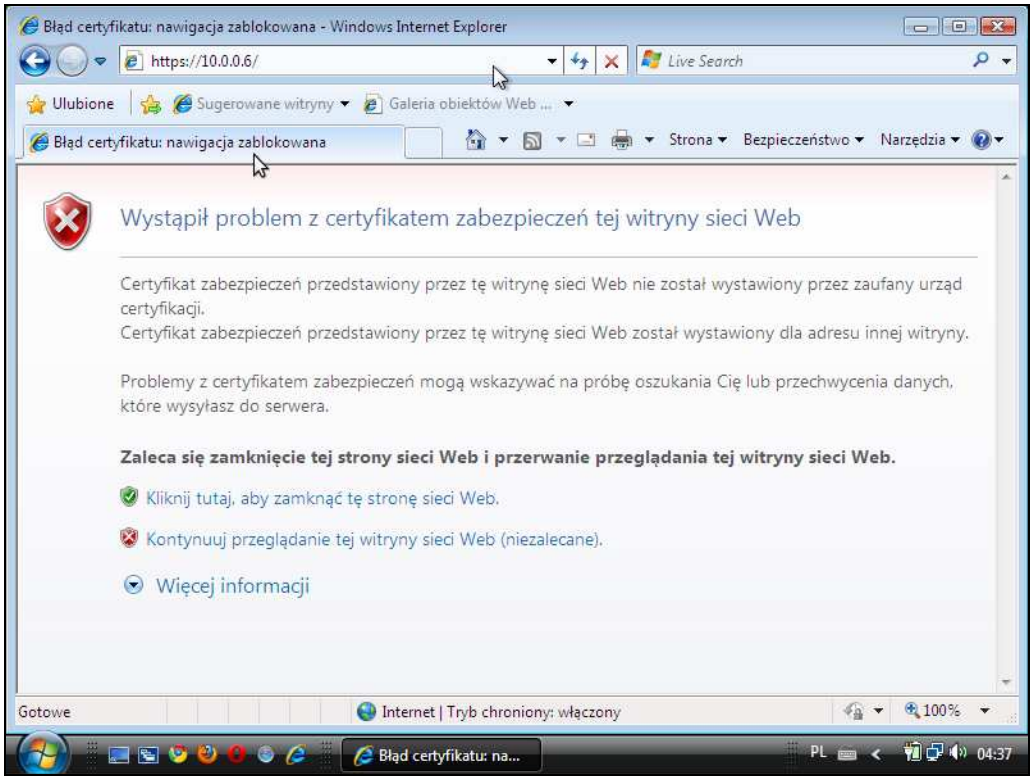
Rysunek 8. Mozilla Firefox: Wyświetlanie strony z dodanym wyjątkiem – krok 5

Jak pokazuje Rysunek 8, dodanie wyjątku powoduje akceptację przez użytkownika stanu aktualnego dla danego certyfikatu i połączenia, co skutkuje w praktyce brakiem wyświetlania przez przeglądarkę dalszych informacji na temat problemów z certyfikatem. Jest to dosyć istotnym problemem, gdyż łatwo sobie wyobrazić przypadek ataku, w którym certyfikat zostaje dodany jako wyjątek do przeglądarki (np. zainstalowanej na komputerze dostępnym w kafejce internetowej), po czym inny jej użytkownik dokonując operacji (np. na spreparowanym portalu bankowości elektronicznej) może zostać łatwo oszukany, nie będąc świadomym zagrożenia.

Ważnym elementem każdej z przeglądarek w kontekście bezpieczeństwa tuneli SSL/TLS powinien być także ogólnodostępny przycisk, umożliwiający zapoznanie się z dodatkowymi informacjami na temat poszczególnych połączeń i certyfikatów je obsługujących. Od pewnego czasu utarł się zwyczaj, iż zarówno bezpieczeństwo, jak i własności tunelu SSL/TLS symbolizowane są przez żółtą kłódeczkę. W przypadku przeglądarki Mozilla Firefox – jest ona umieszczona w prawym dolnym rogu okna aplikacji. Po jej kliknięciu staje się dostępny zestaw informacji na temat połączenia (por. Rysunek 9 poniżej). Można także wyświetlić certyfikat, jednak nie ma tam niestety informacji o fakcie, iż jeszcze przed chwilą połączenie to nie było dla traktowane jako właściwe (i nadal nie jest, jeśli spowodowało ono wystąpienie błędów o nie znanej przez użytkownika przyczynie).



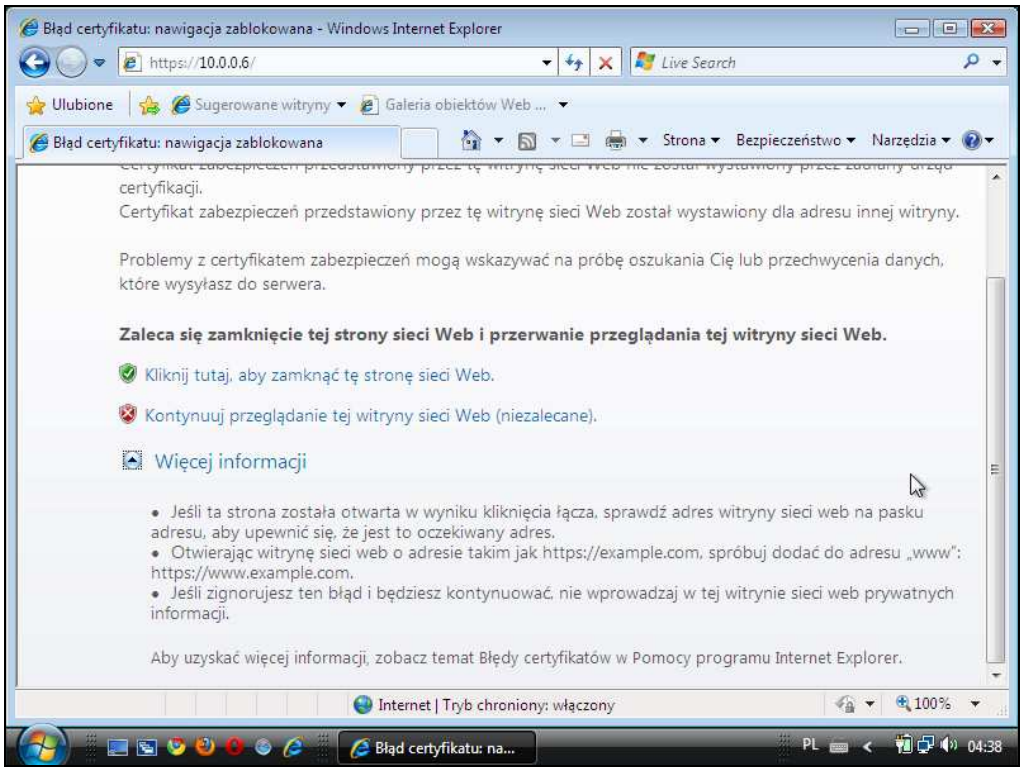
Rysunek 9. Mozilla Firefox: „kłódeczka" i informacje od niej uzyskane – krok 6



Rysunek 10. Internet Explorer: Pierwsza informacja po rozpoznaniu błędnego certyfikatu/połączenia – krok 1

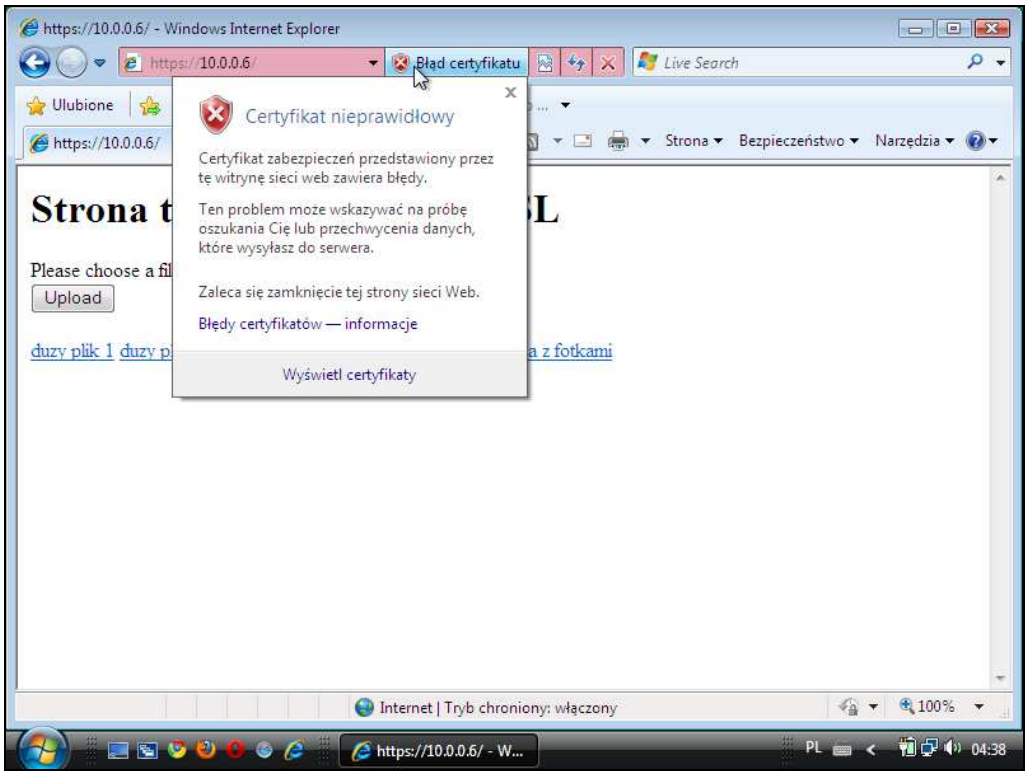
W przypadku przeglądarki Microsoft Internet Explorer początkowe informacje ukazujące problemy z nawiązaniem bezpiecznego połączenia SSL/TLS mają bardzo podobną formę, jak w przypadku

opisanego poprzednika. Strona jest czytelna i także pokazuje dosyć dużo tekstu, przedstawionego w przystępny sposób. Elementem graficznym, który jako pierwszy rzuca się w oczy, jest czerwona tarcza z białym znakiem „X", która znana jest z innych produktów tej samej firmy i prawidłowo kojarzona jest z zagrożeniem bezpieczeństwa. W tym przypadku zestaw elementów graficznych wykorzystany na stronie (2-krotnie użyta czerwona tarcza i raz zielona) daje „automatycznym klikaczom" złudzenie, iż to po kliknięciu zielonej tarczy wyświetlona zostanie zawartość strony. Jak łatwo można się domyślić, jest to zabieg celowy, który ma uchronić tego typu nieświadomych użytkowników przed podstawowymi zagrożeniami.



Rysunek 11. Internet Explorer: Szczegóły techniczne błędnego połączenia – krok 2

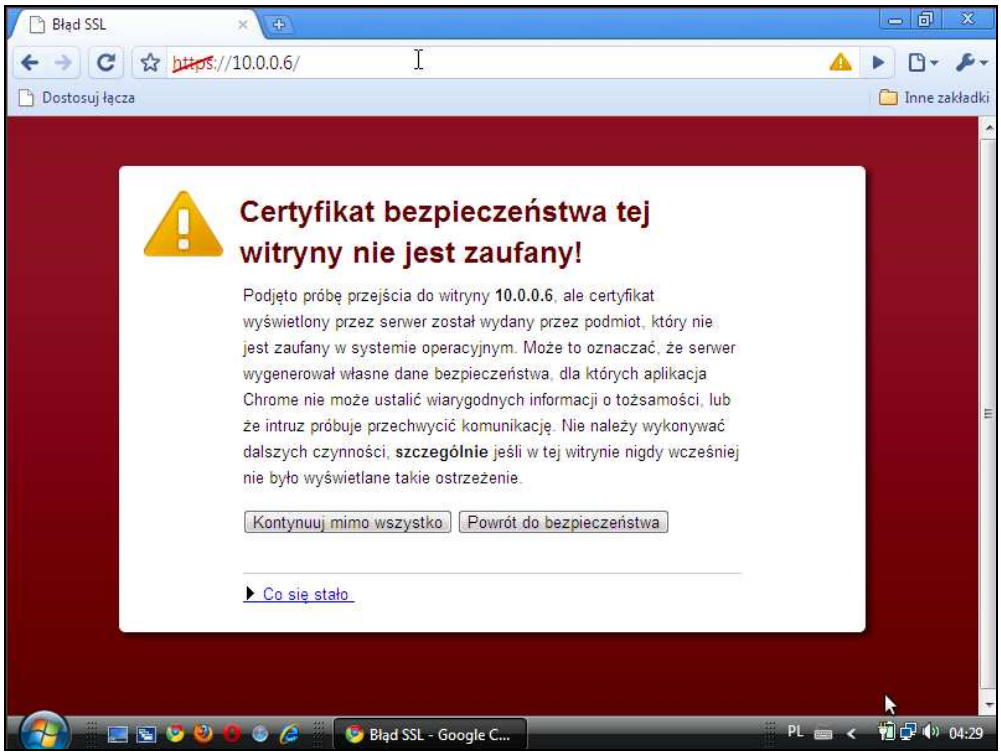
Naturalnym niemalże wydaje się także pokazanie przez producenta na pierwszej stronie przycisku, umożliwiającego bardziej doświadczonym użytkownikom uzyskanie szczegółowych informacji na temat problemu, który wystąpił. Ku zdziwieniu użytkownika – znajduje się tam jedynie zestaw standardowych rad, niezależnych od obecnego połączenia, które sugerują użytkownikowi możliwe rozwiązania użytkownikowi. Jest to oczywiście element, który należy ocenić pozytywnie – jednak mógł on zostać dużo bardziej dopracowany.



Rysunek 12. Internet Explorer: niezaufany certyfikat – krok 3

W testowanej (najnowszej) wersji programu Microsoft Internet Explorer także zauważyć można w okolicach prawego dolnego rogu okna przeglądarki kłódeczkę, która kojarzy się z szyfrowanymi połączeniami SSL/TLS. Tu także spotkać może zorientowanego technicznie użytkownika niemiłe zaskoczenie, gdyż okazuje się, iż kłódeczka ta (ze strzałką) odpowiedzialna jest za zupełnie inne mechanizmy, które wydają się być dużo mniej znaczące niż bezpieczeństwo szyfrowanych tuneli. Ostateczną ocenę w tym punkcie autorzy raportu pozostawiają Czytelnikowi.

Bardzo miłym zaskoczeniem jest tu jednak inny aspekt strony wyświetlanej przez przeglądarkę, a przesyłanej przy pomocy tunelu zawierającego błędy. Chodzi tu mianowicie o przycisk umieszczony na pasku adresu: „Błąd certyfikatu” (wraz ze znaną już ikoną czerwonej tarczy), a także o kolor samego paska, sygnalizujący niebezpieczeństwo. Jak łatwo można się domyślić, po kliknięciu przycisku uzyskać można kolejną garść porad wraz z linkiem prowadzącym do szczegółów technicznych certyfikatu. Tego typu podejście projektantów w opinii autorów raportu zasługuje na znaczącą pochwałę.

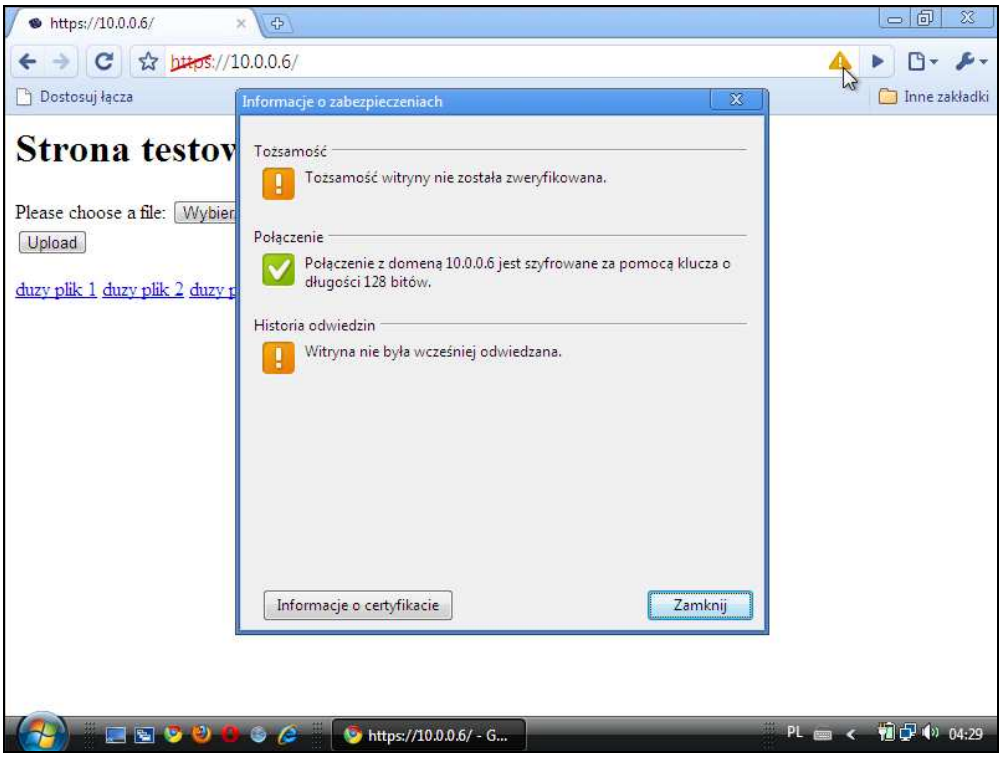


Rysunek 13. Google Chrome: Pierwsza informacja po rozpoznaniu błędnego certyfikatu/połączenia – krok 1

Kolejny produkt, mogący obsługiwać bezpieczne połączenia SSL/TLS, to przeglądarka Google Chrome. W jej przypadku problemy z połączeniem SSL/TLS (tak samo jak inne problemy bezpieczeństwa) sygnalizowane są przez bardzo widoczny czerwony kolor tła pierwszego komunikatu. Jak wskazuje Rysunek 13, pierwszym rzucającym się w oczy elementem jest komunikat o fakcie zaistnienia problemu. Niestety, projektanci tego narzędzia nie poszli w ślady autorów uprzednio opisanych aplikacji i nie rozróżnili znacząco (np. piktogramami czy kolorami) przycisków potwierdzenia i zaprzeczenia woli przeglądania treści danej strony mimo ostrzeżenia. Powodować to może problemy dla osób, które szybko i odruchowo będą szukały przycisku kontynuacji operacji, przez co niepotrzebnie mogą narazić się na ataki.

Po dokonaniu potrzebnych potwierdzeń, użytkownik trafia na upragnioną stronę internetową. Jej widok przedstawia Rysunek 14. Jak widać, projektanci chcieli zaznaczyć na niej problemy z tunelem/certyfikatem i wykonali to za pomocą symbolicznego przekreślenia (i zaznaczenia w kolorze czerwonym) nazwy protokołu w pasku adresu. W następnej wersji ww. przeglądarki (nie uwzględnionej w niniejszym raporcie, ponieważ występuje ona w wersji beta i to nie na wszystkie systemy operacyjne) problem ten został jeszcze bardziej uwydatniony przez dodanie w pasku adresu symbolu trupiej czaszki z piszczelami.

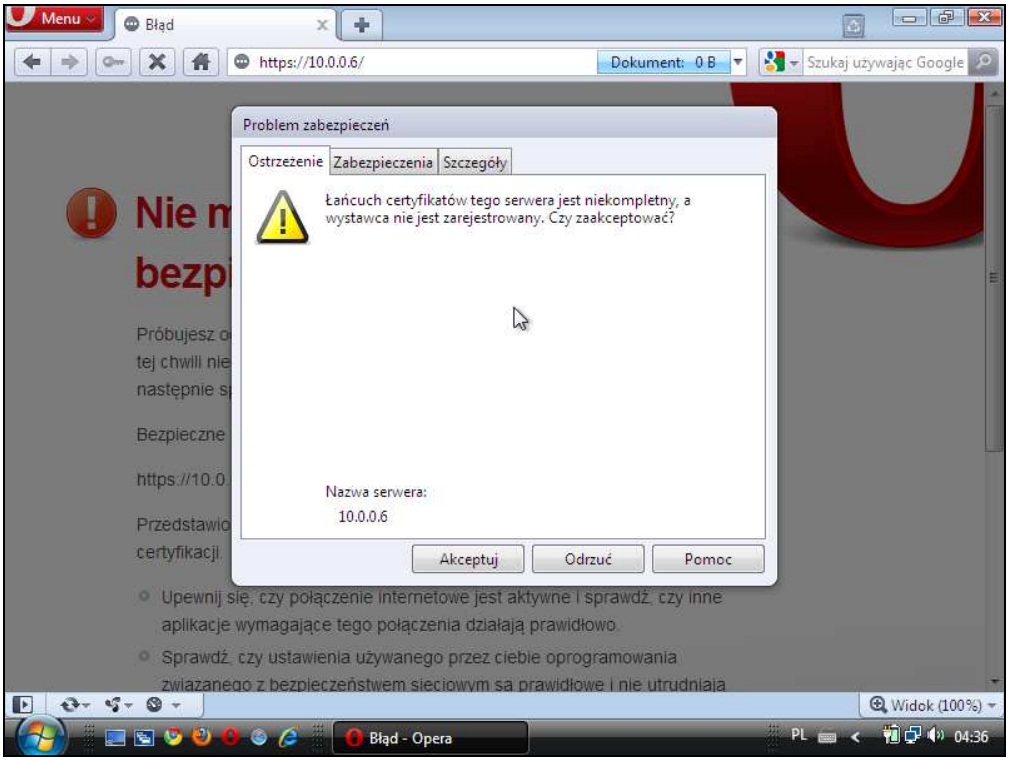
Rysunek 14. Google Chrome: Widok strony po połączeniu – krok 2



Rysunek 15. Google Chrome: informacje dodatkowe – krok 3

Innym elementem sugerującym możliwe problemy jest znaczek trójkąta ostrzegawczego położonego po prawej stronie paska adresu. Po kliknięciu trójkąta ukazuje się okno dialogowe, zawierające niektóre kwestie krytyczne rozpoznane i zweryfikowane przez przeglądarkę. Niestety,

jest to bardzo ograniczony zestaw problemów, który nie zawiera szczegółowych informacji o samym ostrzeżeniu w przypadku testowanego połączenia. Tak samo, jak w przypadku Internet Explorera, dostępny jest link do informacji szczegółowych (przycisk w lewym dolnym rogu okna) o samym certyfikacie, co nie jest jednak tożsame z pełnym zestawem informacji opisujących dane połączenie.



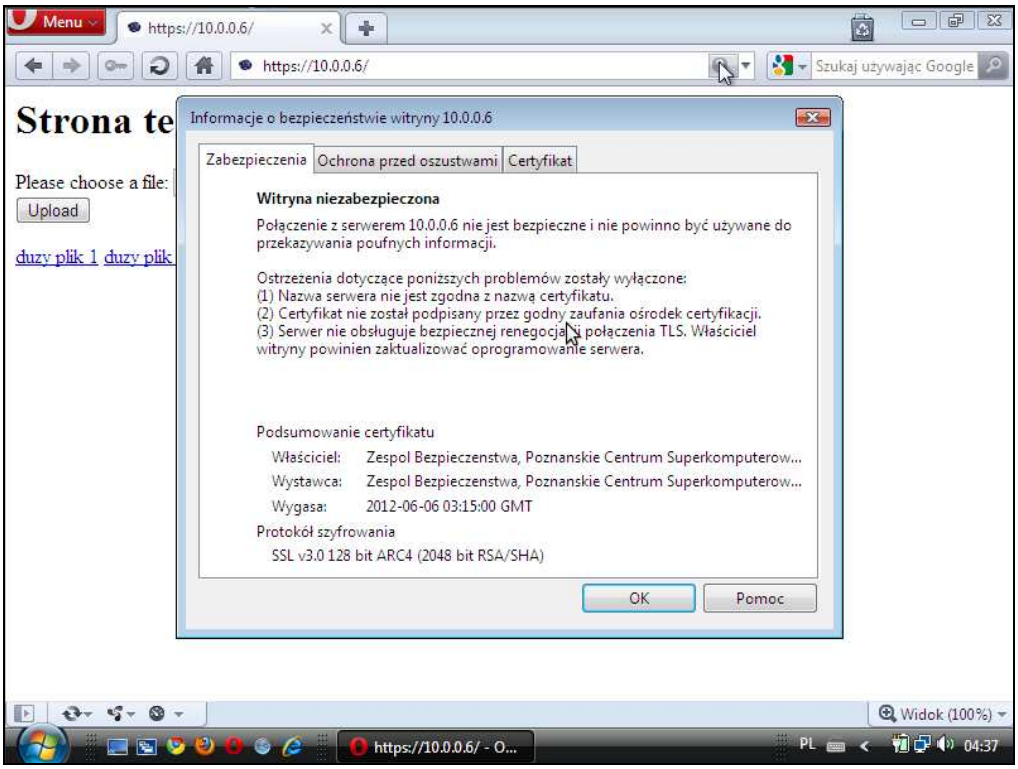
Rysunek 16. Opera: Widok strony po połączeniu – krok 1

W przypadku przeglądarki Opera, obraz pierwszej strony po próbie nawiązania połączenia obrazuje Rysunek 16. Wyraźnie widoczne jest inne podejście projektantów tego narzędzia do sposobu interakcji z użytkownikiem. Nie jest to już zawartość umieszczona w miejscu, gdzie internauta spodziewa się treści oczekiwanej strony, ale za to informacje na temat problemów wyświetlane są jako osobny komunikat w oknie dialogowym. Dzięki temu (i dodatkowemu dodaniu w tle strony informującej o błędzie oraz wyszarzeniu tego tła) użytkownik w ułamku sekundy orientuje się, że rozpoznany problem nie jest trywialny.

Niestety, projektanci ww. przeglądarki nie uchronili się przed błędem potwierdzenia pojedynczym kliknięciem, przez co użytkownik, wykorzystując standardowe przyciski, znajdujące się w spodziewanym (standardowym) miejscu okna dialogowego, może zadziałać zbyt odruchowo i potwierdzić wyjątek.

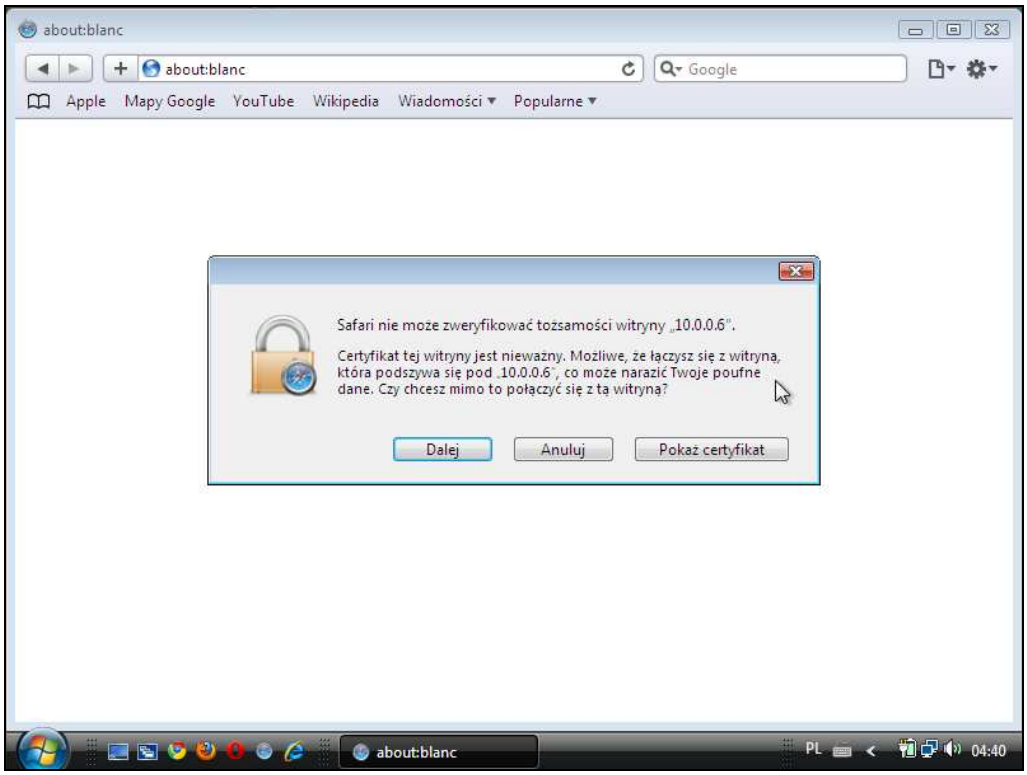
Innym aspektem, sugerowanym jako jedno z kryterium oceny przeglądarki przez użytkownika, była kwestia standardowych ustawień stałego zapamiętywania wyjątków. Należy przyznać, że Opera problem ten rozwiązała bardzo dobrze. Przycisk, służący do zapisywania wyjątków na stałe,

przeniesiony został na inną zakładkę (przez co w celu jego użycia potrzebne jest jeszcze jedno, świadome, kliknięcie) i przy standardowych ustawieniach jest domyślnie wyłączony!



Rysunek 17. Opera: Wyświetlona strona oraz informacje dodatkowe – krok 2

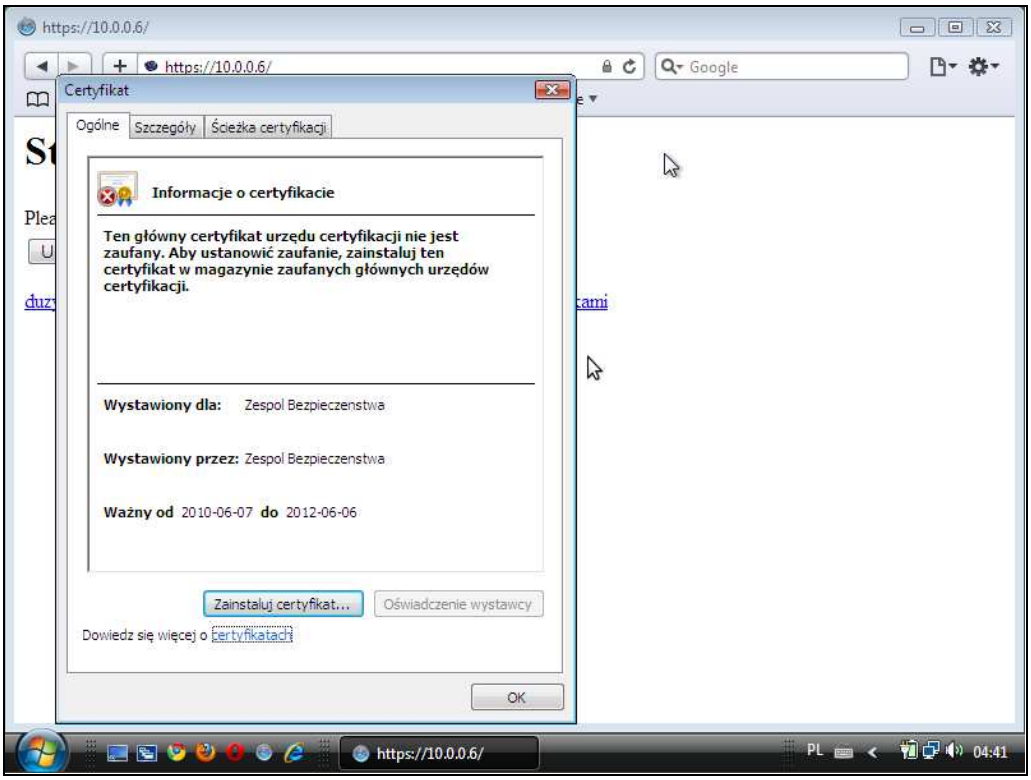
Po wyświetleniu strony, nie zawiera ona niestety znaczących elementów, które mogłyby wskazywać na problemy z szyfrowanym połączeniem. Widoczny jest jedynie (po prawej stronie paska adresowego) mały znaczek szarego znaku zapytania, który kryje bardzo dużo ciekawych informacji na temat samego połączenia. W oknie, otwartym po kliknięciu znaku zapytania, widoczne są trzy zakładki, opisujące poszczególne problemy wraz ze szczegółami połączenia oraz certyfikatu. Bezspornie zaprezentowanie tego typu danych stawia Operę na pierwszym miejscu wśród testowanych przeglądarek w omawianej kwestii.



Rysunek 18. Safari: niezaufany certyfikat – krok 1

Ostatnią testowaną przeglądarką był produkt Apple Safari. Testując pierwszą wyświetloną podczas nawiązywania połączenia stronę, trudno powiedzieć, czy projektanci poświęcili dodatkowy czas na analizę zachowania użytkownika w przypadkach błędów występujących w szyfrowanych tunelach. Na podstawie elementów takich jak: domyślnie zaznaczony przycisk potwierdzenia, brak znaczących symptomów ostrzegających o zagrożeniu, ale też potwierdzenie wyjątku pojedynczym kliknięciem, można przypuszczać, że analiza taka nie została przeprowadzona lub położono na nią zbyt mały nacisk.

Dodatkowym problemem stał się w tym przypadku także interaktywny pasek adresu, który w opinii autorów niniejszego raportu był przyczyną ograniczeń możliwości analizy problemu z tunelami (chodzi konkretnie o brak lub sposób wyświetlania adresów podczas nawiązywania połączenia). Dokładnie mówiąc, w zależności od specyficznych warunków przejścia na stronę (wpisanie adresu, kliknięcie linku, błąd w podstronie/tabelce z opóźnionym czasem wyświetlania), pasek adresu zostawał wypełniony docelowym adresem lub pozostawał opisany zgodnie z poprzednim adresem strony, co nie pozwalało na pierwszy rzut oka jednoznacznie określić przyczyny błędu (jego lokalizacji).



Rysunek 19. Safari: niezaufany certyfikat – krok 2

Po dodaniu wyjątku i nawiązaniu połączenia łatwo jest odszukać zwyczajową kłódeczkę, aby zweryfikować własności połączenia, jednak po jej naciśnięciu wyświetlane jest tylko standardowe okno systemowe, zawierające informacje o certyfikacie, które znane jest już z linków dostępnych w innych przeglądarkach.

Niestety i tutaj nie przemyślano budowy poszczególnych elementów (pasek adresu, mechanizm ostrzeżeń itp.), co wskazuje jednoznacznie, iż interfejs omawianej przeglądarki nie był wystarczająco analizowany pod kątem bezpieczeństwa podczas procesu projektowania.

2.4.3 Przesyłanie danych w błędnym tunelu ukrytym w stronie

Znaczącą kwestią w trakcie przeglądania zwykłych stron internetowych jest możliwość weryfikacji, w jaki sposób przesyłane są wszystkie elementy na stronie. Można wyobrazić sobie sytuację, w której część materiałów przesyłana jest tekstem otwartym, zgodnie z głównym protokołem witryny, a część przechodzi tunelem SSL/TLS, który na stronie jest ukryty.

Momentalnie nasuwa się pytanie, dlaczego część takiej strony miałaby być przesyłana w postaci szyfrowanej. Otóż także agresorzy, atakujący nasze zasoby, pragną względem niektórych systemów (Systemy Detekcji i Prewencji przed Intruzami – IDS/IPS) pozostać niewykryci.

Podstawowe zadanie, które może wykonać tu przeglądarka, to pokazanie użytkownikowi w prosty, ale dający dużo informacji sposób, iż w zawartości strony istnieje dodatkowy tunel, który nie odpowiada parametrom, jakie wystąpiłyby przy tworzeniu standardowego tunelu do obsługi połączeń szyfrowanych. Można tu wyobrazić sobie kilka możliwych przypadków ukrycia

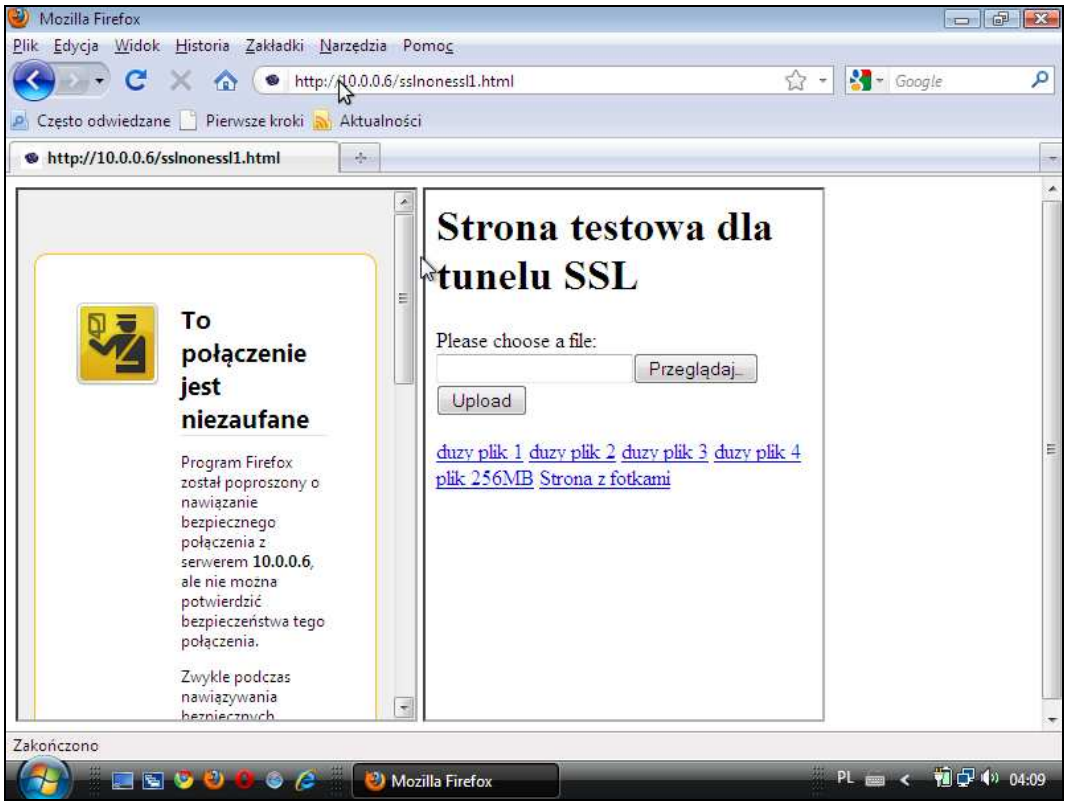
dodatkowego tunelu, jednak autorzy wykorzystali tu jedynie opcję budowy tunelu w tej samej domenie, jednak z niepoprawnym certyfikatem.

Poniższe zrzuty ekranu pokazują zachowanie poszczególnych przeglądarek przy prezentacji prostej strony, której jednym z elementów jest ramka, zawierająca zawartość przesyłaną tunelem SSL/TLS.

Dla użytkownika kryterium oceny wykorzystania każdej z poniższych przeglądarek powinny być w tym przypadku:

- sam fakt poinformowania użytkownika o pojawieniu się błędnego tunelu w treści strony,
- sposób zaznaczenia obecności treści przesyłanych tunelem (wyróżnienie lub specjalne wskazanie),
- monit informujący o obecności takiej treści (przydatny zwłaszcza w przypadku tuneli ukrytych w bardzo małych obiektach, które mogłyby zostać przeoczone przez użytkownika).

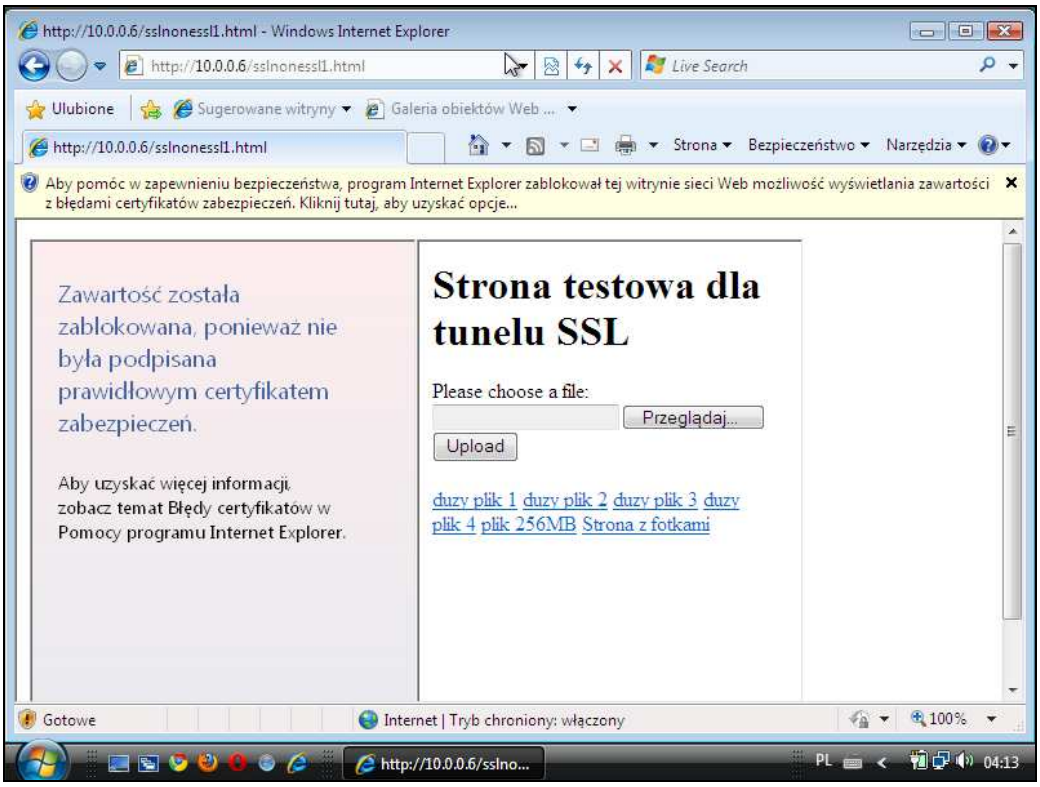
Zaprezentowana ramka SSL/TLS (jak i jej nieszyfrowany odpowiednik) została przez autorów raportu celowo uwydatniona, aby pokazać, w jaki sposób producenci przeglądarek pokazują tę różnicę.



Rysunek 20. Ukryty tunel: Mozilla Firefox

Obsługa ukrytego w stronie tunelu w przypadku przeglądarki Mozilla Firefox jest dosyć znacząco zaakcentowana, a co ważniejsze – mimo braku monitu, użytkownik nie ma możliwości prostej akceptacji parametrów tego tunelu, jeśli sama ramka będzie znacząco mała. Widok ramki jest bardzo podobny do ramki, która pojawia się przy standardowych błędach tuneli SSL/TLS, gdy cała

strona jest przesyłana w ww. tunelu. Aby potwierdzić i/lub dodać wyjątek dla tego tunelu, należy przejść wszystkie standardowe kroki jak w przypadku normalnego tunelu.



Rysunek 21. Ukryty tunel: Internet Explorer

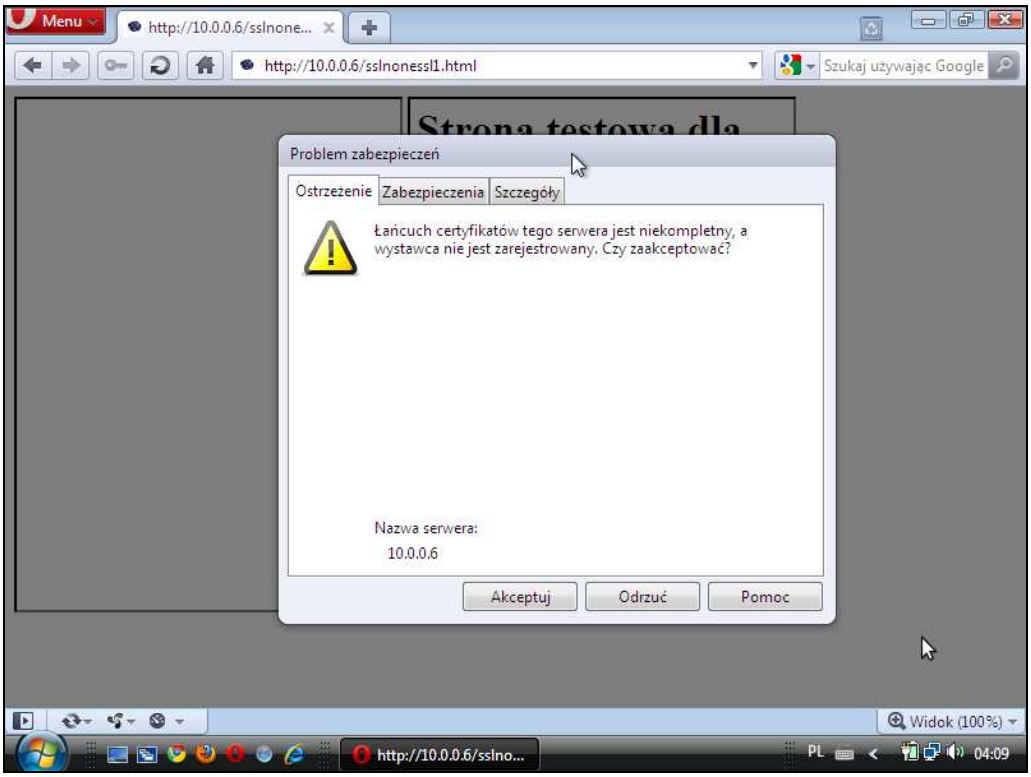
W przypadku przeglądarki Internet Explorer zauważyć można znany mechanizm ostrzeżenia przed (z różnych względów) niestandardową treścią, pokazujący się na samej górze treści wyświetlanej strony. Komunikat taki będzie widoczny niezależnie od tego, w jakim rozmiarze zostanie umieszczony element na stronie (a także w przypadku wykorzystania innych metod ukrycia treści). Sama informacja pokazywana w miejscu umieszczenia tunelu nie jest tu tak znacząco uwydatniona, jak jest to zaimplementowane w przypadku innych przeglądarek, jednak odmienna taktyka producenta może tu mieć także swoje pozytywne aspekty.

Autorzy raportu zalecają tu (jak i w każdym z punktów), niezależne testy własne, w celu analizy, który z mechanizmów jest najwygodniejszy (i oczywiście najskuteczniejszy) dla Czytelnika.

Rysunek 22. Ukryty tunel: Chrome

Interesującym podejściem może pochwalić się przeglądarka Google Chrome, której twórcy bardzo starają się uwidocznić elementy mogące powodować w późniejszym czasie (po odblokowaniu) problemy. Kolor i ogólny wydźwięk ramki pasuje do ogólnej polityki producentów tejże przeglądarki i może być (bardzo słusznie) kojarzony z monitami na temat niebezpiecznych domen (z wykorzystaniem mechanizmów Google SafeBrowsing) lub z domenami o niskiej reputacji (dane pobierane z innych systemów).

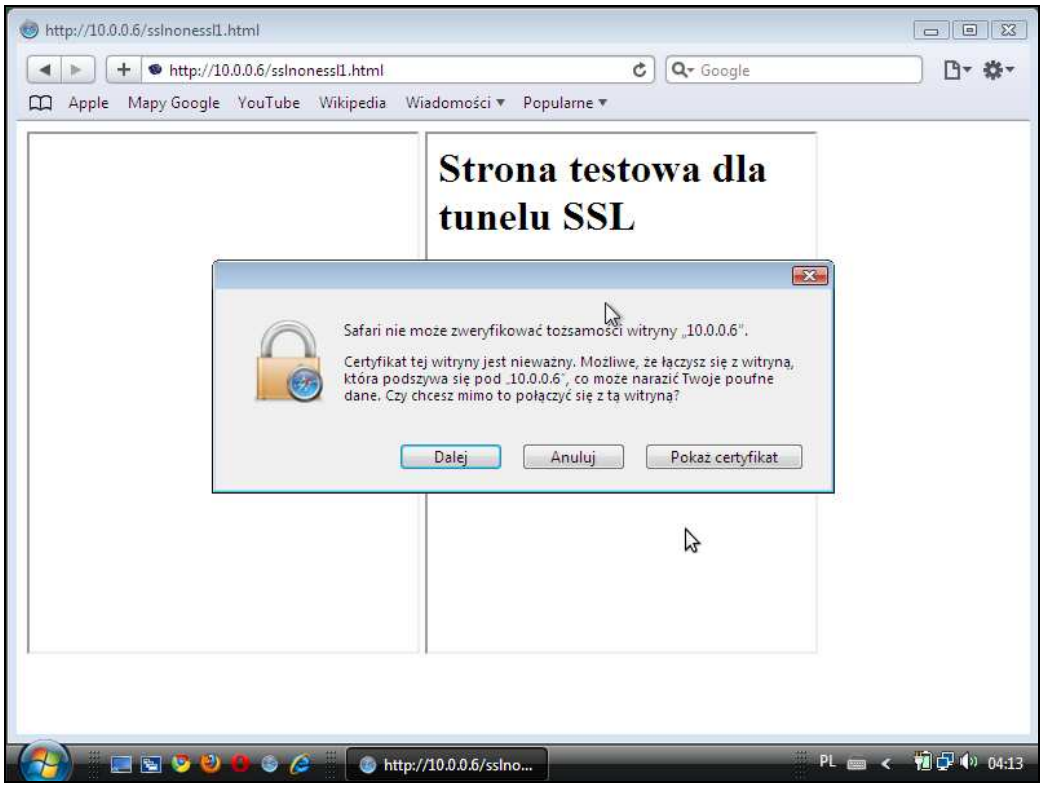
Wartym zaznaczenia jest tu także fakt, iż przeglądarka ta nie daje użytkownikowi możliwości decyzji, czy dany tunel powinien być odblokowany, czy też nie. Można wyobrazić sobie zarówno przypadki, gdzie zachowanie takie jest ze wszech miar pożądane, jak i scenariusze, gdy może to być znacząco uciążliwe. Jednak, aby nie wchodzić w zbyt daleko idące dywagacje na temat odrębnych aspektów bezpieczeństwa, autorzy postanowili pozostawić te rozmyślania w gestii bardziej dociekliwych Czytelników.



Rysunek 23. Ukryty tunel: Opera

W przypadku najnowszej przeglądarki z serii Opera rzucają się od razu w oczy trzy pozytywne aspekty interfejsu. Pierwszym z nich jest fakt, iż nie ma możliwości przeoczenia tunelu w zawartości strony (monit jest tu wyświetlony dla całej strony, mimo iż w pasku adresu widnieje protokół http). Kolejny element, na który należy zwrócić uwagę, to domyślne ustawienie, zabraniające przeglądarce dodawania wyjątku dla danej strony (które można oczywiście zmienić). Na koniec bardzo znaczący dla użytkownika element – informacje uzyskiwane od przeglądarki są tu bardzo czytelne i dokładne. W przypadku tego produktu bardziej świadomi użytkownicy poczują się mile zaskoczeni faktem, iż przeglądarka (pośrednio przez producentów) nie traktuje ich jak „przeciętnych surferów". Z kolei mniej doświadczeni użytkownicy, których nie interesują informacje szczegółowe, mogą je zupełnie pominąć (nie klikając zakładki „Szczegóły"), nie ignorując oczywiście przy tym samego zagrożenia wynikającego z istnienia błędów na stronie.

Jedynym minusem, o którym mimo wszystkich wymienionych pozytywów nie wolno zapomnieć, jest w tym przypadku fakt, iż potwierdzenie błędnego certyfikatu dla tunelu następuje po jednokrotnym kliknięciu. Jest to nieprawidłowe założenie, ponieważ wielu użytkowników komputerów (zwłaszcza użytkowników domowych, a także pracowników firm niezorientowanych technicznie i tylko korzystających z komputerów jako np. narzędzia do komunikacji i tworzenia dokumentów) przyzwyczajonych jest do zatwierdzania wyświetlanych komunikatów – bez szczegółowego rozważania znaczenia zaprezentowanej im treści.



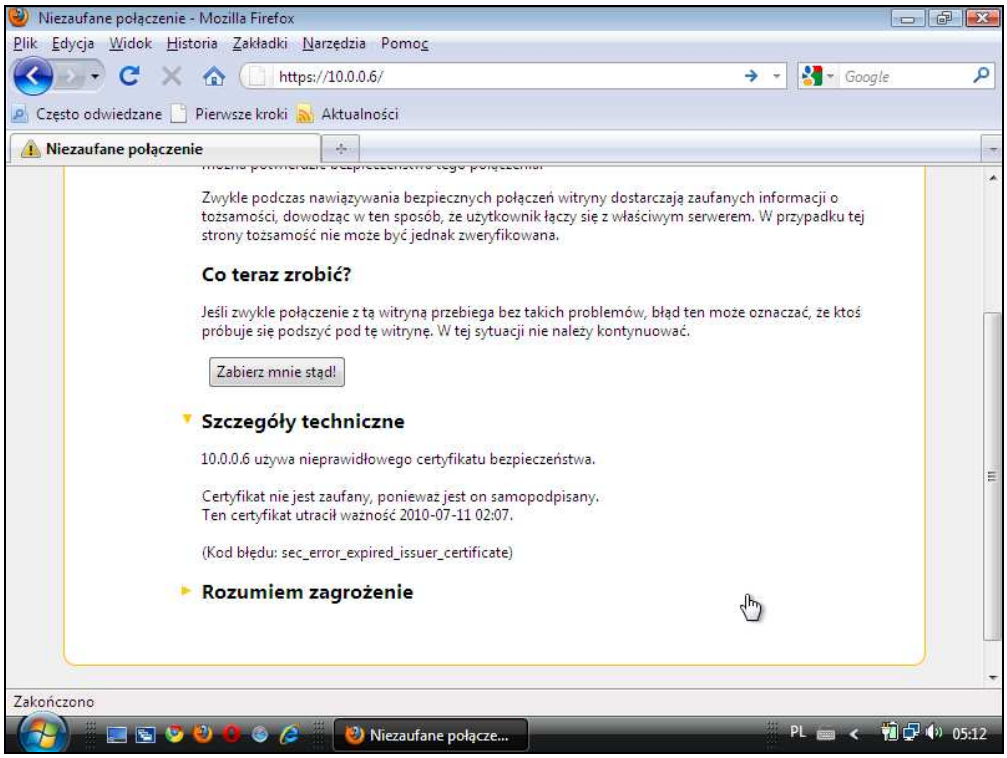
Rysunek 24. Ukryty tunel: Safari

Obsługa tuneli dosyć słabo zaprezentowana została w przeglądarce Apple Safari, jednak zastosowane mechanizmy (monit i potwierdzenie) powinny być wystarczające dla zapewnienia minimum bezpieczeństwa w przypadku otwarcia strony z ukrytym tunelem SSL.

2.4.4 Niepodpisany certyfikat

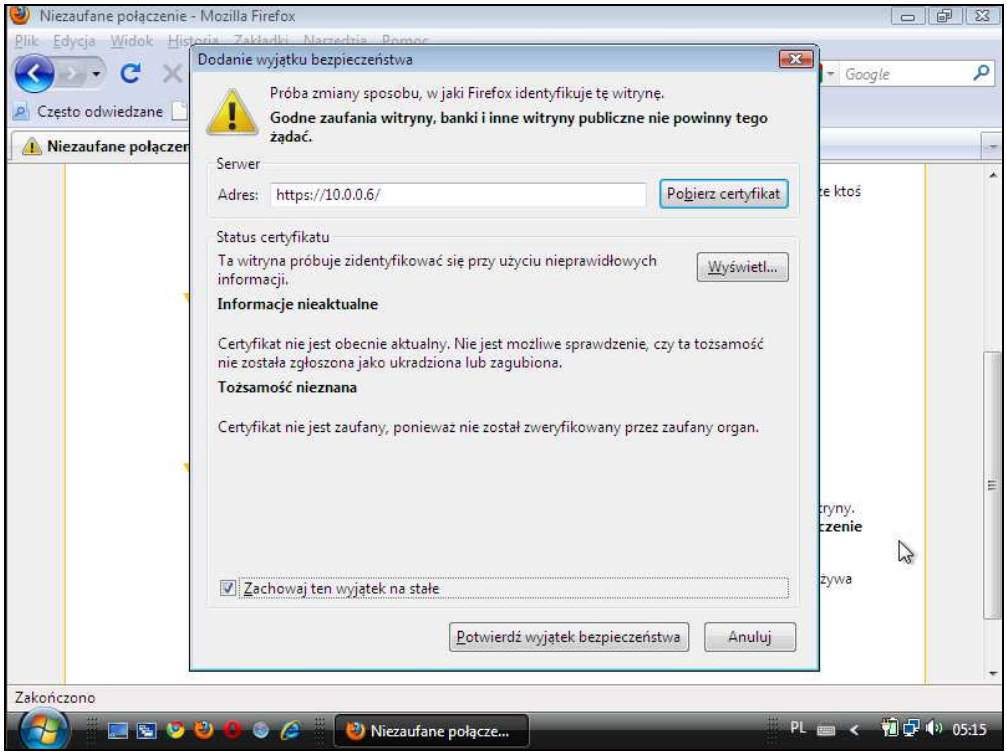
Poniższe zrzuty ekranowe przedstawiają zachowanie przeglądarek podczas łączenia się ze spreparowaną witryną, legitymującą się certyfikatem niepodpisanym przez żadne z zaufanych Centrów Autoryzacji (ang. Certification Authority, CA). Internauta może napotkać na analogiczny problem np. łącząc się ze przygotowaną przez napastnika stroną internetową po przeczytaniu maila – elementu ataku typu *phishing*. Napastnik może w takiej wiadomości zamieścić link do witryny dostępnej za pośrednictwem protokołu HTTPS i „chronionej" wygenerowanym przez siebie samego certyfikatem. Absolutnie nie jest to wyrafinowany sposób postępowania – jednakże nawet najprostsze ataki tego rodzaju są dziś wymierzane w użytkowników, którzy z usług internetowych korzystają (często z wewnętrznymi oporami) od niedawna, nie znając możliwych zagrożeń i pożądanych na nie reakcji.

Akcja, oczekiwana tu od przeglądarki, to poprawne zdefiniowanie problemu i odpowiednia jego prezentacja na każdym z etapów przeprowadzania użytkownika przez proces analizy (zarówno pierwsza strona, jak i informacje szczegółowe).



Rysunek 25. Niepodpisany certyfikat: Mozilla Firefox – krok 1

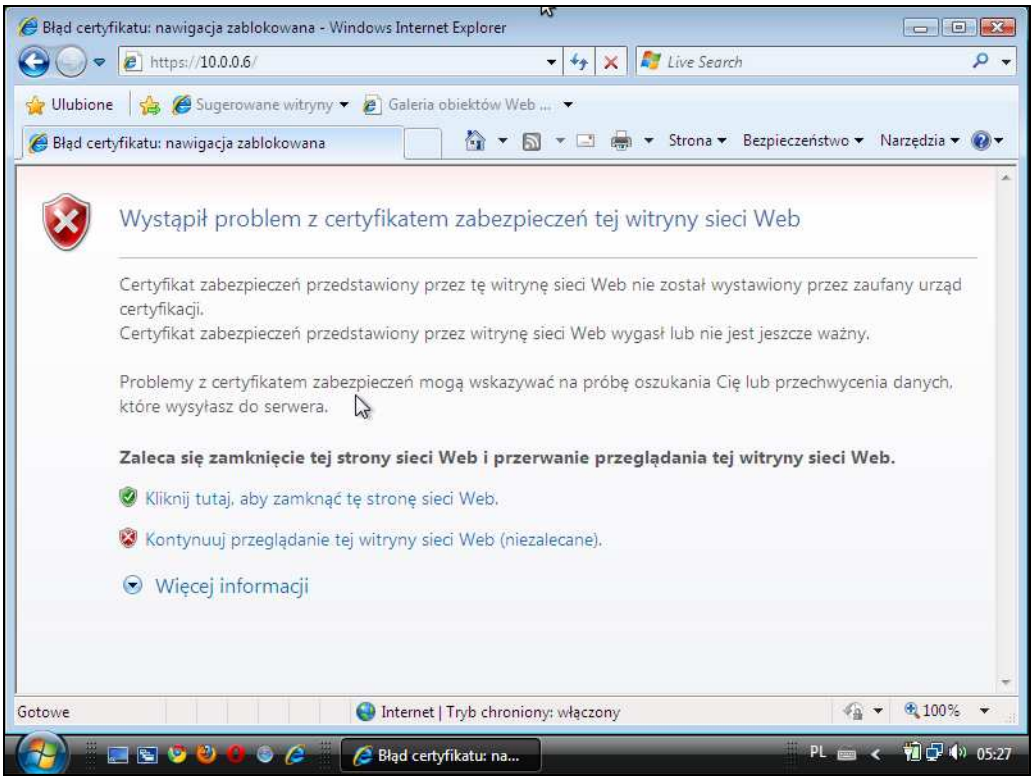
Jak pokazano we wstępie niniejszego rozdziału, użytkownik uzyskuje od przeglądarki Mozilla Firefox informacje o problemie z certyfikatem lub tunelem SSL/TLS w miejscu opisanym jako „Szczegóły techniczne". W przypadku certyfikatu, w którym problemem jest brak odpowiedniego podpisu, zauważyć można, iż został on przez przeglądarkę rozpoznany jako samopodpisany.



Rysunek 26. Niepodpisany certyfikat: Mozilla Firefox – krok 2

Klikając jednak przycisk w sekcji „Rozumiem zagrożenie", zauważyć można, że formatka dodawania wyjątku bezpieczeństwa jest dużo lepiej przygotowana. Producent zamieścił tam dokładnie wyróżnione sekcje, z których każda podejmuje inny problem związany z certyfikatem (w pokazywanym przypadku, autorzy intencjonalnie umieścili dwa niezależne problemy w pojedynczym certyfikacie, aby ukazać to rozgraniczenie). Jak widać, użytkownik już w sekcji „Tożsamość nieznana" otrzymuje wystarczająco dokładny opis, który jednoznacznie określa problem, na jaki napotkał.

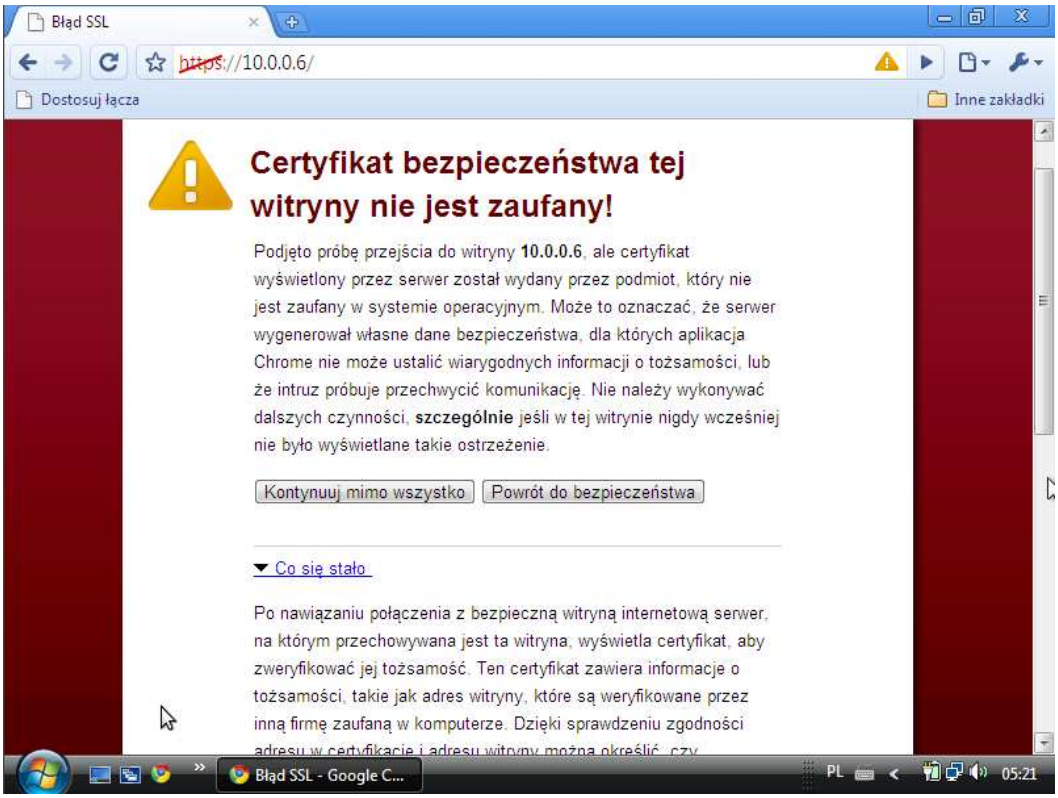
Na uwagę zasługuje tu jeszcze niestety fakt, iż przy tego typu błędach opcja „Zachowaj ten wyjątek na stałe" jest zaznaczona domyślnie, co spowoduje brak wyświetlenia informacji o problemach z certyfikatami dla tej domeny w przyszłości (problem ten był już poruszany w niniejszym opracowaniu).



Rysunek 27. Niepodpisany certyfikat: Internet Explorer

Ku sporemu zaskoczeniu autorów raportu, w przypadku przeglądarki Internet Explorer nie można było znaleźć jakichkolwiek informacji na temat błędów certyfikatów lub tuneli przy standardowych ustawieniach przeglądarki poza tymi udostępnionymi w standardowym interfejsie. Oczywiście, przy zagłębieniu się w ustawieniach, jest dostępna możliwość podglądu samego certyfikatu (co wymaga pewnej wiedzy technicznej), jednak zarówno na pierwszej stronie, jak i w sekcji „*Więcej informacji*", a także dalej – w pasku adresu w polu „*Błąd certyfikatu*" – nie znaleziono informacji o tym, jaki konkretnie błąd wygenerował omawiany (lub jakikolwiek inny) wyjątek. Wygląda na to, iż projektanci omawianej przeglądarki zapomnieli, lub intencjonalnie postanowili nie informować

użytkownika o dokładnej przyczynie jego problemów, co wydaje się bardzo krzywdzące dla użytkowników bardziej dociekliwych i zaawansowanych technicznie.



Rysunek 28. Niepodpisany certyfikat: Google Chrome

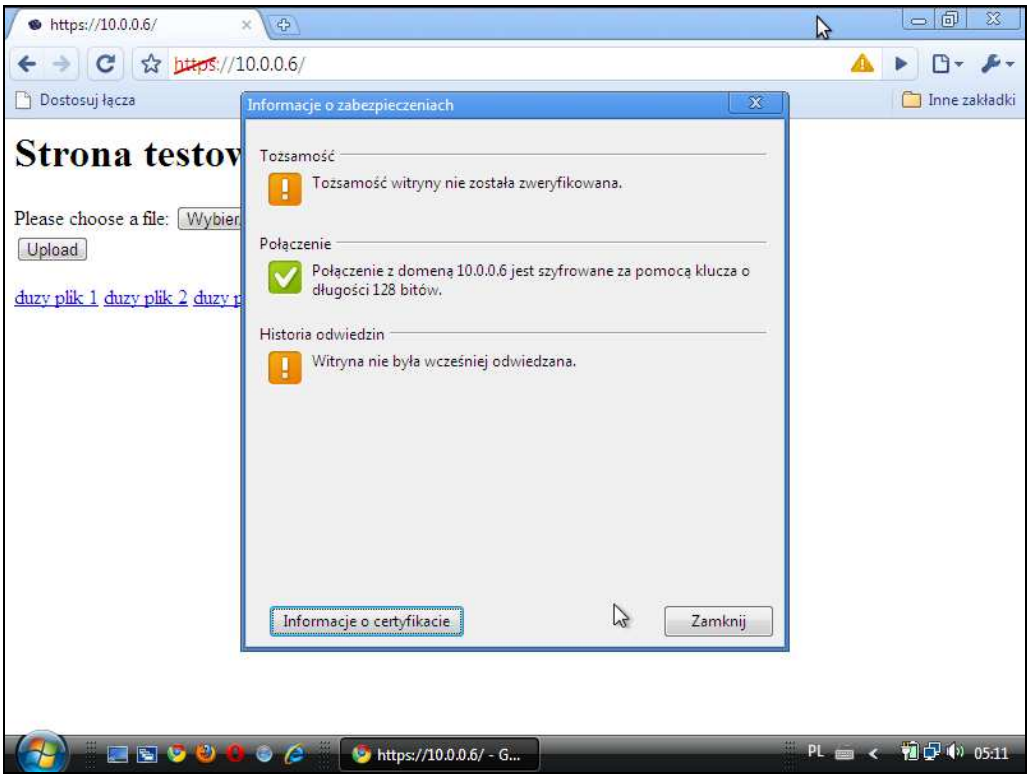
W przypadku przeglądarki Google Chrome, powyższy obraz pokazuje, iż projektanci przeglądarki prawdopodobnie wprowadzili system priorytetów w przypadku błędów tuneli/certyfikatów. Można tu także zakładać (również w oparciu o pozostałe testy), iż omawiana przeglądarka stosuje też grupowanie błędów, dla których przygotowano jednakowe opisy. Jak pokazuje Rysunek 28, opis jest dosyć długi i wyczerpujący, ale i wygodny, gdyż podpowiada użytkownikowi, co ten powinien wykonać w poszczególnych przypadkach.

Dużo słabszym elementem przeglądarki Google Chrome jest pokazanie użytkownikowi już przeglądającemu stronę, że nadal istnieje problem z jego bezpiecznym połączeniem. Okno informacyjne po kliknięciu ikonki w pasku adresu pokazuje kilka informacji, będących tylko statusami wybranymi przez producenta. Na szczęście, mimo iż nie jest to oczywiste dla standardowego użytkownika, projektanci nie zapomnieli o szczegółowych informacjach. Wyświetlane są one jednak dosyć specyficznie, co pokazuje niżej Rysunek 30.

Dodatkowo można oczywiście wykorzystać przycisk „*Informacje o certyfikacie*", jednak jest to tylko odgórny podgląd certyfikatu, przy którego użyciu do diagnozy problemów wymagana jest dużo szersza wiedza, niż zakładana dla standardowego użytkownika Internetu.

Na uwagę zasługuje tu także fakt, iż nie ma możliwości dodania tzw. „wyjątku" na stałe. Dość ciekawym pomysłem może okazać się zastąpienie dodawanego wyjątku statusem liczby odwiedzin

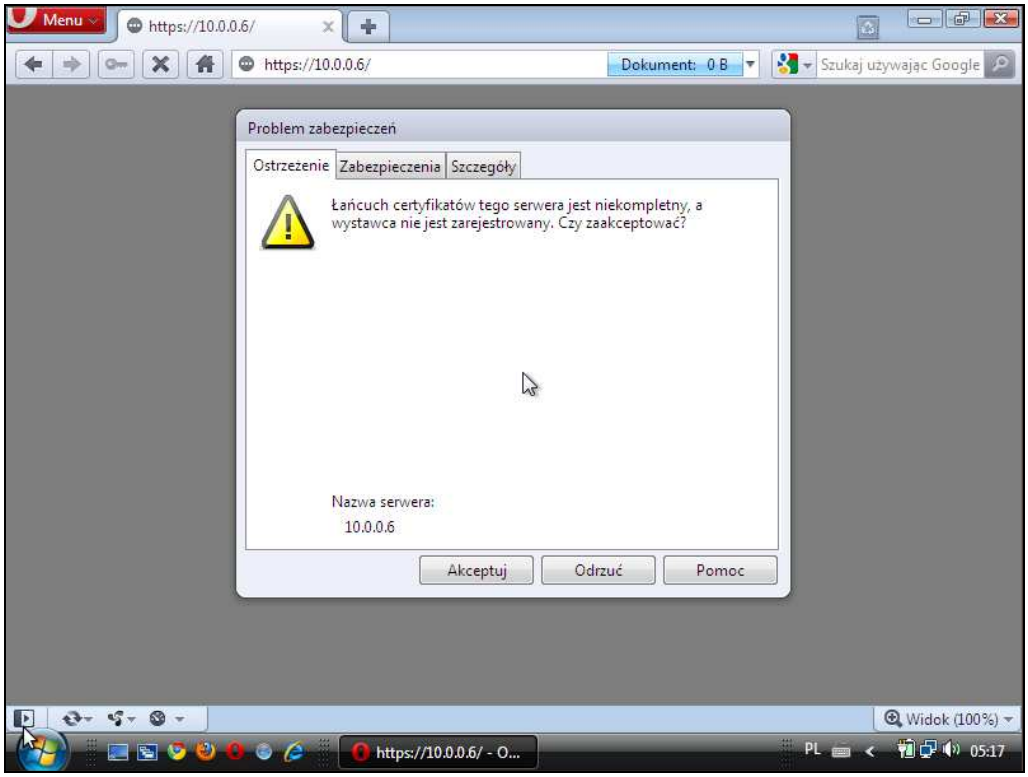
na danej podstronie, wraz z datą ostatniej wizyty. Pokazuje to dokładnie Rysunek 29 – jako sekcja „*Historia odwiedzin*".



Rysunek 29. Niepodpisany certyfikat: Google Chrome

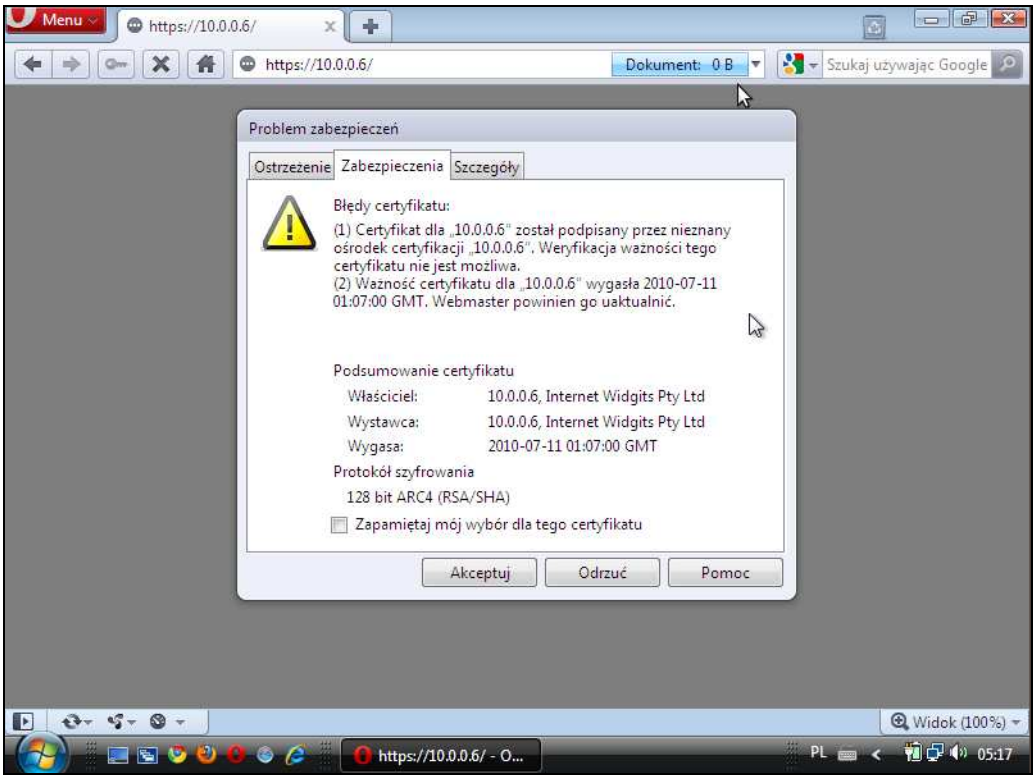
Rysunek 30. Niepodpisany certyfikat: „ukryte" informacje w Google Chrome

Bardzo interesującym podejściem producentów przeglądarki Google Chrome jest zamieszczenie ukrytej informacji o błędach konkretnego tunelu, pokazywanej w postaci prostej „chmurki" po najechaniu na trójkąt ostrzegawczy w pasku adresu. Rysunek 30 obrazuje, jak w prosty i czytelny sposób można zaprezentować problemy z szyfrowanym tunelem.



Rysunek 31. Niepodpisany certyfikat: Opera – krok 1

W przypadku niepodpisanego certyfikatu w przeglądarce Opera pokazywany jest prosty komunikat (choć ilość miejsca na formatce sugeruje, że w innych przypadkach może być to dużo dłuższa treść), zawierający jak najbardziej poprawne dane. Pytanie jest dosyć trywialne i niestety zachodzi tu możliwość odruchowego potwierdzenia błędnego certyfikatu, jednak – jak zaznaczano już w poprzednich rozdziałach niniejszego opracowania – certyfikat nie zostanie automatycznie dodany do zestawu zapamiętanych reguł. Jak pokazuje Rysunek 32, odznaczone jest bowiem domyślnie pole zapamiętywania reguł. Autorzy raportu uznali to za bardzo pozytywny aspekt zarządzania tego typu wyjątkami.

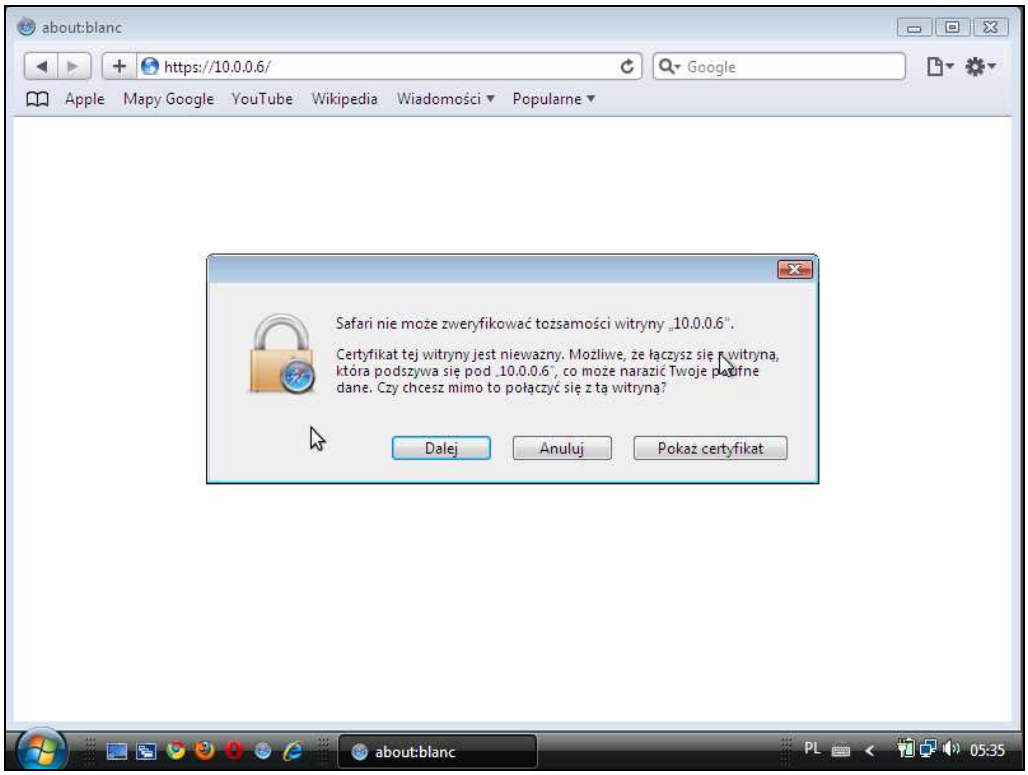


Rysunek 32. Niepodpisany certyfikat: Opera – krok 2

Rysunek 32 pokazuje dokładnie dane prezentowane użytkownikowi nieco bardziej zaawansowanemu. W prosty sposób przestawione są „Błędy certyfikatu", a pod nimi pozostałe istotne dla weryfikacji połączenia parametry, takie jak algorytm szyfrowania, wystawca czy czas wygasania samego certyfikatu. W subiektywnej ocenie autorów, takie rozmieszczenie informacji pozwala na bardzo szybkie i trafne rozpoznanie problemów z szyfrowanym połączeniem bez dodatkowych narzędzi, co z pewnością byłoby znacząco trudniejsze w przypadku innych przeglądarek.

Na formatce, która pojawiła się podczas nawiązywania połączenia zawierającego błędy, zauważyć można jeszcze jedną zakładkę „Szczegóły", pokazującą część pól z certyfikatu, jednak dla zaawansowanych użytkowników to z pewnością za mało informacji, a dla mniej zaawansowanych – są one raczej zbyteczne.

Ciekawostką jest tu także automatyczna zmiana aktywnej zakładki po czasowej bezczynności użytkownika, przenosząca go z powrotem bezpośrednio na pierwszą zakładkę „Ostrzeżenie", jakby przeglądarka chciała zaakcentować swoje wcześniej zadane pytanie „Czy zaakceptować?".



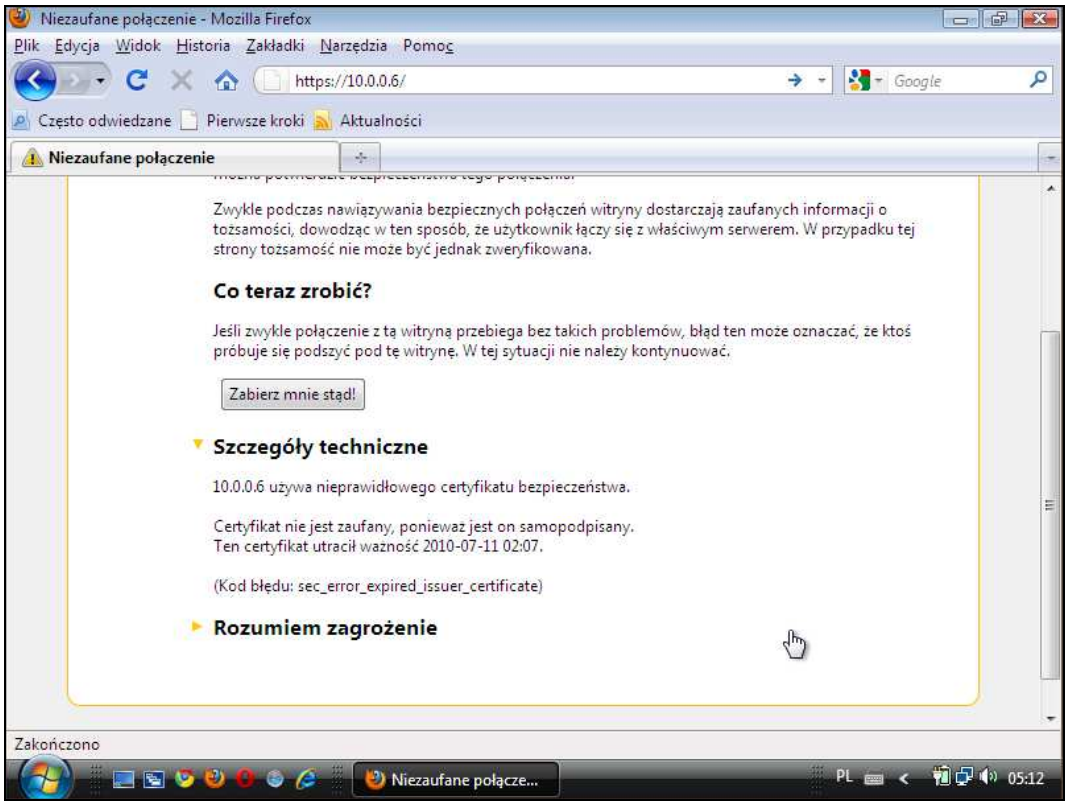
Rysunek 33. Niepodpisany certyfikat: Safari

Obsługa błędów w tunelach SSL/TLS w przeglądarce Apple Safari jest bardzo mocno ograniczona, co powoduje brak dostępu do szczegółowych informacji na temat certyfikatu lub parametrów tunelu w trakcie nawiązywania, a także trwania połączenia. Widać tu bardzo wyraźnie zarysowany trend firmy, dotyczący zdejmowania odpowiedzialności i wymagania wiedzy od użytkowników. W opinii autorów raportu jest to bardzo krzywdzące dla trochę bardziej zaawansowanych użytkowników i może spowodować ich odejście w stronę konkurencyjnych produktów.

2.4.5 Certyfikat z błędnym czasem życia

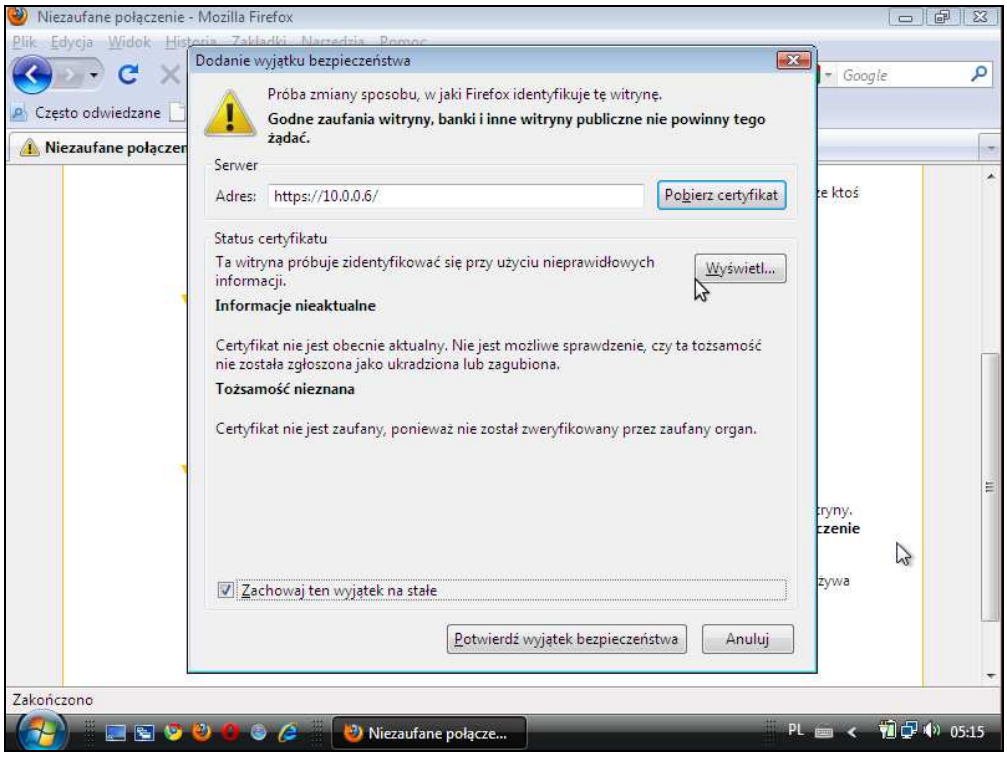
Stosunkowo często (nawet w dużych portalach lub bankach) przytrafia się, że administrator przeoczy czas życia certyfikatu (moment jego wygaśnięcia) i nie wygeneruje (wykupi) na czas następnego. W bardzo krótkim czasie dowiaduje się o tym, uzyskując informacje od klientów, którym wyświetliły się ostrzeżenia o wygaśnięciu certyfikatu. Zgodnie ze specyfikacją, certyfikaty do obsługi połączeń szyfrowanych zostają wydane na ściśle określony czas. W Internecie występuje aktualnie bardzo duża liczba domen, które zostały przejęte przez inne przedsiębiorstwa i przekwalifikowane. Łatwo sobie wyobrazić, że certyfikat bez ograniczeń czasowych byłby działającym certyfikatem także dla następnego (przejmującego) przedsiębiorstwa, co skutkować mogłoby skutecznymi atakami na klientów tejże domeny. W grę wchodziłaby w takim przypadku także możliwość unieważnienia certyfikatu, jednak nie będzie to w tym rozdziale omawiane.

Prowadzono testy zarówno dla certyfikatów, których okres ważności (czas życia) zakończył się, jak i dla takich, dla których się jeszcze nie rozpoczął. Nie wykryto jednak różnic w reakcjach testowanych przeglądarek dla tych obu przypadków.



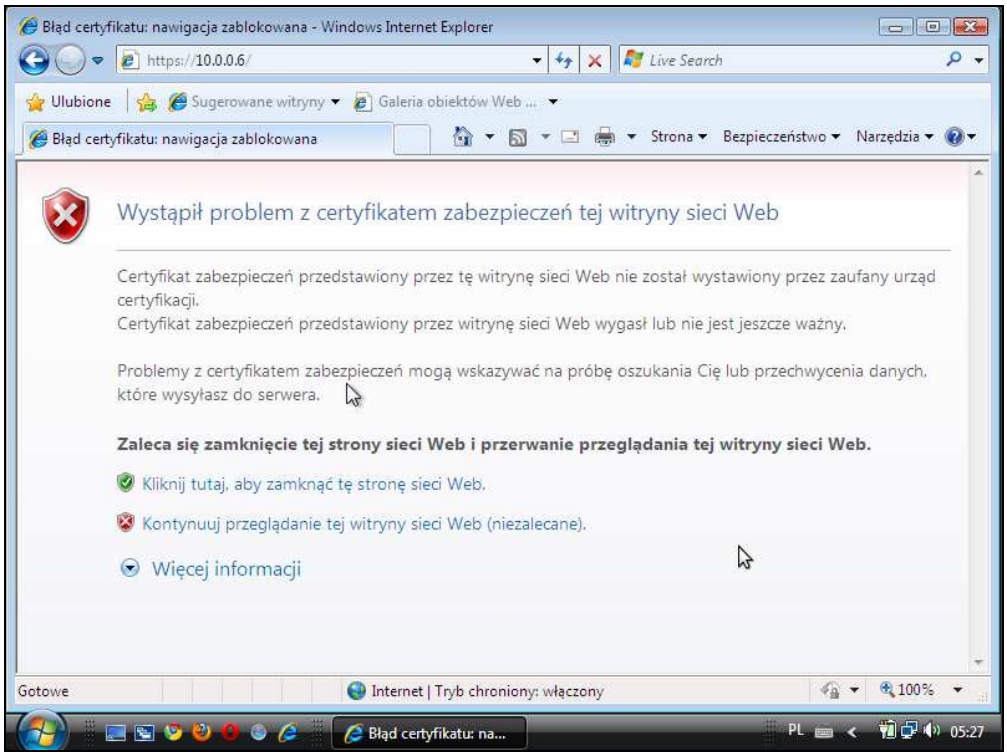
Rysunek 34. Certyfikat z błędnym czasem życia: Mozilla Firefox – krok 1

W przypadku przeglądarki Mozilla Firefox (Rysunek 34) można zauważyć, że w sekcji „Szczegóły techniczne" jasno pokazana jest informacja o utracie ważności certyfikatu. Zawarto również informację, kiedy się to stało. Bardzo miłym aspektem dla bardziej zaawansowanych użytkowników będzie tu z pewnością kod błędu, dzięki któremu można odszukać dokładne informacje na temat problemu w dokumentacji.



Rysunek 35. Certyfikat z błędnym czasem życia: Mozilla Firefox – krok 2

Także na kolejnym etapie dodawania wyjątku widoczne są informacje o problemie z czasem życia testowanego certyfikatu.

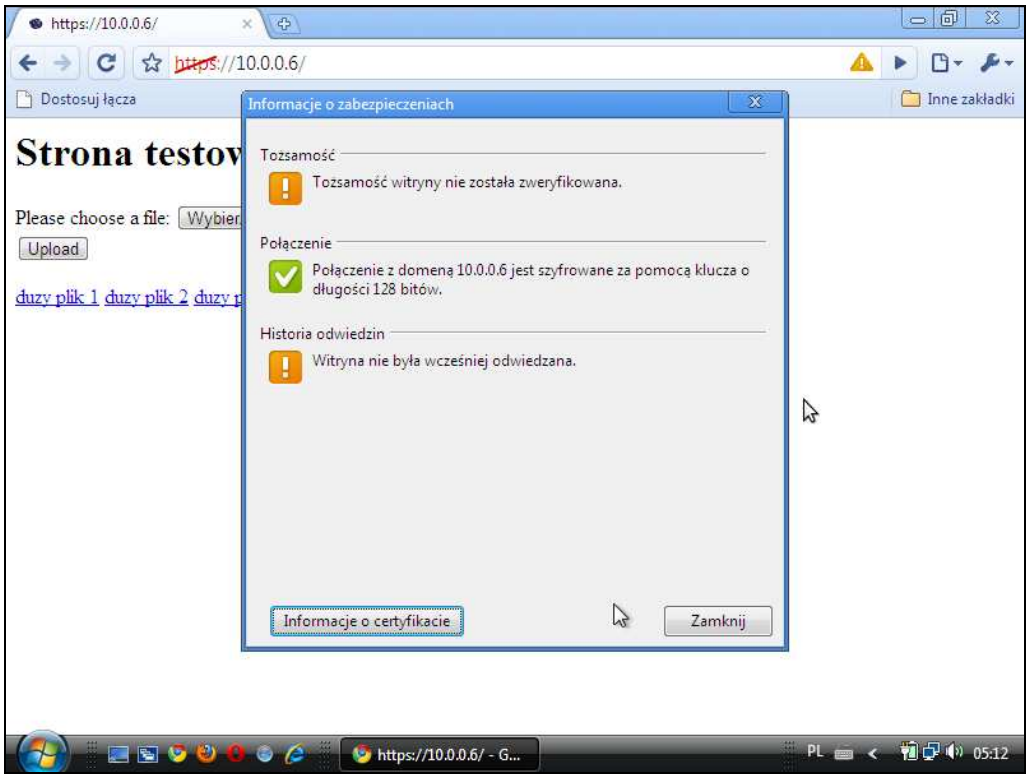


Rysunek 36. Certyfikat z błędnym czasem życia: Internet Explorer

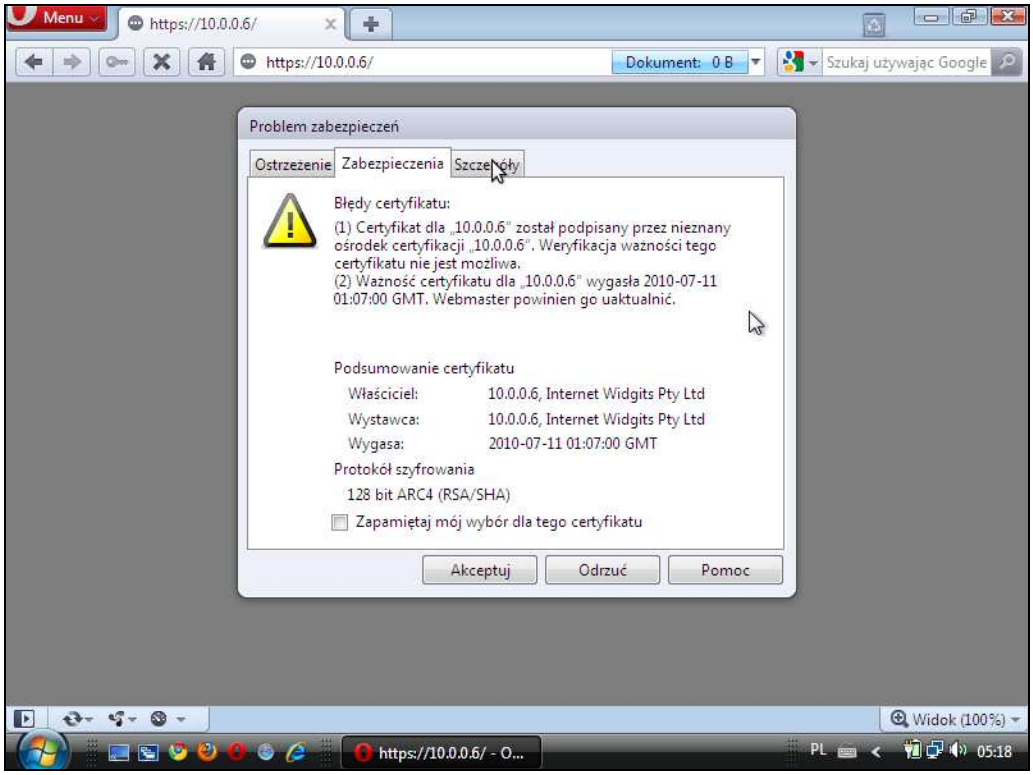
W przypadku przeglądarki Internet Explorer wszystkie przedstawione informacje widoczne są w pierwszym wyświetlanym oknie. Błąd wygenerowany przez certyfikat został sklasyfikowany jako błąd czasu (także), w związku z czym została wyświetlona informacja, iż certyfikat „[…] wygasł lub nie jest jeszcze ważny". Są to niestety jedyne informacje, które w prosty sposób można uzyskać od tej przeglądarki.

Rysunek 37. Certyfikat z błędnym czasem życia: Chrome

W przypadku przeglądarki Google Chrome pojawia się kilka miejsc, w których spodziewać się można uzyskania informacji na temat pojawiających się błędów. Pierwszym z nich jest wyświetlane okno „Informacje o zabezpieczeniach", gdzie dane o błędach są kategoryzowane (a czas życia certyfikatu nie jest widocznie sprawą najważniejszą). Kolejne – i bardziej interesujące – miejsca to formatka, dostępna po kliknięciu znaku ostrzegawczego trójkąta (Rysunek 38), oraz chmurka pojawiająca się po przesunięciu myszką na omawianą ikonę (Rysunek 37). Niestety, w pierwszym przypadku dostępnych jest bardzo mało informacji na temat błędów czasowych (możliwe, że i tu z powodu kategoryzacji zostały one uznane za zbędne).



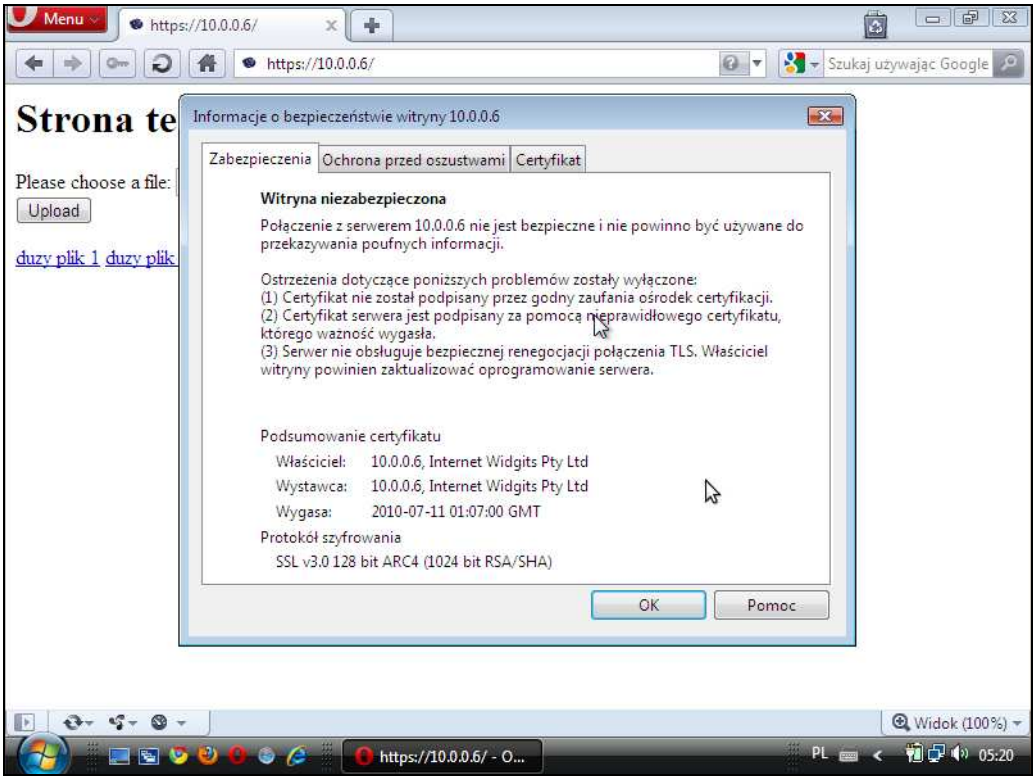
Rysunek 38. Certyfikat z błędnym czasem życia: Chrome – formatka 2



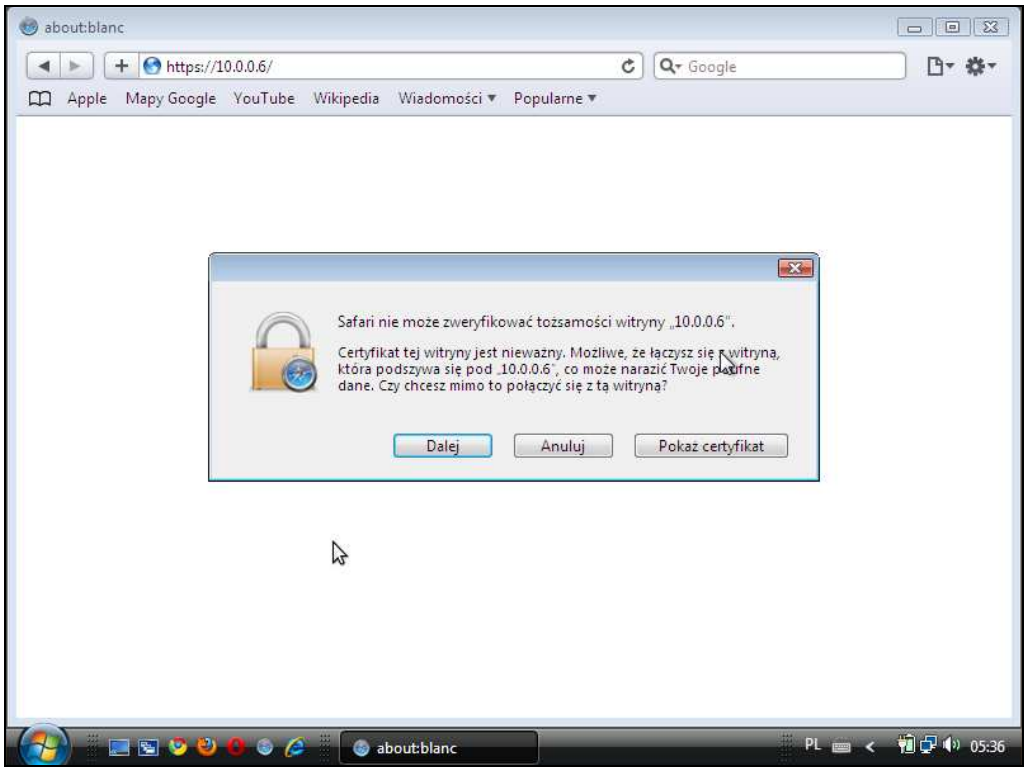
Rysunek 39. Certyfikat z błędnym czasem życia: Opera – krok 1

W przypadku przeglądarki Opera, jak już wiele razy wcześniej autorzy raportu podkreślali, istnieje możliwość uzyskania dużej ilości danych na różnym poziomie szczegółowości (zarówno dla osób

o węższej, jak i szerszej wiedzy technicznej). Pierwsze miejsce, w którym zamieszczono sporo interesujących informacji na temat błędów wygenerowanych przy nawiązywaniu połączenia, to wyświetlana przy inicjacji połączenia formatka z informacjami. Znaleźć tam można dokładne przyczyny błędów – „Certyfikat dla ... wygasł ...” oraz informacje, co powinno się w takim wypadku zrobić: „Webmaster powinien go uaktualnić”. Opisaną formatkę prezentuje Rysunek 39. Innym miejscem, gdzie informacje takie można uzyskać, jest formatka otwierająca się po kliknięciu ikonki szarego znaku zapytania „Informacje o bezpieczeństwie witryny”. Dokładnie sytuację tę przedstawia Rysunek 40. Wyraźnie wyliczono tam wszelkie błędy, jakie wystąpiły w przypadku zarówno certyfikatu, jak i samego połączenia. Oczywiście, tak jak i w przypadku większości przeglądarek, użytkownikowi udostępniono możliwość podejrzenia konkretnych pól certyfikatu (przy użyciu zakładki „Certyfikat”).



Rysunek 40. Certyfikat z błędnym czasem życia: Opera – formatka 2



Rysunek 41. Certyfikat z błędnym czasem życia: Safari

W przypadku ostatniej omawianej przeglądarki – Safari – niestety, uzyskać można bardzo mało informacji. Projektanci przeglądarki starali się uniknąć pracy związanej z implementacją wyjątków i rad dla działań użytkownika dla konkretnych błędów. Udostępniono jedynie przycisk „Pokaż certyfikat", otwierający standardową formatkę, zawierającą część podstawowych pól samego certyfikatu. W przypadku podstawowych błędów okazuje się jednak, że identyfikacja i rozwiązanie problemu zajmuje nawet doświadczonym użytkownikom sporo czasu. Formatkę sygnalizującą błąd w przypadku certyfikatu z błędnym czasem życia wskazuje Rysunek 41.

2.4.6 Zbyt słabe klucze RSA

Kolejne grupy testów dotyczyły reakcji przeglądarek na witryny, w których zastosowano certyfikaty wykorzystujące zbyt słabe klucze szyfrujące.

Zdarza się, niestety, że dostawcy usług wyposażają je w certyfikaty, do zabezpieczenia których wykorzystano słabe klucze. Latem 2008 roku Zespół Bezpieczeństwa PCSS prowadził badania nad łamaniem certyfikatów wygenerowanych przez zawierający podatność kryptograficzną generator liczb pseudolosowych OpenSSL w systemie Linux Debian (wynikiem badań było wystąpienie na konferencji SECURE 2008, a prezentację można pobrać spod adresu [8]). Podczas badań 64 874 certyfikatów stwierdzono, że 1460 z nich (ok. 2,25%, w tym zabezpieczające pocztę elektroniczną i usługi dostępne przez HTTPS) zawierało klucze prywatne o długości 512 bitów, już wtedy uznane za zbyt słabe.

Brak odpowiednio długiego klucza może oczywiście skutkować jego złamaniem w dostatecznie krótkim czasie, aby wykonanie ataku na poufność danych było realne nawet przy dysponowaniu względnie niewielkimi (dostępnymi dla pojedynczego użytkownika lub ich niewielkiej grupy, a nie np. instytucji, ośrodka naukowego czy armii) zasobami obliczeniowymi. Z kolei trudno wymagać np. od użytkownika domowego, aby znał aktualne zalecenia organizacji standaryzacyjnych odnośnie długości kluczy i sprawdzał pod tym kątem każdy certyfikat. Dlatego tak ważne jest, aby przeglądarka potrafiła rozpoznać niebezpieczeństwo i ostrzec użytkownika.

Dlaczego krótki klucz nie zapewnia wymaganego poziomu bezpieczeństwa?

Podstawowym elementem certyfikatu jest para kluczy, służących do obsługi operacji wykonywanych przez certyfikat (szyfrowanie, podpisywanie itp.). Współczesne zabezpieczenia kryptograficzne nie opierają się na tajności algorytmu (jest wręcz zalecane, aby ten był jawny i zbadany przez niezależne grono kryptologów), ale klucza.

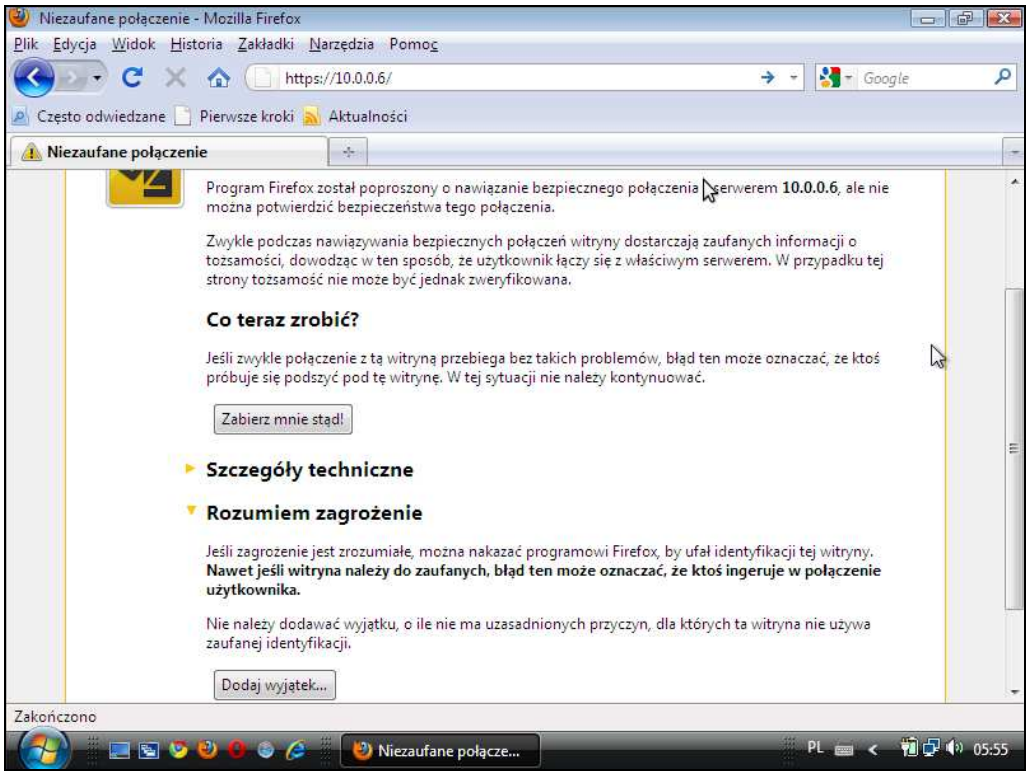
Teoretycznie każdy klucz można złamać, jeśli sprawdzi się wszystkie możliwe kombinacje jego wartości (tzw. atak siłowy lub w języku angielskim *brute force*). Gdy klucz jest odpowiednio krótki (czyli słaby), liczba kombinacji staje się na tyle niewielka, że jest realne ich sprawdzenie w rozsądnym czasie (np. kilka miesięcy) przy pomocy istniejących obecnie specjalnych rozwiązań sprzętowych, superkomputerów czy działających w skoordynowany sposób grup komputerów (gridów, klastrów). Jeżeli klucz jest za długi, jest to niemożliwe.

Zarówno możliwości komputerów, jak i stan badań nad łamaniem kluczy, stają się coraz bardziej zaawansowane. Dlatego twórcy rozwiązań bezpieczeństwa (a także np. administratorzy maszyn i projektanci oprogramowania użytkowego) muszą pozostawać na bieżąco z aktualnymi zaleceniami co do długości kluczy kryptograficznych, wydawanymi przez niezależne organizacje standaryzacyjne, takie jak National Institute of Standards and Technology (NIST).

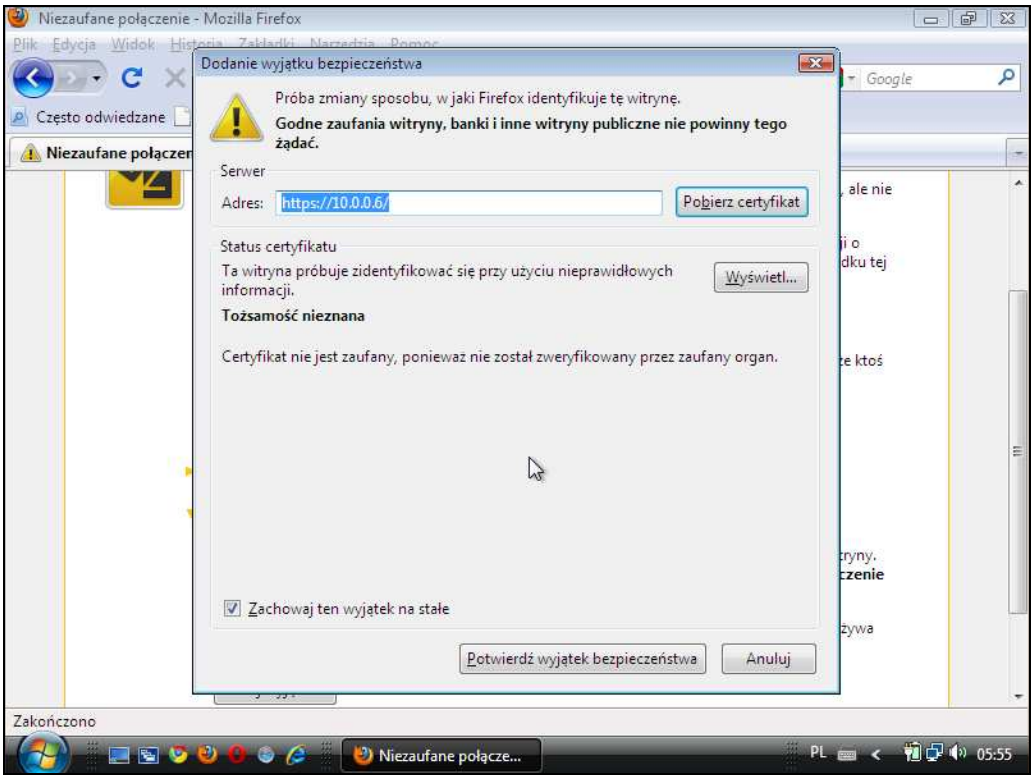
Wykonano kilka grup testów. W pierwszym rzędzie testowano certyfikaty z kluczem RSA, którego długość wynosiła 384 bity (jest to najmniejsza długość, jaką można obecnie zdefiniować przy wykorzystaniu standardowych poleceń biblioteki OpenSSL). Kolejną sprawdzoną wartością było 512 bitów (jako wielkość standardowa, choć już dziś nierekomendowana). Wreszcie zweryfikowano również, jako niestandardową, długość klucza 511 bitów.

2.4.6.1 Klucz RSA o długości 384 bity

Przy wykorzystaniu certyfikatu serwera o bardzo małej długości klucza (384-bitowego) zauważyć można, że przeglądarka Mozilla Firefox nie wykrywa problemu podczas próby inicjacji połączenia. Może to wynikać z faktu, iż informacje tego typu (długość klucza certyfikatu) wcale nie są w początkowym stadium połączenia weryfikowane. Pokazują to dokładnie poniższe rysunki.



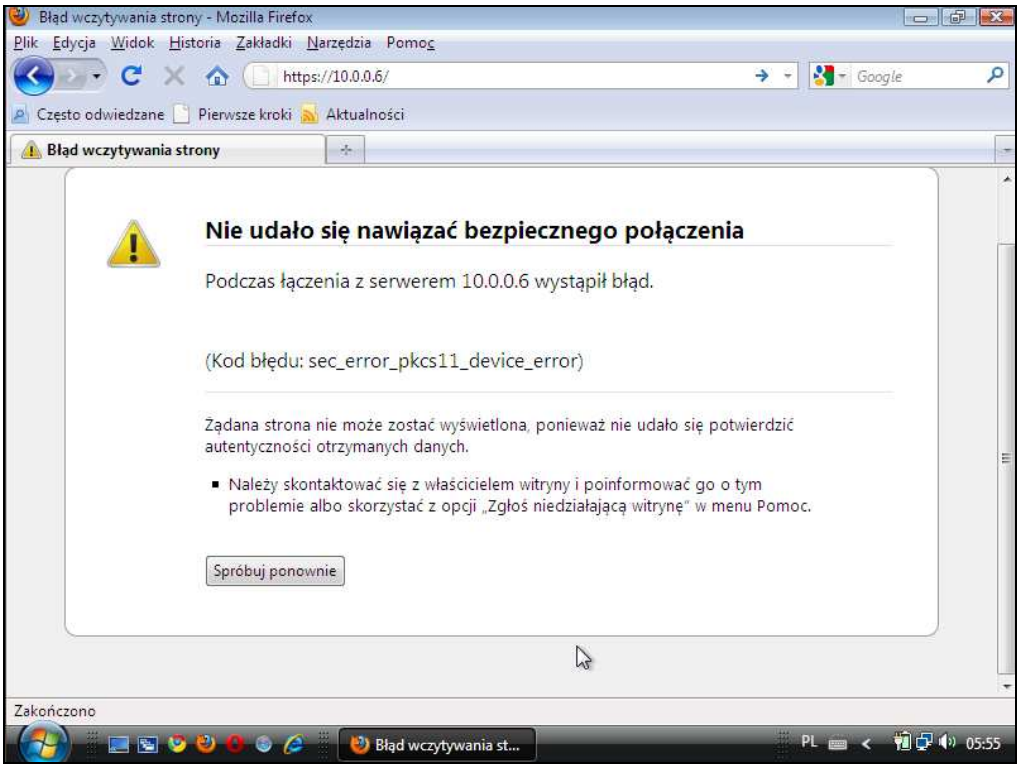
Rysunek 42. Klucz RSA o długości 384 bity: Mozilla Firefox – krok 1



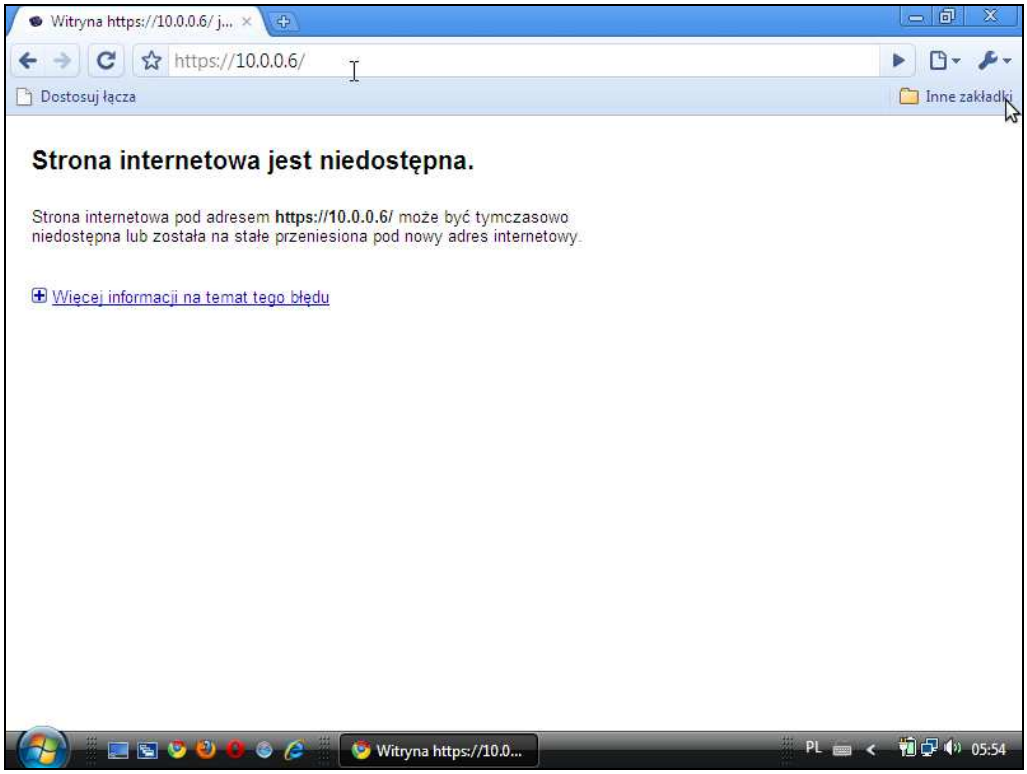
Rysunek 43. Klucz RSA o długości 384 bity: Mozilla Firefox – krok 2

Jak pokazuje Rysunek 44 – weryfikacja ta w praktyce następuje dopiero w momencie próby nawiązania połączenia. Przeglądarka zwraca wtedy błąd „sec_error_pkcs11_device_error", który jednak nie wskazuje bezpośrednio na fakt, że przeglądarka otrzymała za krótki klucz serwera.

Można tu po części domniemywać, iż przeglądarka Mozilla Firefox nie posiada poprawnie zbudowanego bloku wyjątków, mogących przechwytywać tego typu zdarzenia. Dla wnikliwych czytelników może to być punkt zaczepienia przy dalszych atakach na samą aplikację przeglądarki.

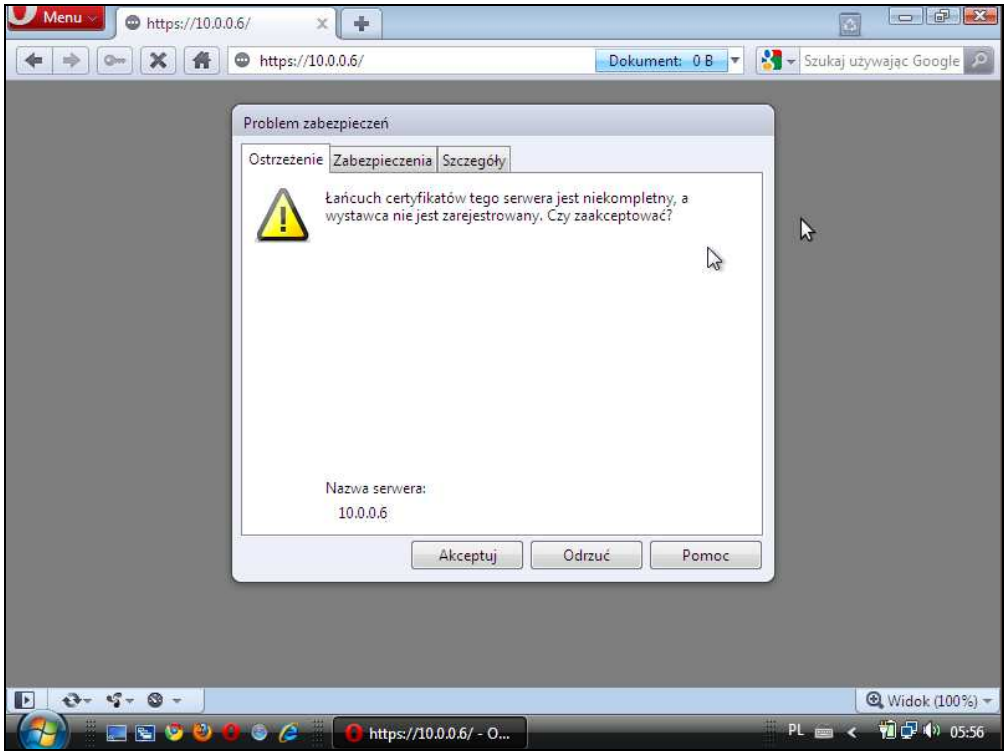


Rysunek 44. Klucz RSA o długości 384 bity: Mozilla Firefox – krok 3

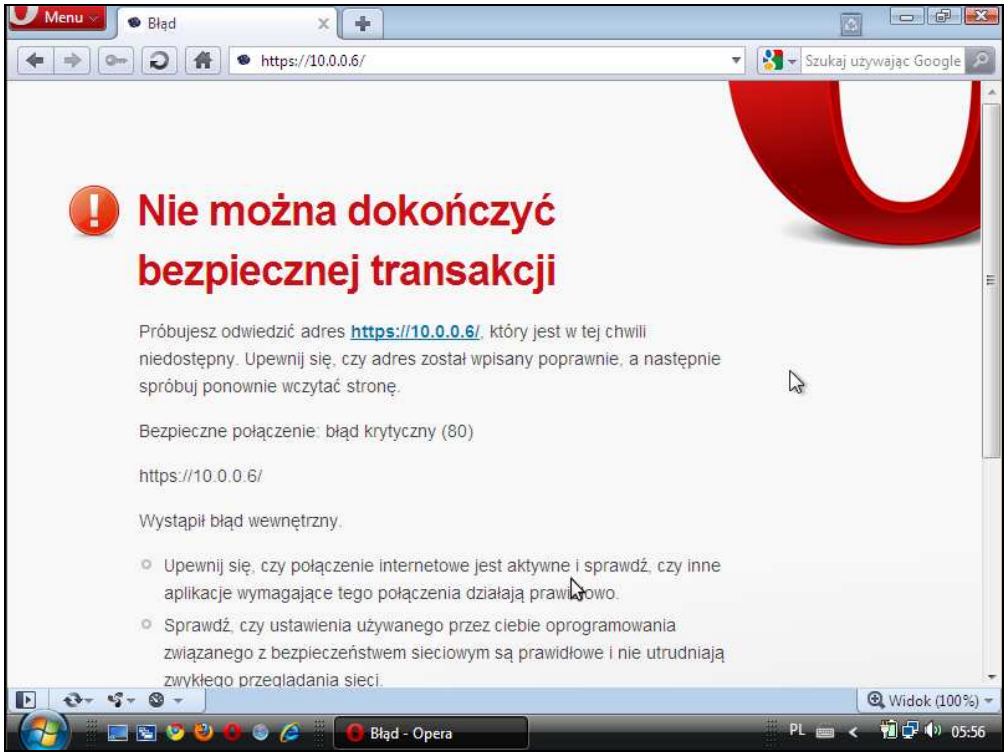


Rysunek 45. Klucz RSA o długości 384 bity: Chrome

W przypadku przeglądarki Google Chrome połączenie jest zrywane na tyle wcześnie, że sama przeglądarka interpretuje błędy w połączeniach jako niedostępność strony. Prawdopodobnie także i tu nie wzięto pod uwagę możliwości obsługi tego typu wyjątków, jednak jest to stosunkowo daleko idący wniosek w przypadku tak małej ilości udostępnionych użytkownikowi danych.

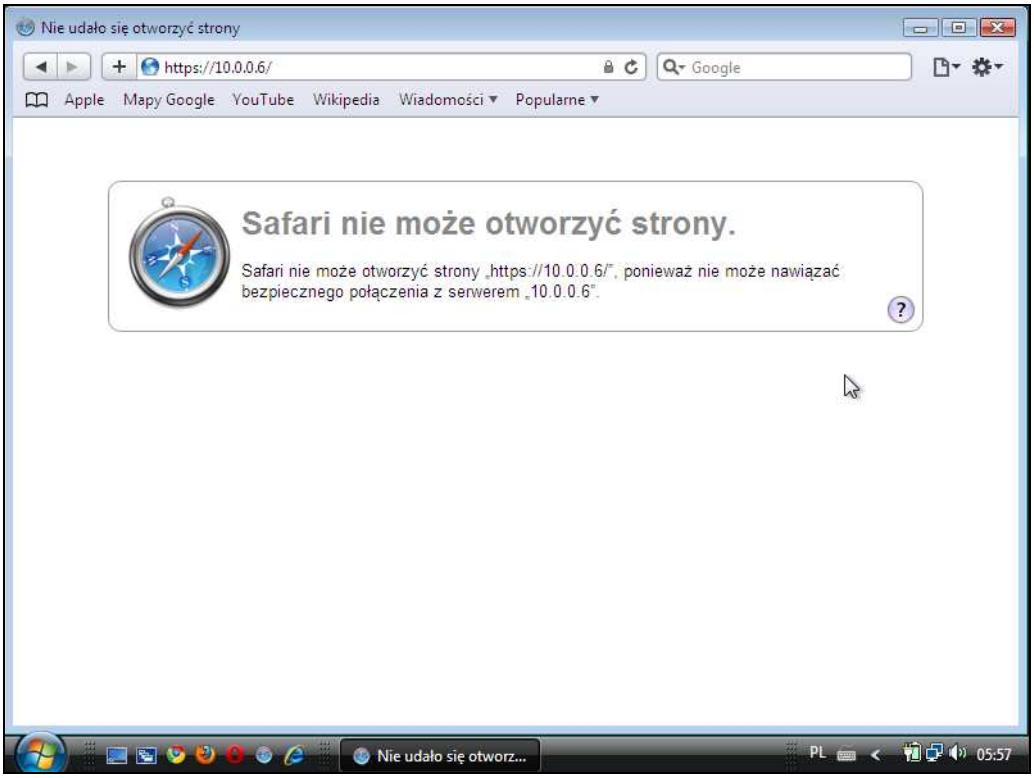


Rysunek 46. Klucz RSA o długości 384 bity: Opera – krok 1



Rysunek 47. Klucz RSA o długości 384 bity: Opera – krok 2

Przeglądarka Opera zachowuje się podobnie do programu Mozilla Firefox, czyli z początku nie rozpoznaje zagrożenia (za krótki klucz), a później z powodu „błędu krytycznego" w obsłudze wyjątków – nie serwuje zawartości strony bez podania sensownego uzasadnienia takowego stanu rzeczy. Pokazują to dokładnie Rysunek 46 oraz Rysunek 47.

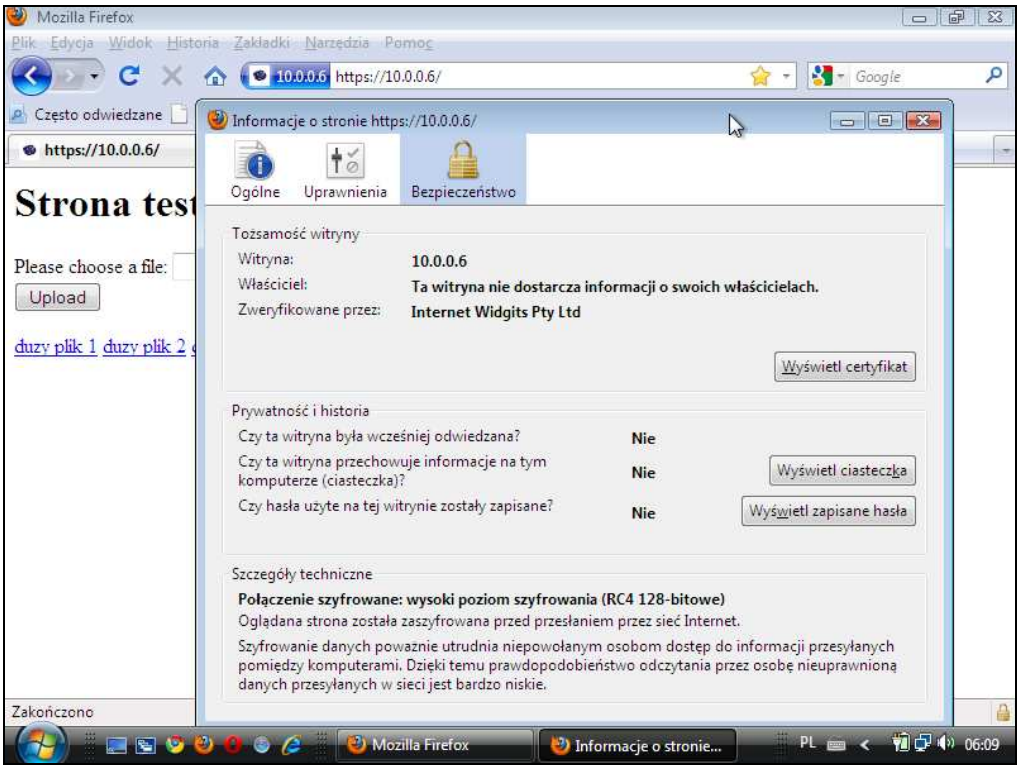


Rysunek 48. Klucz RSA o długości 384 bity: Safari

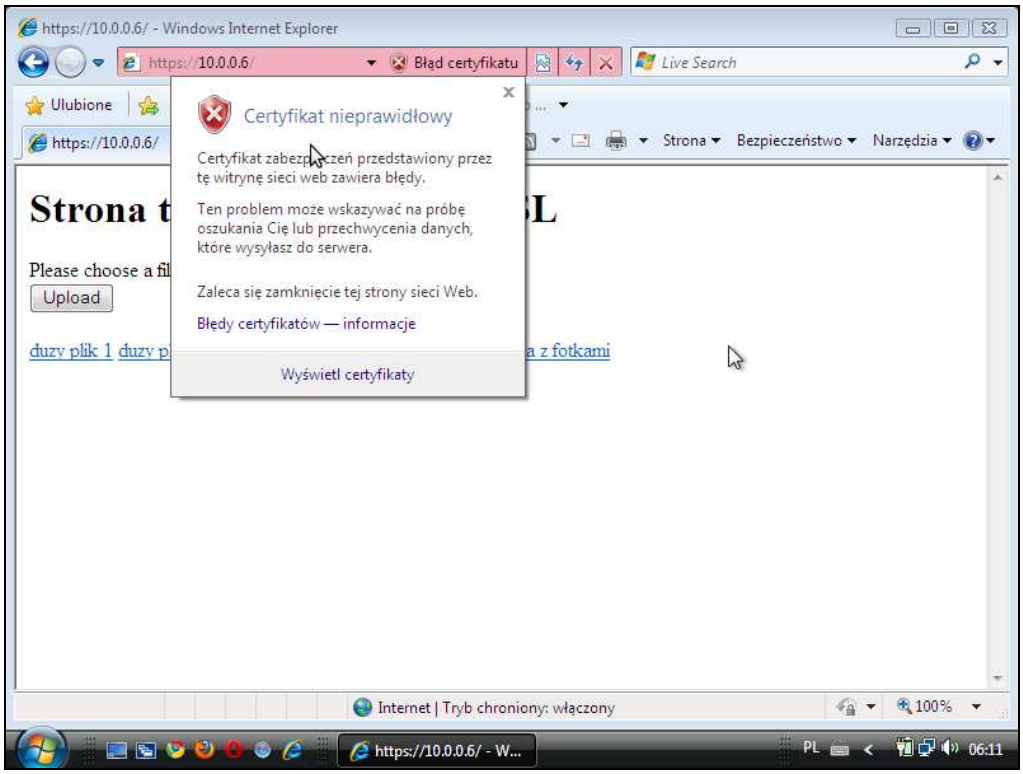
Safari, przyzwyczajając już użytkownika do minimalistycznego podejścia do udostępnianych mu komunikatów, stwierdza jedynie, iż nie może nawiązać bezpiecznego połączenia. Wszelkie dostępne przyciski także nie pozwalają na uzyskanie dodatkowych informacji o problemach mogących powodować błąd.

2.4.6.2 Klucz RSA o długości 512 bitów

Testy z wykorzystaniem certyfikatu serwera zawierającego klucz o długości 512 bitów miały na celu weryfikację, czy informacje o kluczach tej długości (standardowej, ale już od pewnego czasu niezalecanej) są wyświetlane w jakikolwiek specyficzny sposób. Okazuje się, że w ogólności w przeglądarkach, tak jak w widocznym poniżej zrzucie ekranowym formatki z programu Mozilla Firefox (Rysunek 49), brakuje przekazanej w prosty sposób informacji, jakiej długości jest sam klucz. Oczywistym jest, iż nie ma tu już problemów z odpowiednim wyświetleniem strony przy pomocy tunelu opartego na kluczu 512-bitowym (takich, jak napotkanych w przypadku klucza długości 384 bitów).

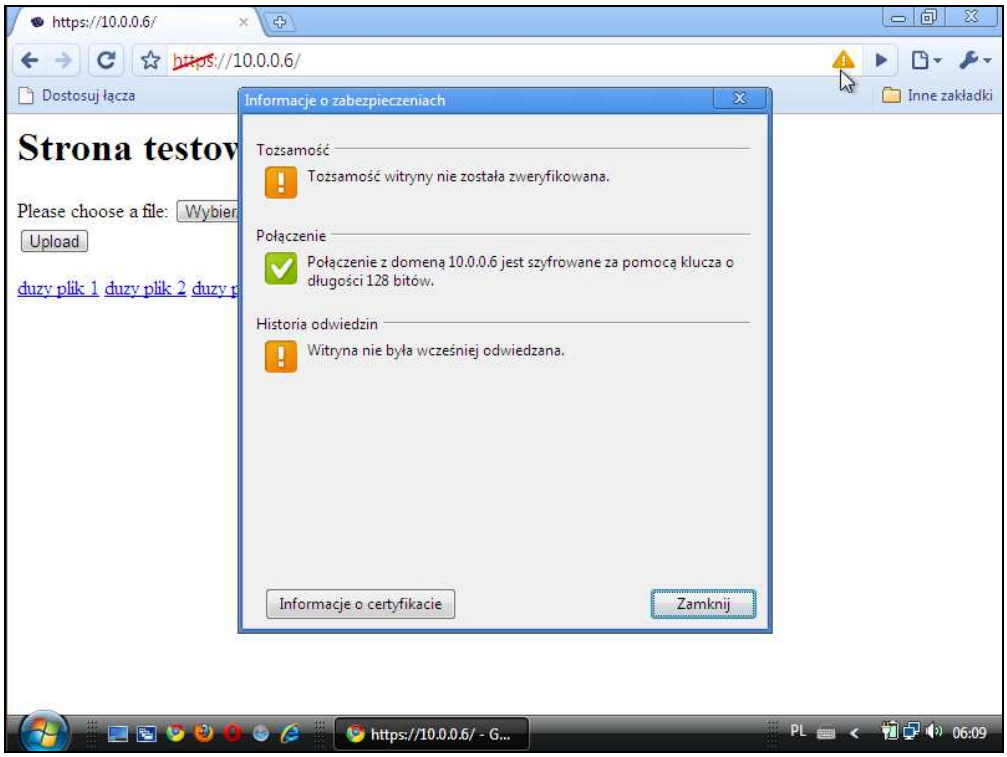


Rysunek 49. Klucz RSA o długości 512 bitów – Mozilla Firefox



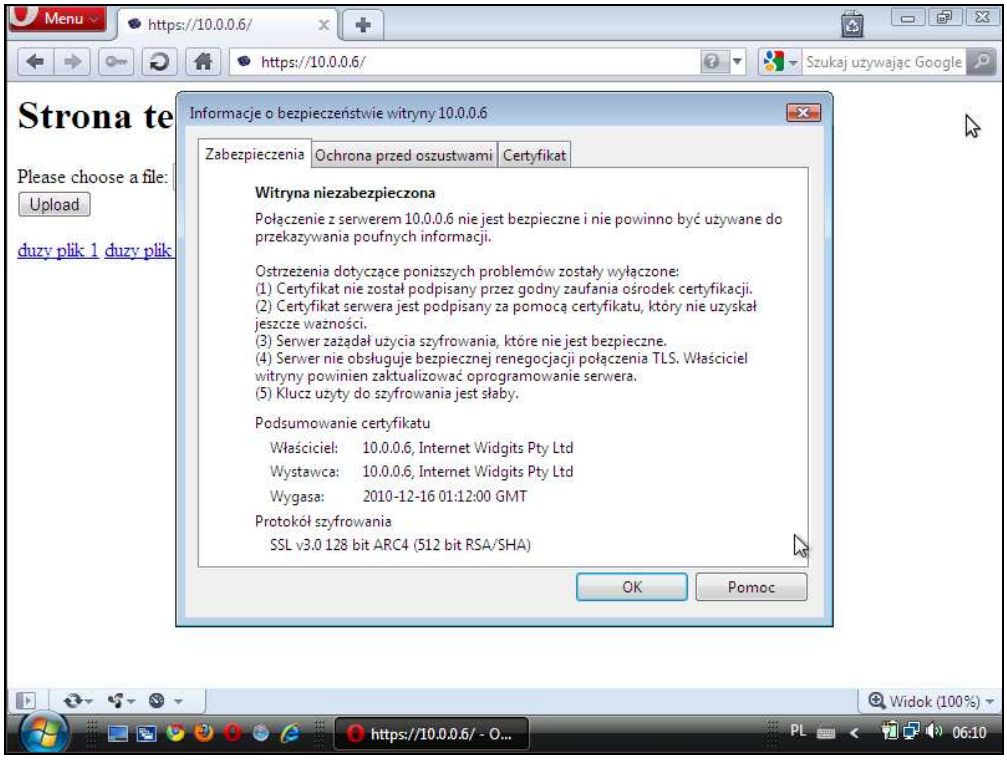
Rysunek 50. Klucz RSA o długości 512 bitów – Internet Explorer

Program Internet Explorer niestety także nie przekazuje w odpowiedni – zdaniem autorów niniejszego raportu – sposób informacji na temat kluczy (oraz ich długości).



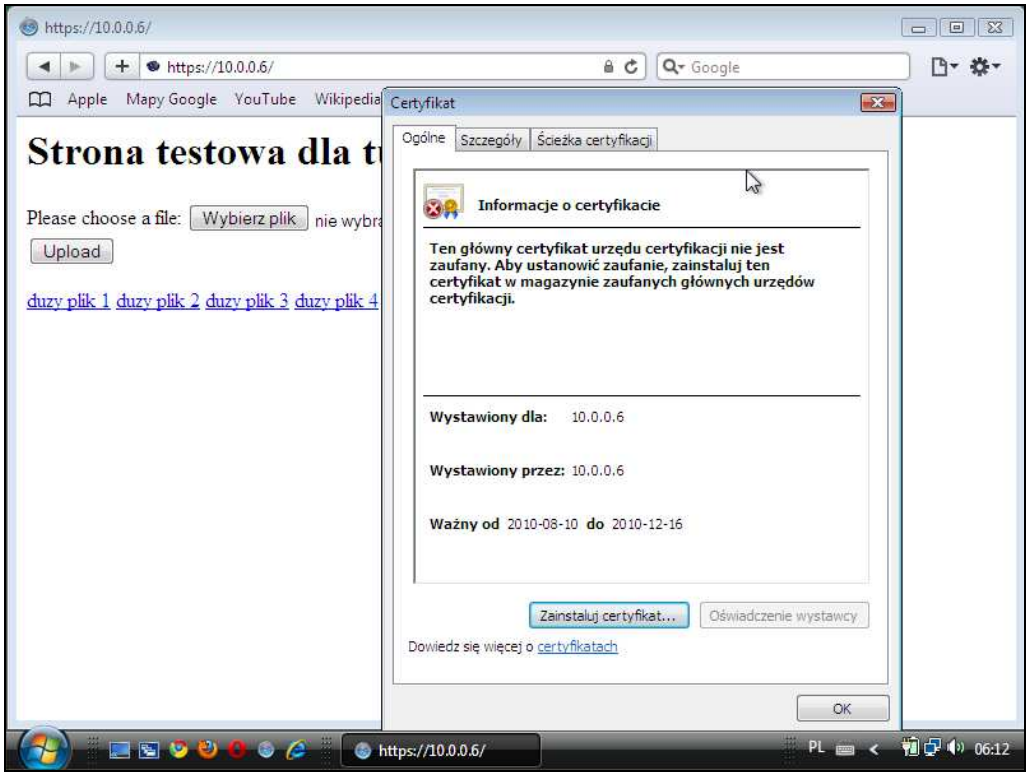
Rysunek 51. Klucz RSA o długości 512 bitów – Chrome

Podobnie jak i w poprzednich przypadkach – brak prosto podanej informacji na temat długości klucza. Nie są w żaden szczególny sposób traktowane połączenia nawiązywane przy pomocy certyfikatów ze słabym kluczem.



Rysunek 52. Klucz RSA o długości 512 bitów – Opera

Jak widać na rysunku powyżej – zgoła inaczej (najlepiej, zdaniem autorów raportu!) prezentuje się zestaw informacji wyświetlanych przez przeglądarkę Opera. Na dole formatki w sekcji „Protokół szyfrowania" zauważyć można informacje zarówno o wykorzystywanym protokole (SSLv3), algorytmie szyfrowania przesyłanych danych (ARC4) jak i długości klucza wraz z zestawem szyfru do generacji podpisu i funkcji skrótu (512 bit RSA/SHA). Jak już wielokrotnie wcześniej podkreślano – projektanci programu Opera położyli na interakcję z użytkownikiem (czy to doświadczonym, czy laikiem) największy nacisk i wkład pracy.

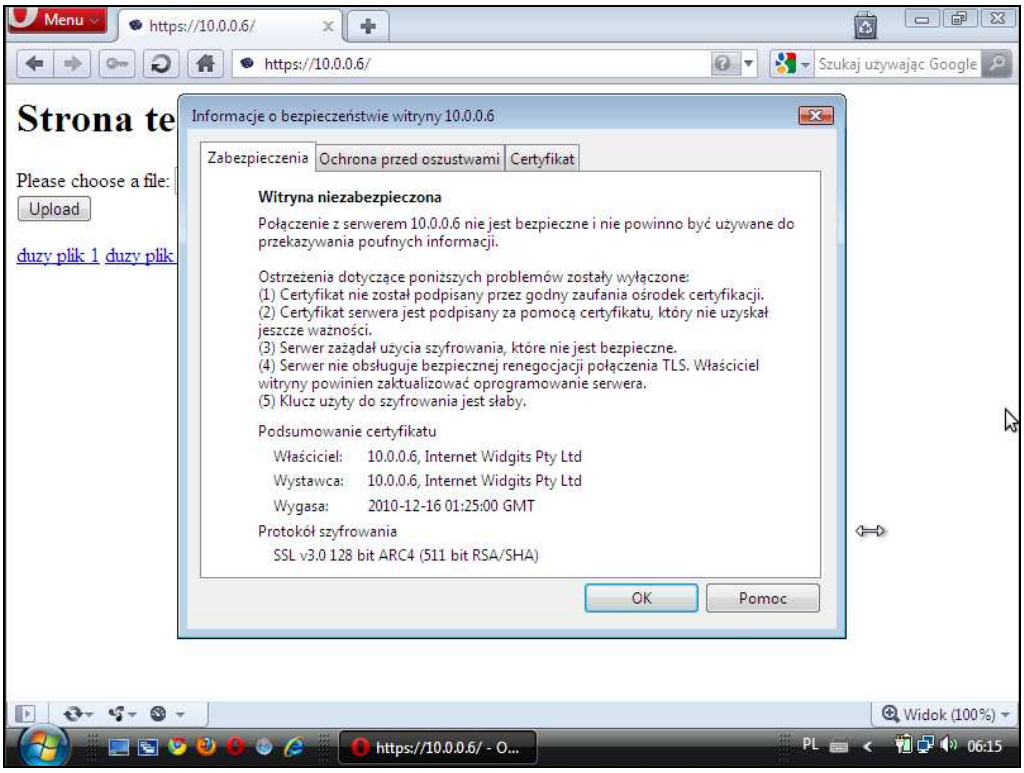


Rysunek 53. Klucz RSA o długości 512 bitów – Safari

W przypadku przeglądarki Safari nie zaprezentowano jakichkolwiek dodatkowych informacji, a także nie udostępniono użytkownikowi informacji na temat długości klucza w certyfikacie (pomijając formatkę informacji o certyfikacie, której intuicyjne użycie jest z pewnością domeną jedynie bardziej doświadczonych użytkowników).

2.4.6.3 Klucz RSA o długości 511 bitów

W przypadku kluczy o niestandardowej długości, autorzy raportu chcieli zweryfikować, czy ich obsługa jest także w odpowiedni sposób zaimplementowana. Ku ich zaskoczeniu okazało się, że zarówno krótsze, jak i dłuższe klucze o niestandardowej długości były przez przeglądarkę poprawnie obsługiwane. Niestety, tylko i wyłącznie przeglądarka Opera pokazuje to otwarcie na swoich formatkach, co przestawia Rysunek 54.



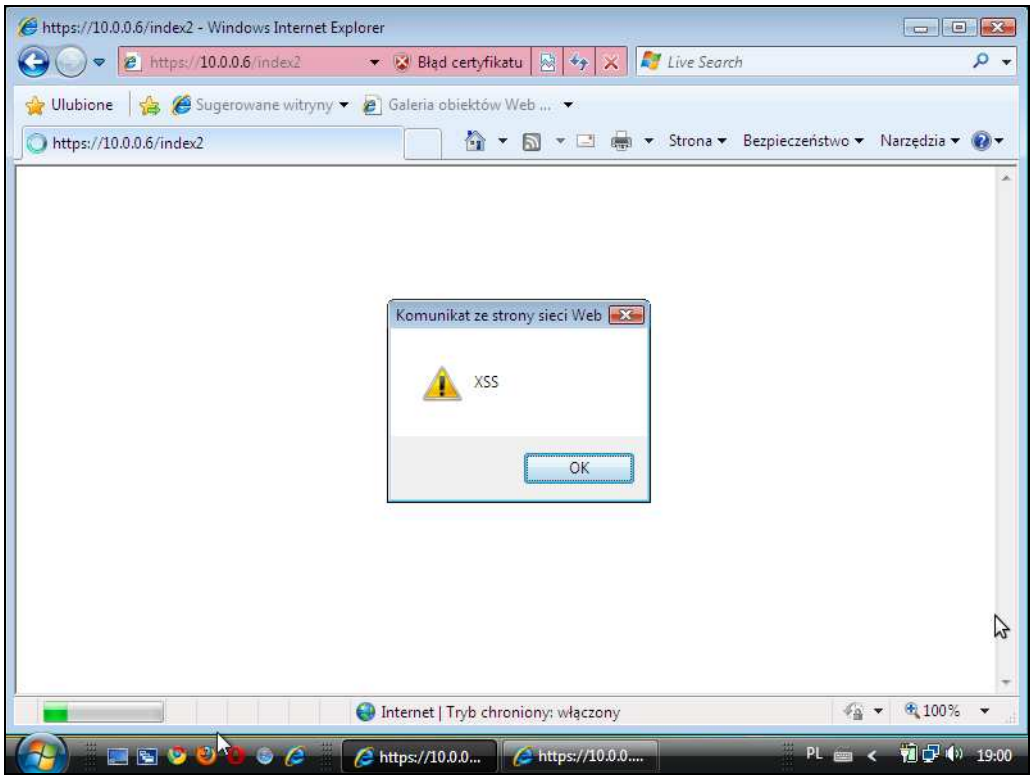
Rysunek 54. Klucz RSA o długości 511 bitów – Opera

2.4.7 Informacje o zerwaniu tunelu

Stosunkowo ważnym aspektem działania tuneli jest informacja na temat ich zerwania. Podczas testów autorzy wykonali dwie podstawowe symulacje zerwania połączenia przy założeniu, iż nie zostanie w ciągu ich trwania przekroczony czas życia połączenia (*keep alive*). Symulacje te to: zerwanie fizyczne połączenia oraz restart serwera. Obie zostały przeprowadzone podczas trwającej transmisji danych, a więc w trakcie pełnej aktywności tunelu. Otrzymane wyniki opisane zostały poniżej.

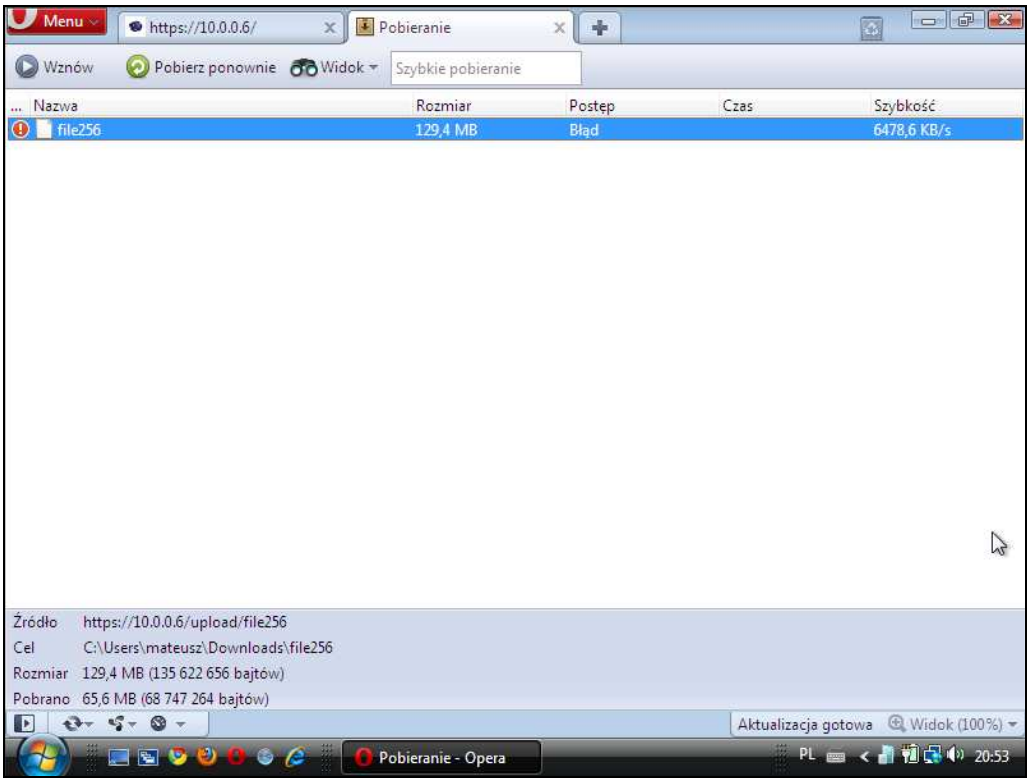
W przypadku fizycznego zerwania połączenia wszystkie z przeglądarek zachowały się niemal identycznie, czyli w czasie utrzymywania połączenia oczekiwały na ponowne jego ustanowienie i wznawiały działanie (uruchamiano ponowne ściąganie zasobów) od momentu, w którym pobieranie zostało przerwane.

Według audytorów należy tu jeszcze zaznaczyć jedną dość krytyczną lukę, mimo iż nie jest ona zupełnie związana z funkcjonowaniem tuneli SSL/TLS. Otóż, zgodnie ze standardowymi ustawieniami przeglądarki Internet Explorer, program po kliknięciu na link do pliku bez rozszerzenia – stara się ten plik pobrać i, bezpośrednio po tym, wyświetlić na ekranie. Oczywistym są tu problemy, które mogą być następstwem podstawienia spreparowanych stron, zawierających plik ze złośliwą zawartością. Przykład oczywistego problemu pokazuje tu Rysunek 55.



Rysunek 55. Domyślne otwieranie przez przeglądarkę Internet Explorer plików nieposiadających rozszerzenia

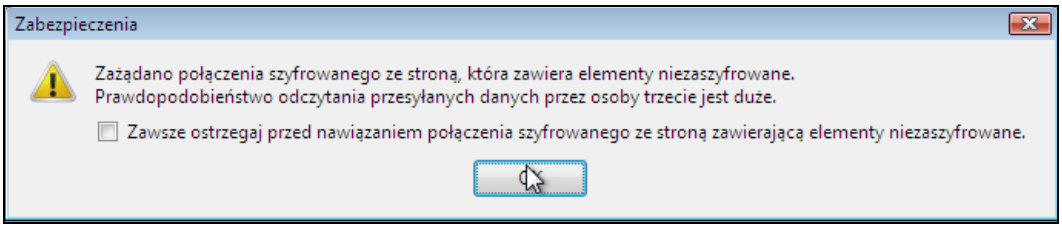
Drugim testowanym sposobem oddziaływania na tunel było jego zerwanie przez dokonanie restartu serwera WWW. Zgodnie z procedurą restartu – wszystkie połączenia zostają wtedy zakańczane, a ich inicjacja musi zostać rozpoczęta ponownie. Ku zdziwieniu autorów raportu – cztery z pięciu przeglądarek dokonały wprawdzie zapisu przesyłanego pliku o długości odpowiadającej ilości danych, które udało się pobrać, ale nie poinformowały użytkownika, że nie powiodło się pobranie całej zawartości pliku. Brak informacji o tego typu problemie jest z pewnością bardzo uciążliwy w przeglądarkach takich jak: Mozilla Firefox, Internet Explorer, Google Chrome oraz Safari. Jedynie w przypadku przeglądarki Opera nastąpiło jasne stwierdzenie wystąpienia błędu w trakcie przesyłania zawartości oraz prezentacja informacji na temat tego błędu (co przedstawia Rysunek 56).



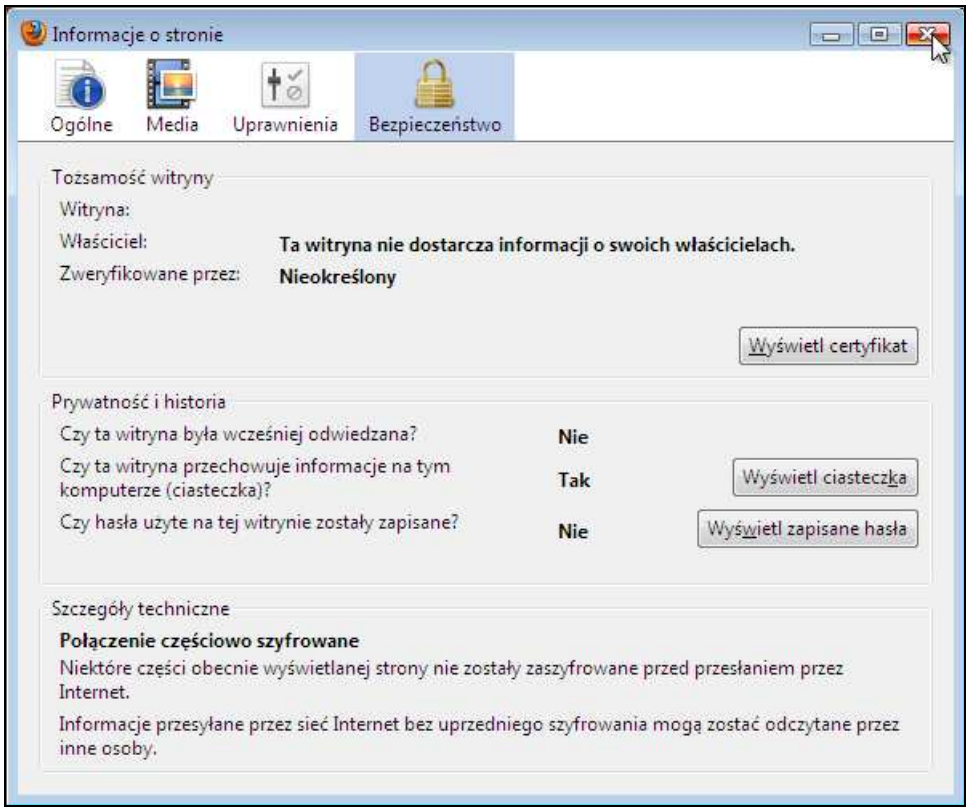
Rysunek 56. Informacja o błędzie podczas pobierania pliku – Opera

2.4.8 Przesyłanie danych szyfrowanych i nieszyfrowanych na jednej stronie

W przypadku przesyłania danych szyfrowanych i nieszyfrowanych na jednej stronie internetowej w przypadku dwóch przeglądarek nie zaobserwowano generowania ostrzeżeń, a tym bardziej przekazania jakichkolwiek dodatkowych informacji użytkownikowi (Opera oraz Safari). Na szczęście pozostałe testowane przeglądarki sygnalizują, gdy część zawartości strony przesyłana jest w tunelu szyfrowanym, a pozostała część – otwartym tekstem.

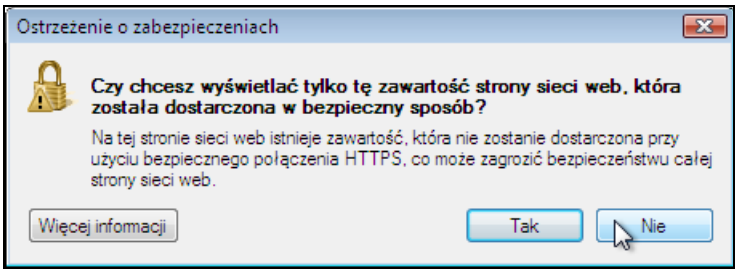


Rysunek 57. Komunikat ostrzegawczy – Mozilla Firefox



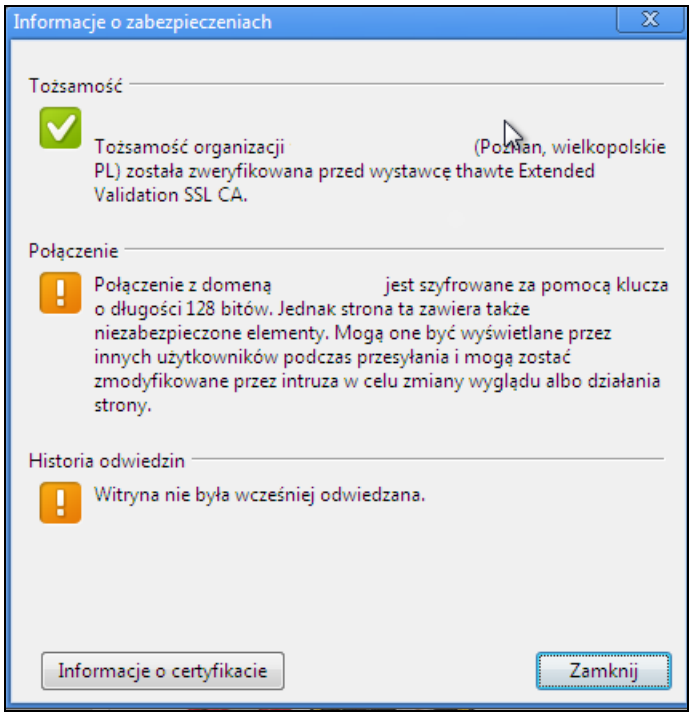
Rysunek 58. Mozilla Firefox – informacja o częściowym szyfrowaniu strony

Jak pokazano powyżej (Rysunek 57, Rysunek 58), producenci przeglądarki Mozilla Firefox zadbali o poprawne poinformowanie użytkownika, iż nie wszystkie jego dane przesyłane są w szyfrowanym tunelu. Jedynym problemem, na który warto zwrócić uwagę jest fakt, iż domyślne ustawienia komunikatu zakładają odznaczenie pola wyboru „Zawsze ostrzegaj przed nawiązaniem połączenia szyfrowanego ze stroną zawierającą elementy nieszyfrowane", co samo w sobie jest negacją wyjątku. Paradoksalnie – jeśli użytkownik wykona akcję domyślną, a więc nie zaznaczy odpowiedniego pola – nie zostanie powiadomiony o tego typu problemie następnym razem, co może narazić go na ataki.



Rysunek 59. Internet Explorer – informacja o częściowym szyfrowaniu strony

Bardzo dobrze z problemem prezentacji informacji na temat niepełnego szyfrowania danych na stronie poradził sobie program Internet Explorer. Komunikat dla użytkownika jest prosty i czytelny, dzięki czemu użytkownik w prosty i świadomy sposób może podjąć działania na rzecz zabezpieczenia swoich danych.



Rysunek 60. Google Chrome – informacja o częściowym szyfrowaniu strony

W przypadku przeglądarki Google Chrome informacja na temat braku kompleksowego zabezpieczenia danych przesyłanych przez tunel SSL/TLS jest zawarta na formatce informacji dotyczących certyfikatu w sekcji Połączenie. Informacja jest bardzo wyczerpująca, jednak już na etapie działania skryptów sniffujących w sieci między klientem a serwerem – jest już zupełnie bez znaczenia.

3. Podsumowanie

Raport ten jest jedynie wstępem do szerzej zakrojonych badań nad przeglądarkami internetowymi. Stanowi tylko wycinek analizy ich funkcjonalności i bezpieczeństwa, w szczególności aspektów ważnych dla niezorientowanych technicznie użytkowników Internetu. Autorzy nie chcą także tutaj mówić o subiektywnych wrażeniach użytkowania poszczególnych programów – w zasadzie tego rodzaju porównania na tyle zależą od indywidualnego gustu, że testy takie mógłby przeprowadzić na własne potrzeby każdy użytkownik Internetu.

Bezpieczeństwo informatyczne od wielu już lat postrzegane jest na świecie w trzech aspektach: poufności, integralności i dostępności.

Każda z przetestowanych wyżej przeglądarek w różnym stopniu adresuje powyższe aspekty, ale i wymagania użytkowników w odniesieniu do programów tego rodzaju są różne – w zależności od tego, jakie działania w Internecie będą przeprowadzać. Można sobie bardzo łatwo wyobrazić, że jeśli użytkownicy mają przesyłać dane poufne (np. dane bankowe lub tajemnice firmowe), najbardziej zależeć im będzie na poufności i oczywiście integralności tych danych. Mogą być za to dużo bardziej wyrozumiali w odniesieniu do dostępności serwisu. Zupełnie inaczej ma się sprawa w przypadku ogólnie dostępnych i anonimowych usług, takich jak serwisy internetowe typu VOD, serwisy informacyjne, serwisy rozrywkowe. Tu dostępność (i niekiedy integralność) danych staje się podstawowym dobrem, które z pewnością jest przedkładane nad zabezpieczeniem poufności – wręcz trudno sobie wyobrazić, żeby dane publicznie dostępne były jeszcze w jakiś szczególny sposób zabezpieczane przed podsłuchaniem.

Dokładnie taka sama sytuacja występuje po stronie klienckiej. Jeżeli użytkownik zamierza dokonać transakcji lub przesłać swoje dane osobowe, wyszuka – a przynajmniej powinien być w stanie to zrobić i dysponować narzędziem, które na to pozwala – serwisy, które umożliwią mu to w możliwie bezpieczny (poufność oraz integralność danych) sposób. Autorzy zachęcają w tym miejscu do zapoznania się z wynikami testów zawartych w raportach [9] oraz [10], opublikowanych wcześniej przez Zespół Bezpieczeństwa PCSS. Jeżeli zależy im jedynie na przeglądaniu treści, której podsłuchanie nie stanowi dla nich problemu – wybiorą do tego celu przeglądarki szybsze i w lepszy sposób prezentujące zawartość strony (np. z lepszą obsługą błędów języków prezentacji treści).

Autorzy niniejszego raportu chcieli pokazać w nim podstawowe różnice między najpopularniejszymi na rynku przeglądarkami, występujące w różnych aspektach bezpieczeństwa, ale odnoszące się do konkretnego zagadnienia (w tym przypadku obsługi tuneli SSL). Ponadto autorzy starali się patrzeć na problem oczami użytkownika, który nie posiada zaawansowanej wiedzy na temat zagrożeń i zabezpieczeń sieci (a także nie poświęca czasu na dokładną konfigurację oprogramowania).

Jak wynika z rezultatów przeprowadzonych testów, niektóre przeglądarki idealnie nadają się do bezpiecznego przesyłania danych (testy algorytmów szyfrowania) – w tym aspekcie przy standardowych ustawieniach najlepiej wypadły Mozilla Firefox, Google Chrome i Opera.

W odniesieniu do prędkości przesyłania danych w stronę serwera bardzo mocno swoją pozycję zaakcentowały Mozilla Firefox oraz Internet Explorer. Programy te – co interesujące – stosunkowo słabo wypadły przy przesyłaniu danych w przeciwnym kierunku. Szczególnie niekorzystne rezultaty osiągnęła tu przeglądarka Mozilla Firefox, pozostając bardzo daleko w tyle za liderem w pobieraniu dużych plików – przeglądarką Apple Safari.

Pozostała część raportu przedstawia zmagania przeglądarek z największym problemem producentów przeglądarek – i zarazem z najsłabszym obecnie ogniwem w łańcuchu bezpieczeństwa informatycznego – czyli samym użytkownikiem, jego zachowaniami oraz błędami. Na wielu zrzutach ekranu widać tu dokładnie, iż przeglądarka Opera, oraz zaraz za nią Mozilla Firefox, są najlepiej przystosowane do wykorzystania przez bardziej zaawansowanych użytkowników, a przeglądarka Google Chrome daje pewne poczucie bezpieczeństwa użytkownikom mniej zaawansowanym. Jeśli chodzi o przeglądarki Internet Explorer oraz Apple Safari – nadal mają one wiele niedociągnięć, choć w przypadku Internet Explorera widoczne są już podstawowe efekty wysiłków projektantów, którzy rozwijają tę przeglądarkę w kierunku zapewnienia wyższego poziomu bezpieczeństwa osobom zamierzającym korzystać z usług zabezpieczonych tunelami SSL/TLS.

4. Referencje

[1]. Eric Rescorla, SSL and TLS, Designing and Building Secure Systems, Addison-Wesley, 2000

[2]. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3555 – informacja o błędzie w mechanizmie renegocjacji protokołu TLS

[3]. http://en.wikipedia.org/wiki/Man-in-the-middle_attack - informacje o ataku Man-in-the-middle

[4]. http://en.wikipedia.org/wiki/RC4 - informacje na temat szyfru strumieniowego RC4

[5]. http://en.wikipedia.org/wiki/Transport_Layer_Security - informacje na temat protokołów SSL i TLS

[6]. http://en.wikipedia.org/wiki/X.509 - informacje na temat standardu X.509 PKI

[7]. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip - archiwum 1 miliona najpopularniejszych stron internetowych

[8]. http://security.psnc.pl/files/13.05.08.pdf - Gerard Frankowski, Błażej Miga – „13.05.08" – prezentacja na konferencję SECURE 2008 dotycząca błędu w generatorze liczb pseudolosowych OpenSSL w systemie Linux Debian

[9]. http://security.psnc.pl/reports/e-banking_polska_ssl_report.pdf - Zespół Bezpieczeństwa PCSS, „«Bezpieczna» e-bankowość" – raport nt. bezpieczeństwa szyfrowanej transmisji w połączeniach z polskimi portalami bankowości elektronicznej

[10]. http://security.psnc.pl/reports/sklepy_internetowe_cookies.pdf - Zespół Bezpieczeństwa PCSS, „Bezpieczeństwo sklepów internetowych – sesje i ciasteczka" – raport nt. bezpieczeństwa implementacji sesji HTTP w sklepach internetowych

[11]. http://www.alexa.com – portal Alexa, m.in. źródło listy najpopularniejszych stron w Internecie

[12]. http://www.apple.com/safari - przeglądarka internetowa Apple Safari

[13]. http://www.firefox.com – przeglądarka internetowa Mozilla Firefox

[14]. http://www.google.com/chrome - przeglądarka internetowa Google Chrome

[15]. http://www.ietf.org – Internet Engineering Task Force (IETF)

[16]. http://www.ietf.org/rfc/rfc2246.txt - RFC 2246: The TLS Protocol – Version 1.0

[17]. http://www.ietf.org/rfc/rfc4346.txt - RFC4346: The Transport Layer Security (TLS) Protocol - Version 1.1

[18]. http://www.ietf.org/rfc/rfc5246.txt - RFC5246: The Transport Layer Security (TLS) Protocol - Version 1.2

[19]. http://www.ietf.org/rfc/rfc5746.txt - RFC 5746: Transport Layer Security (TLS) Renegotiation Indication Extension

[20]. http://www.microsoft.com/windows/ie - przeglądarka internetowa Microsoft Internet Explorer

[21]. http://www.openssl.org – strona projektu OpenSSL

[22]. http://www.opera.com – przeglądarka internetowa Opera

[23]. http://www.rfc-editor.org/rfcsearch.html - wyszukiwarka dokumentów RFC

[24]. http://www.youtube.com/watch?v=FxOIebkmrqs – film w komiczny sposób przedstawiający obciążanie użytkowników systemów komputerowych zbędnymi decyzjami

5. Dane kontaktowe

Poznańskie Centrum Superkomputerowo – Sieciowe
Zespół Bezpieczeństwa
ul. Noskowskiego 10
61-704 Poznań
POLSKA