
	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

POLITYKA BEZPIECZEŃSTWA W MODULE IRAP

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 1
--	-------------------------	------------	--------------	----------

Spis treści

Wprowadzenie.....	3
Protokół SSL/TLS	4
Lista zestawień algorytmów szyfrujących (cipher list).....	5
Certyfikaty SSL	6
Importowanie certyfikatów w przeglądarce internetowej.....	8
Generowanie Certyfikatów dla modułu iRap.....	11
Open SSL	11
Przygotowanie przestrzeni roboczej.....	12
Konfiguracja OpenSSL	13
Tworzenie głównego certyfikatu CA	15
Generacja klucza prywatnego certyfikatu CA	15
Generowanie samo-podpisanego certyfikatu CA	15
Generowanie certyfikatu pośredniego.	17
Generowanie klucza prywatnego pośredniego certyfikatu CA.....	17
Tworzenie CSR certyfikatu pośredniego CA.....	17
Generowanie łańcucha certyfikacji.....	19
Generowanie certyfikatu serwera iRap.	19
Konfiguracja iRap	22
Konfiguracja obsługiwanych protokołów SSL/TLS	23
Konfiguracja Cipher list	23
Ocena poziomu bezpieczeństwa na serwerze iRap.	25
Załącznik A. Plik konfiguracyjny głównego certyfikatu CA.....	27
Załącznik B. Plik konfiguracyjny pośredniego certyfikatu CA.....	29
Załącznik C: Raport SSL Labs dla domyślnej konfiguracji serwera iRap	31
Załącznik D: Instrukcja skrócona – generowanie certyfikatu SSL i konfiguracja IRAP	35

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

Wprowadzenie

Funkcjonalność modułu **iRap** z jednej strony polega na udostępnianiu raportów magazynowo - finansowych na serwerze WWW, a z drugiej strony pozwala na wymianę informacji między aptekami tj. składanie zapytań ofertowych, realizację zamówień, przesunięcia między-magazynowego itp. Każda z tych funkcjonalności opiera się na transmisji poufnych informacji, przy pomocy medium jakim jest Internet. Połączenie nieszyfrowane, w którym treści przekazywane są otwartym tekstem, stosunkowo łatwo przechwycić. Dlatego ważną kwestią jest zapewnienie wysokiego poziomu bezpieczeństwa transmisji danych, poprzez zastosowanie protokołów zabezpieczających: SSL (*Secure Socket Layer*) lub TLS (*Transport Layer Security*). Protokoły te odpowiedzialne są za szyfrowanie przesyłanych informacji, co pozwala zapobiegać niechcianym atakom tj. próbą wykradzenia danych.

iRap jest swojego rodzaju serwerem WWW, który determinuje sposób wymiany danych z klientem. Możliwe jest ustawienie połączenia niezabezpieczonego (HTTP), mieszanego (HTTP \ HTTPS) bądź też wyłącznie szyfrowanego (HTTPS). W najnowszej wersji iRap obsługiwane są powszechnie używane protokoły zabezpieczeń, począwszy od SSL v2 po TLS v1.2.


Niniejszy dokument ma na celu przybliżyć czytelnikowi kwestię zabezpieczeń transmisji danych. Pokróćce omówiony zostanie temat komunikacji opartej na protokołach SSL/TLS wraz z wyjaśnieniem terminu **Cipher list**. Czytelnik zapozna się z pojęciami: infrastruktury kluczy publicznych (**PKI**) oraz certyfikatami SSL i sposobem ich instalacji w przeglądarkach internetowych.

W kolejnych rozdziałach - odtworzony zostanie proces generowania certyfikatów SSL przy pomocy otwartej biblioteki Open SSL oraz omówiona zostanie konfiguracja modułu iRap.

Ostatecznie, przy pomocy usługi internetowej **SSL Labs**, wykonane zostaną testy serwera WWW, w celu wykazania poziomu bezpieczeństwa nowo powstałej konfiguracji.

W załącznikach A i B umieszczono pliki konfiguracyjne przydatne przy tworzeniu własnych certyfikatów. Załącznik C przedstawia raport (SSL Labs) z testu zabezpieczeń serwera testowego iRap (konfiguracja domyślna). Załącznik D zawiera **skróconą instrukcję** generowania certyfikatów SSL i konfiguracji iRap (potrzebne pliki umieszczono na <ftp://ftp.kamsoft.pl/pub/KS-APW/2017/Inne/ssl.zip>).

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 3
--	-------------------------	------------	--------------	----------

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

Protokół SSL/TLS

Protokół SSL/TLS służy zabezpieczeniu transmisji danych przez Internet. Bezpieczeństwo jest zapewnione poprzez uwierzytelnianie (w naszym przypadku – autoryzowanie certyfikatu SSL serwera iRap), szyfrowanie danych (zapewnienie poufności) oraz integralność informacji (przeciwdziałanie utracie danych).

Połączenie Klienta z Serwerem tworzy sesję, której inicjalizację ogólnie można opisać w następujących punktach:

1) Klient rozpoczyna połączenie, po czym następuje negocjacja z serwerem parametrów zabezpieczających (tzw. TLS **Handshake** Protocole).

Ustalane są m.in.:

- Wersja protokołu SSL\TLS (SSL1, SSL2, SSL3, TLS1.0, TLS1.1, TLS1.2)
- Liczby losowe, które posłużą do tworzenia klucza wymiany danych. Generowane są niezależnie, zarówno przez Serwer jak i Klienta.
- Sposób kompresji danych lub jego brak.
- Zestawienie algorytmów szyfrujących. Przy rozpoczęciu negocjacji parametrów zabezpieczeń - Klient przesyła na Serwer „Listę obsługiwanych zestawień szyfrujących” (ang. *Cipher suites*) - zwaną dalej **Cipher list**. Serwer, w odpowiedzi wskazuje jedno wybrane zestawienie szyfrujące. Dzięki temu Klient oraz Serwer są w stanie określić w jaki sposób odbędzie się zakodowana wymiana danych. Więcej informacji na temat *cipher list* znajduje się w rozdziale pt. „Lista zestawień algorytmów szyfrujących (*Cipher List*)”

2) Serwer wysyła certyfikat SSL w celu uwierzytelnienia swojej tożsamości. Certyfikat zawiera w sobie klucz publiczny.

3) Klient weryfikuje tożsamość serwera. W przypadku wystąpienia problemów sesja zostaje przerwana. Najczęściej, w tym momencie widzimy w przeglądarce komunikat błędu. Przyczyną przerwania sesji może być nieważny certyfikat SSL, czy też niezauwany Urząd Certyfikacji będący podstawą łańcucha certyfikacji.

4) Klient generuje tajny ciąg znaków, tzw. „**premaster secret**”. Ciąg ten jest szyfrowany przy użyciu klucza publicznego, po czym następuje jego przekazanie na Serwer. Połączenie Klient-Serwer jest inicjalizowane algorytmem asymetrycznym, który ustalono przy wyborze zestawienia szyfrującego z *cipher list*. To oznacza, że wymagane jest posiadanie przez Serwer pary kluczy – publicznego oraz prywatnego. Dane zaszyfrowane kluczem publicznym można odkodować tylko przy pomocy klucza prywatnego (np. *RSA*, *Diffie-Helman*). Podpisany przez Klienta *premaster secret* może być więc odczytany tylko przez Serwer.

5) Na podstawie wygenerowanych na etapie negocjacji - „liczb losowych” oraz wygenerowanego przez Klienta „*premaster secret*” - powstaje właściwy klucz sesji, dzięki któremu obie strony mogą się porozumieć. Klucz ten posłuży zarówno do szyfrowania jak i deszyfrowania przesyłanych danych. Oznacza to, że szyfrowanie transmisji danych nastąpi przy pomocy symetrycznego algorytmu szyfrowania (np. *DES3*, *AES256*). Algorytm ten również został określony przy doborze zestawienia szyfrującego z *cipher list*.

6) Klient wysyła informację do Serwera, że od teraz będzie używać uzgodnionych parametrów zabezpieczających.

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 4
--	-------------------------	------------	--------------	----------

7) W ostatnim etapie następuje przesłanie zakodowanych danych

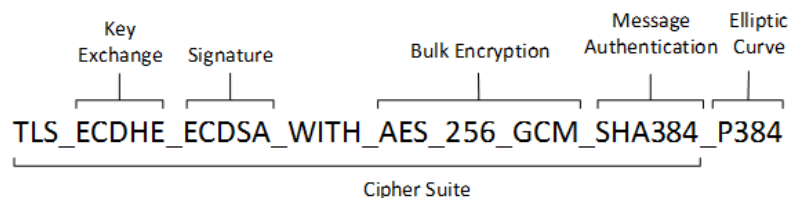
Więcej informacji dotyczących transmisji danych TLS w najnowszej wersji 1.2 można odnaleźć w RFC 5246 <https://tools.ietf.org/html/rfc5246>

Lista zestawień algorytmów szyfrujących (cipher list)

Jak wspomniano w rozdziale „Protokół SSL/TLS” - *cipher list*, zwany również *cipher suites*, jest listą zestawień algorytmów szyfrujących. Przy rozpoczęciu negocjacji parametrów sesji między Klientem a Serwerem ustalane jest jedno zestawienie wg którego odbędzie się dalsze postępowanie zabezpieczające transmisję.

Pojedyncze zestawienie algorytmów szyfrujących najczęściej opisane jest przy pomocy jednego łańcucha znaków. Zapis ten różni się w zależności od wykorzystywanego serwera WWW (Windows/Apache...) .

Przykładowo dla niektórych serwerów Windows, zestawienie algorytmów szyfrujących TLS oraz krzywych eliptycznych jest zapisywane jak poniżej:

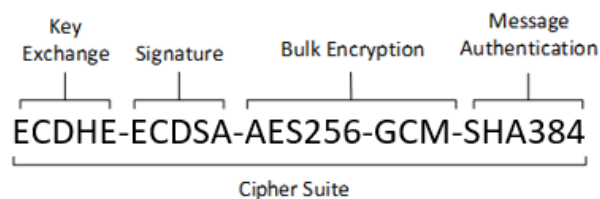


Rys. 1 Zestawienie algorytmów szyfrujących – notacja Windows

Źródło: [https://msdn.microsoft.com/pl-pl/library/windows/desktop/aa374757\(v=vs.85\).aspx](https://msdn.microsoft.com/pl-pl/library/windows/desktop/aa374757(v=vs.85).aspx)

Zapis ten różni się od notacji stosowanego w interesującej nas bibliotece OpenSSL - co wynika z różnic w implementacji protokołów SSL/TLS.

Odpowiednikiem powyższego ciągu *Cipher suite* w OpenSSL jest:




Rys. 2 Zestawienie algorytmów szyfrujących – notacja OpenSSL.

Na budowę jednej sekwencji z *cipher list* wchodzi:

- 1) Metoda wymiany klucza– (ang. *Key Exchange*) np. RSA, Diffie-Helman, ECDH –służy do definicji sposobu wymiany klucza *premaster secret*.
- 2) Sygnatura – określa sposób podpisu cyfrowego.

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 5
--	-------------------------	------------	--------------	----------

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

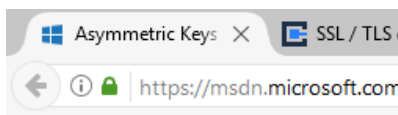
- 3) Symetryczny algorytm szyfrowania – metoda zaszyfrowania przesyłanych danych (np. AES 256, DES3).
- 4) Funkcja skrótu – zapewnia spójność i integralność danych, np. MD5,SHA1.

Dobór odpowiedniej listy zestawień algorytmów szyfrujących, dostępnej na serwerze jest bardzo istotny. Stanowi on o poziomie bezpieczeństwa jaki zapewnia serwer. Określając *cipher list* należy jednak pamiętać, że dobór wyłącznie silnych zestawień szyfrujących, może doprowadzić do sytuacji, w której starsze przeglądarki internetowe nie będą w stanie nawiązać połączenia z naszym serwerem WWW.

Porównując oba ciągi, zauważyć można brak sekcji krzywej eliptycznej (ang. Elliptic curve) w notacji Open SSL. Opisując w skrócie Elliptic Curve Cryptography (ECC) jest jedną z technik szyfrowania asymetrycznego wykorzystującą w swych algorytmach krzywe eliptyczne, zapewnia bezpieczeństwo na poziomie algorytmów RSA, przy zachowaniu wysokiej wydajności (użycie krótszych kluczy). Brak omawianej sekcji nie oznacza, że openssl nie pozwala używać krzywych eliptycznych przy wykonywaniu operacji kryptograficznych. Open SSL umożliwia tworzenie kluczy przy pomocy Elliptic Curve Diffie Hellman (ECDH) oraz Elliptic Curve Digital Signature Algorithm (ECDSA) - polecenie „ecparam”. Więcej informacji można znaleźć na stronie: https://wiki.openssl.org/index.php/Command_Line_Elliptic_Curve_Operations

Certyfikaty SSL

PKI (ang. *Public Key Infrastructure*) to koncepcja systemu kryptograficznego, której celem jest zwiększenie bezpieczeństwa w Internecie. Jej ideą jest wykorzystanie certyfikatów SSL, potwierdzających wiarygodność kluczy publicznych, wykorzystywanych do zabezpieczenia transmisji danych. Uwierzytelnianie zapewnione jest przez Urząd Certyfikacji (ang. *Certification Authority* –zwady dalej CA), który to jest jednym z wielu międzynarodowych podmiotów wydających użytkownikom końcowym podpisane certyfikaty. Powstaje w ten sposób łańcuch zaufania, dzięki któremu CA jest w stanie potwierdzić (jako organ niezależny) tożsamość partnera - jego klientom. Służą temu protokoły SSL/TLS oraz wbudowane w przeglądarki internetowe mechanizmy weryfikujące tożsamość certyfikatu . Dla klienta strony internetowej, informacja o zabezpieczonym połączeniu, jest oznaczona najczęściej symbolem zielonej kłódki, który pojawia się w pasku adresowym przeglądarki. Co ilustruje poniższe zdjęcie.

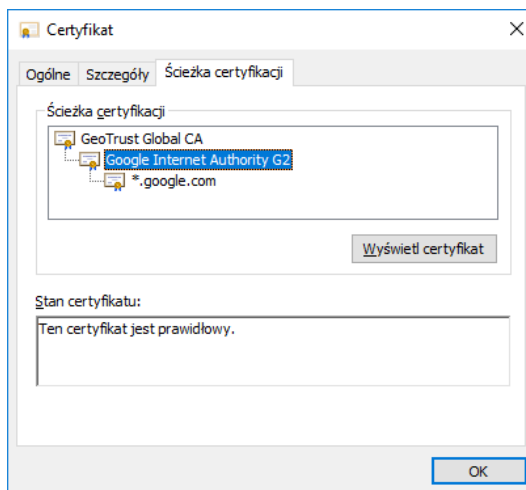


Rys. 3 Przykład oznaczenia zabezpieczonej sesji w przeglądarki internetowej.

Dodanie certyfikatów pośrednich CA między certyfikatem CA (tzw. *root CA*) a certyfikatem końcowym (serwerowym) – zmniejsza prawdopodobieństwo kompromitacji certyfikatu głównego. Jeżeli *root CA* zostałby na skutek „ataku” skompromitowany, to wszystkie poświadczane przez niego certyfikaty również nie mogłyby zagwarantować poufności transmisji danych.

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 6
--	-------------------------	------------	--------------	----------

Poniższy rysunek przedstawia łańcuch zaufania certyfikatu końcowego wydanego dla domeny *.google.com, przez Organ Certyfikacji GeoTrust. Widzimy na szczycie główny certyfikat - root Ca: „GeoTrust Global CA” oraz pośrednio zastosowany przez Google -certyfikat pośredni: „Google Internet Authority G2”.



Rys. 4 Łańcuch certyfikacji pobrany z <https://www.google.pl>


Oprócz Urzędów certyfikacji i użytkowników końcowych w skład PKI wchodzi również:

- Urząd rejestracji, który prowadzi rejestr zweryfikowanych użytkowników.
- Implementacje CRLs (*ang. Certificate Revocation Lists*) umożliwiające prowadzenie repozytorium kluczy, certyfikatów i unieważnionych certyfikatów.

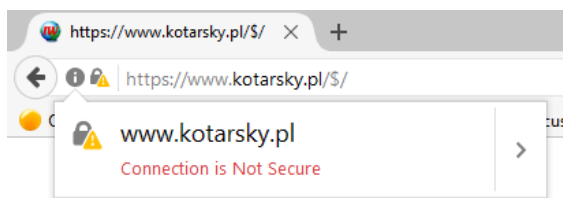
Certyfikat można nabyć w wielu renomowanych Urzędach Certyfikacji np. *CERTUM, Comodo, DigiCert, DomenySSL, GeoTrust, RapidSSL, SwissSign, Symantec (VeriSign)*. Inwestycja taka wiąże się jednak z dużymi kosztami. Należy też pamiętać, że certyfikaty wydawane są na dany okres czasu. Gdy ważność certyfikatu wygaśnie, a przeglądarka internetowa wykryje próbę otwarcia takiej witryny, to od razu przerwie ten proces stosownym komunikatem o zagrożeniu.

Zakup certyfikatu SSL wydaje się konieczny w przypadku gdy dana firma, oprócz zabezpieczenia transmisji danych - musi bądź chce być wiarygodna dla swoich bieżących i docelowych użytkowników.

Istnieje jednak wiele przypadków dla których wiarygodność ta wcale nie jest wymagana. Przykładem tego mogą być witryny firmowe, które to działają lokalnie (w sieci LAN) czy też przez Internet. Szyfrowanie danych jest podstawą bezpieczeństwa. Kwestia wiarygodności sprowadza się do wzajemnego zaufania między dostawcą tejże witryny, a pracownikami firmy. Tym bardziej w przypadku, gdy rozdysponowujemy dostęp do własnej witryny, gdzie chcemy przyznać poświadczenia jedynie sobie i własnym pracownikom. W takiej sytuacji możemy wcielić się w rolę Urzędu Certyfikacji i stworzyć własny, główny certyfikat. Przy jego pomocy utworzymy kolejno certyfikat pośredni i certyfikat serwera www. Wygląda to tak jakbyśmy ręczyli sami za siebie. Rozwiązanie takie jest domyślnie dostarczane z modułem iRap.

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

W przypadku braku certyfikatu poświadczonego przez CA, przy próbie otwarcia danej strony w przeglądarce internetowej, sesja zostanie przerwana. Użytkownik otrzyma komunikat o błędzie z brakiem poświadczenia zaufanego Urzędu Certyfikacji (kod błędu Firefox: SEC_ERROR_UNKNOWN_ISSUER). Jest to normalne zachowanie każdej przeglądarki, dla której stworzony przez nas certyfikat CA jest nieznanym. Można dodać wyjątek do przeglądarki internetowej, co spowoduje załadowanie strony. Problem zaufania jednak nie zniknie. Dodanie wyjątku wciąż nie oznacza dla przeglądarki internetowej, że można w pełni zaufać naszemu certyfikatowi. Co widać na poniższym zdjęciu



Rys. 5 Firefox: Ostrzeżenie po dodaniu wyjątku na stronie zabezpieczonej certyfikatem samo-podpisanym (self-signed).

Widać, że wciąż istnieje wątpliwość co do wydawcy certyfikatu CA. Aby przeglądarka zaufała naszej witrynie iRap należy zapisać na dysk certyfikat główny i certyfikat pośredni, a następnie zaimportować je jako poświadczenia Urzędu Certyfikacji. Każda przeglądarka ma swój własny magazyn certyfikatów. Poniżej opisane czynności, należy wykonać na wszystkich używanych przeglądarkach internetowych.

Importowanie certyfikatów w przeglądarce internetowej

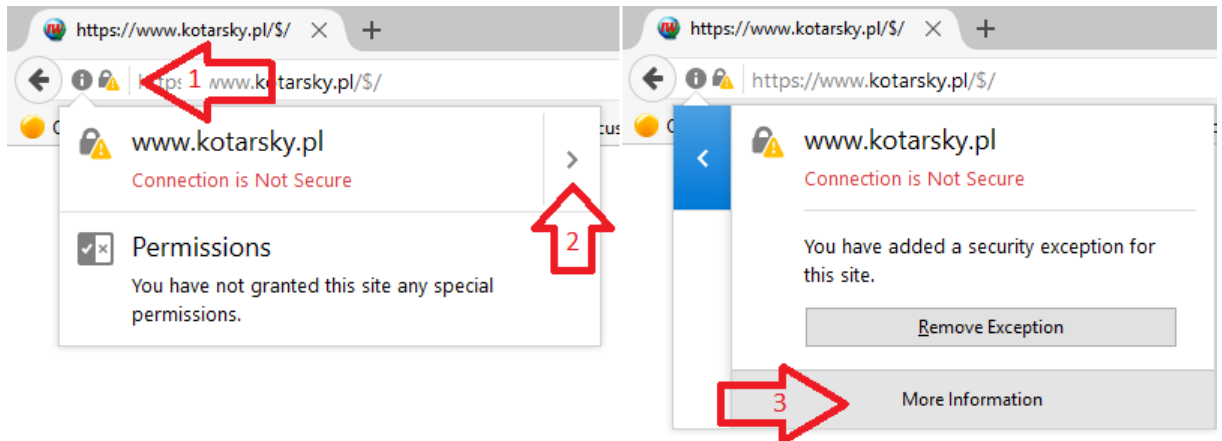
Poniższy opis wykonano na przeglądarce Firefox. Jednakże proces ten powinien wyglądać analogicznie w każdej innej przeglądarce internetowej.

Jeżeli po wejściu na adres domeny pod którą udostępniono (poprzez szyfrowane połączenie) stronę iRap - pojawi się komunikat o nieznanym Urzędzie Certyfikacji, to należy:

- 1) Dodać wyjątek
- 2) Pobrać certyfikat CA oraz certyfikat pośredni CA
- 3) Zaimportować certyfikaty do magazynu przeglądarki jako Urząd\Organ Certyfikacji

Aby pobrać z wyświetlanej strony certyfikaty, należy kliknąć w pasku adresowym kłódkę z żółtym trójkątem (znak uwaga) – co obrazuje rys. 6. Następnie klikamy na strzałeczkę ulokowaną z prawej strony (2) i wybieramy przycisk „Więcej informacji” (3)

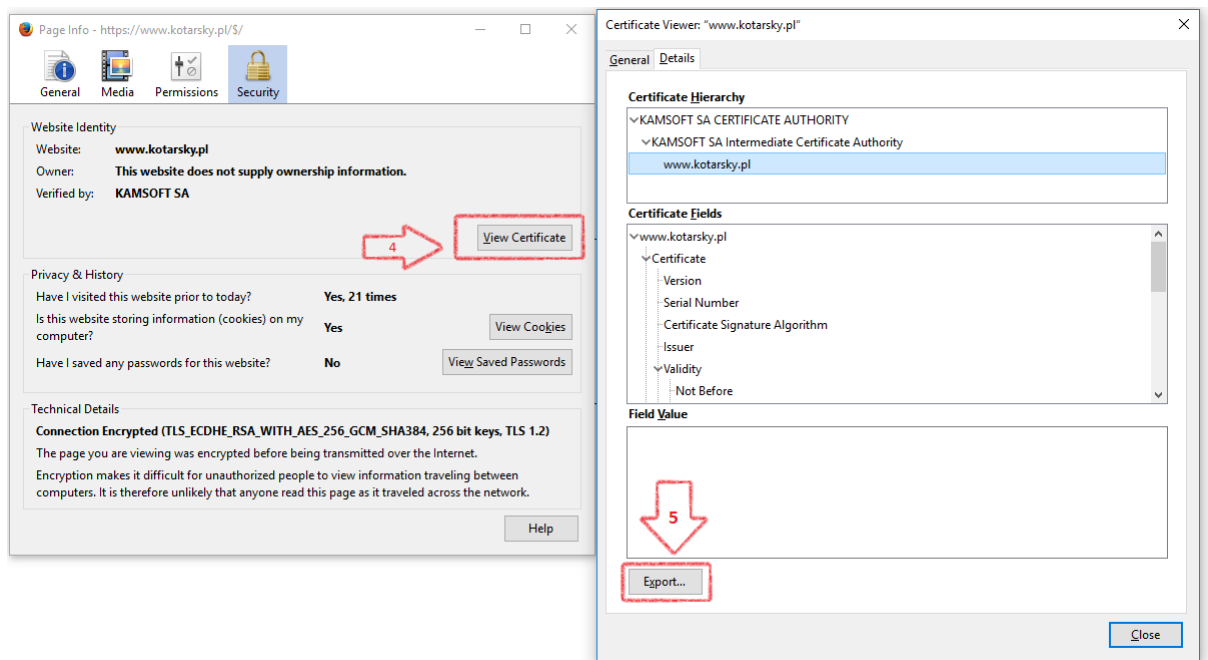
Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 8
--	-------------------------	------------	--------------	----------



Rys 6. Sposób wyświetlenia informacji o stronie WWW i jej certyfikatach.

W okienku „Informacje o stronie” - wybieramy guzik (4) „Pokaż certyfikat”.

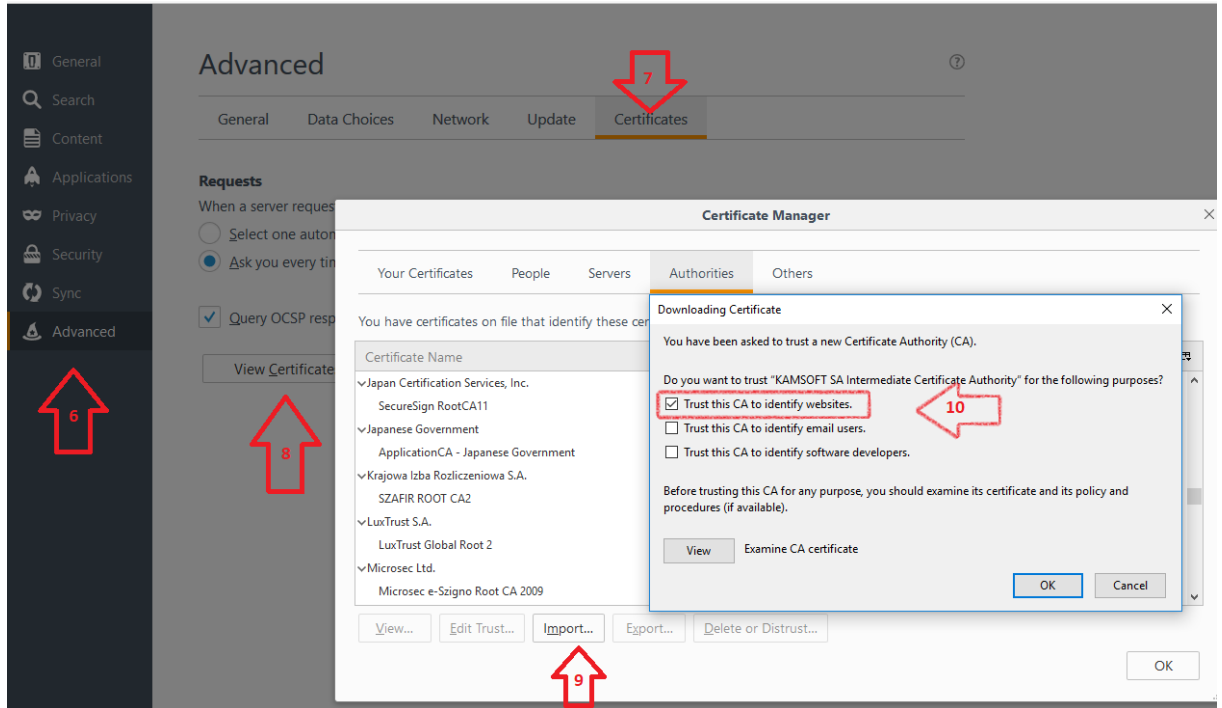
W podglądzie certyfikatu (rysunek z prawej strony) należy zaznaczyć certyfikat główny (pierwszy z góry) i wyeksportować go na dysk (5). Powtarzamy te czynności w celu pobrania certyfikatu pośredniego.



Rys. 7 Podgląd właściwości przeglądanego strony (z lewej) oraz okno właściwości certyfikatu (z prawej).

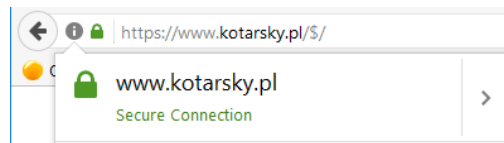
W ostatnim kroku wchodzimy w „opcje” przeglądarki, „ustawienia zaawansowane” (6) , „certyfikaty” (7) i na zakładce „Urzędy certyfikacji” (7) dodajemy (9) certyfikat główny i pośredni. W celu potwierdzenia tożsamości witryny, musimy oznaczyć ją jako zaufaną (10) .

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 9
--	-------------------------	------------	--------------	----------



Rys. 8 Zaawansowane ustawienia przeglądarki Firefox. Dodawanie certyfikatu Urzędu Certyfikacji dla zaufanej strony.

W wyniku powyżej opisanych czynności, użytkownik naszej witryny, będzie mógł cieszyć się komunikatem o zabezpieczonym połączeniu.



Rys. 9 Zielona kłódka, oznaczająca zabezpieczone połączenie.

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 10
--	-------------------------	------------	--------------	-----------

Generowanie Certyfikatów dla modułu iRap.

UWAGA! W załączniku D znajduje się skrócona instrukcja. Dzięki niej można znacznie skrócić czas tworzenia certyfikatów SSL. Kompilacja Open SSL, struktura katalogów, pliki konfiguracyjne są dostępne na firmowym ftp (adres w załączniku).

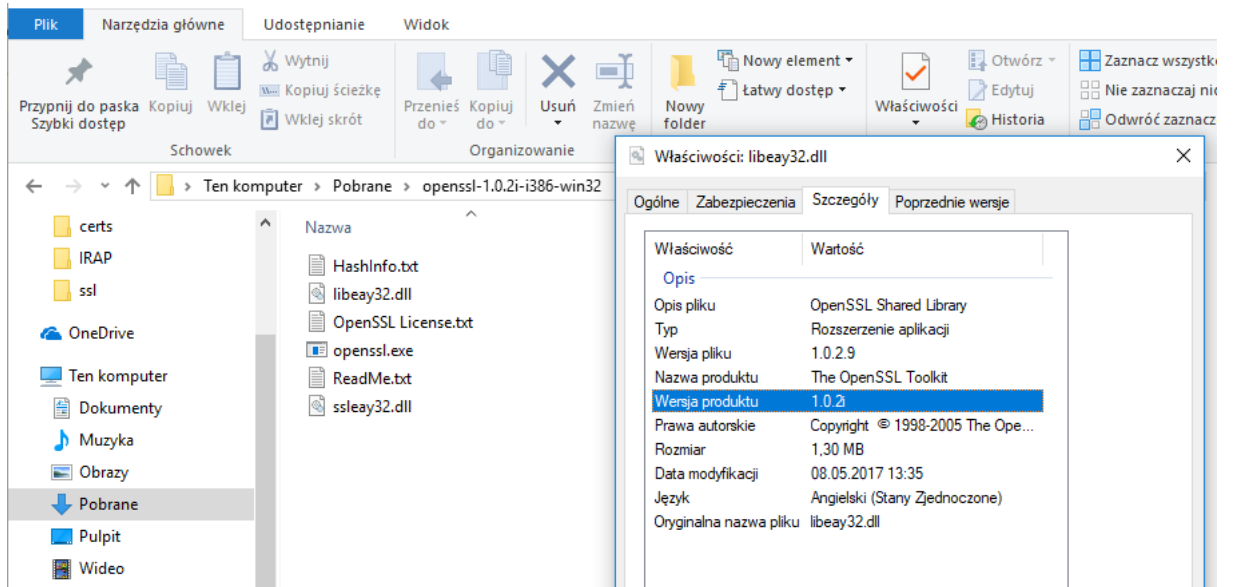
W niniejszym rozdziale wykonamy pełną ścieżkę certyfikacji przy pomocy Open SSL. Wcielając się w rolę **Urzędu Certyfikacji** (ang. Certificate Authority) sporządzimy **certyfikat CA** (tzw. *self-signed root CA*) oraz **pośredni certyfikat CA** (ang. Intermediate Certificate CA). Następnie już jako właściciel witryny iRap - wygenerujemy klucz prywatny oraz żądanie podpisania certyfikatu **CSR** (ang Certificate Signing Request) - na którego podstawie sami sobie wystawimy końcowy certyfikat SSL serwera, dla naszej domeny. W wyniku tych działań powstaną niezbędne pliki do uruchomienia modułu iRap z zabezpieczeniami SSL/TLS. Wymagane jest dostarczenie pliku z certyfikatem SSL serwera (cert.pem), pliku z certyfikatem głównym (root.pem), pliku z kluczem prywatnym (key.pem) oraz tajnego hasła zabezpieczającego ten klucz. Pliki umieszczamy w katalogu IRAP (C:\KS\APW\IRAP). Nazwy plików nie mogą ulec zmianie. Konfigurację modułu iRap przedstawiono w następnym rozdziale.

Open SSL

OpenSSL jest projektem typu *open source*, który zapewnia dostęp do bibliotek i narzędzi implementujących protokoły SSL/TLS oraz mechanizmy kryptograficzne takie jak algorytmy szyfrujące, generowanie kluczy, funkcje skrótu itp.

Open SSL pozwala generować pliki CSR, dzięki którym możliwe jest złożenie zamówienia certyfikatów w Urzędzie Certyfikacji. Ponadto program umożliwi samodzielne utworzenie pełnego łańcucha certyfikatów (od CA poprzez CA pośredni po certyfikat strony www)

Zanim jednak zaczniemy..., konieczne jest wykonanie paru czynności poprzedzających generowanie certyfikatów. Pobieramy OpenSSL w wersji 1.0.2i z Internetu na dysk twardy i sprawdzamy czy po jego rozpakowaniu, w katalogu znajduje się plik wykonywalny openssl.exe oraz biblioteki libeay32.dll, sslib32.dll.



Rys. 10 Zrzut ekranu z katalogu roboczego w którym znajdują się biblioteki OpenSSL

Plik openssl.exe w proponowanej kompilacji, jest programem wykonywalnym. Można go uruchomić zarówno z poziomu wiersza poleceń jak i wywołać dwuklikiem.

Przygotowanie przestrzeni roboczej

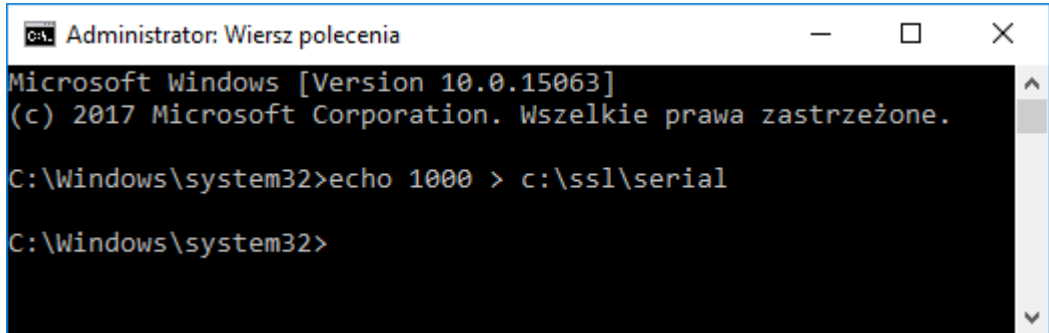
W następnym kroku przygotowujemy strukturę katalogów, dzięki której nasze generowane zbiory będą uporządkowane. Wybór lokalizacji katalogu roboczego jest dowolny, sugeruję jednak utworzenie katalogu `C:\SSL`. Krótka ścieżka do zbiorów usprawni wpisywanie poleceń w konsoli Open SSL.

W katalogu `C:\SSL\` tworzymy następujące podkatalogi:

1. *certs* – przechowuje root CA
2. *private* – przechowuje wygenerowany klucz prywatny do root CA
3. *int* – w tym katalogu będziemy przechowywać strukturę katalogów i pliki certyfikatu pośredniego oraz certyfikatu serwera.
4. *newcerts* – katalog roboczy, powstają w nim certyfikaty tymczasowe na etapie generacji.

...oraz pliki:

1. *index.txt* – baza danych Open SSL, w niej są zapamiętywane dane identyfikacyjne utworzonych certyfikatów przy pomocy certyfikatu CA. Nie należy go edytować ręcznie.
2. *serial* – plik roboczy Open SSL. Utwórz pusty plik (bez rozszerzenia). Uruchom CMD jako administrator i wykonaj polecenie `echo 1000 > c:\ssl\serial`



Rys. 11 Wypełniania pliku *serial* przy pomocy polecenia wykonanego w wierszu poleceń.

3. *openssl.cnf* – plik konfiguracyjny zawierający treść z załącznika A. jeżeli ścieżka do katalogu roboczego uległa zmianie, to należy poprawić ten pliku

Konfiguracja dla pośredniego certyfikatu CA wygląda podobnie jak konfiguracja certyfikatu głównego W katalogu *C:\SSL\int* tworzymy następujące podkatalogi:

1. *certs* – przechowuje pośredni certyfikat CA i certyfikaty serwera
2. *private* – przechowuje wygenerowane klucze prywatne do certyfikatów serwera i pośredniego certyfikatu CA.
3. *csr* – w tym katalogu przechowywane są żądania podpisania certyfikatów serwera i pośredniego certyfikatu CA.
4. *newcerts* – katalog roboczy, powstają w nim certyfikaty tymczasowe na etapie generacji.

...oraz puste pliki

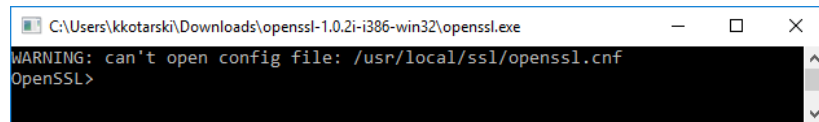
1. *index.txt* – baza danych Open SSL, w niej są zapamiętywane dane identyfikacyjne utworzonych certyfikatów przy pomocy pośredniego certyfikatu CA. Nie należy go edytować ręcznie
2. *serial* – utworzyć podobnie jak dla konfiguracji CA (wypełnić poleceniem *echo* z zachowaniem właściwej ścieżki)
3. *openssl.cnf* – plik konfiguracyjny zawierający treść z załącznika B. jeżeli ścieżka do katalogu roboczego uległa zmianie, to należy poprawić ten pliku.

Konfiguracja OpenSSL

OpenSSL wymaga do pracy pliku konfiguracyjnego *openssl.cnf*. Plik ten ułatwia proces generowania certyfikatów. Zawiera informacje o lokalizacji katalogów roboczych, definicje parametrów: używanych funkcjach skrótów; rozszerzonych ustawieniach certyfikatów itp. Poprawne sporządzenie pliku konfiguracyjnego i uporządkowanej struktury katalogów usprawni w przyszłych latach odnawianie certyfikatów SSL dla naszej domeny. W załącznikach niniejszego dokumentu zawarto treść plików konfiguracyjnych (osobno dla certyfikatu głównego (załącznik A) i osobno dla certyfikatów serwera i certyfikatu pośredniego (załącznik B).

Należy w katalogu roboczym *C:/ssl/* utworzyć plik *openssl.cnf* wypełniony treścią załącznika A. Podobnie w katalogu *C:/ssl/int/* tworzymy plik *openssl.cnf* zawierający treść załącznika B.

Uruchamiając openSSL bez pliku konfiguracyjnego zostanie wyświetlony komunikat informujący nas o jego braku.



Rys 12. Komunikat Openn SSL o braku pliku konfiguracyjnego

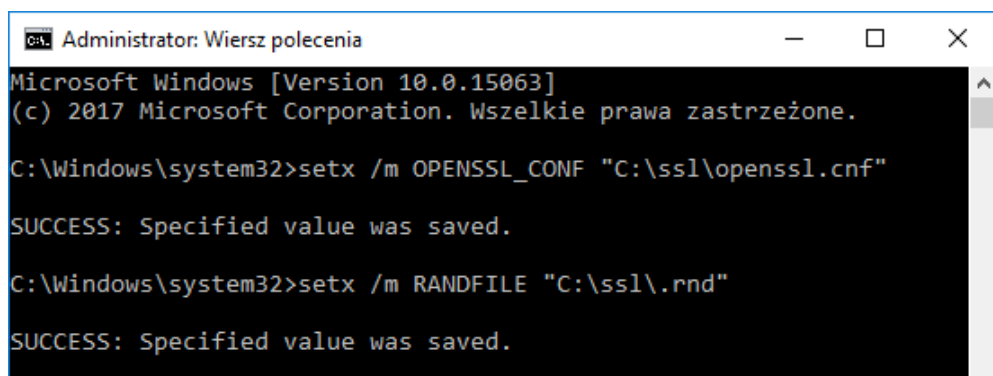
Przed rozpoczęciem pracy z Open SSL należy więc wskazać lokalizację pliku konfiguracyjnego poprzez utworzenie zmiennej środowiskowej OPENSSL_CONF.

Oprócz tego Open SSL w wersji dla Windows wymaga utworzenia zmiennej środowiskowej RANDFILE (plik roboczy).

Wykonać to można na 2 sposoby – z wiersza poleceń lub „okienkowo”.

- 1) Dodanie zmiennej środowiskowej OPENSSL_CONF i RANDFILE poprzez wiersz poleceń. Po uruchomieniu wiersza poleceń z poziomu Administratora wpisujemy komendy:

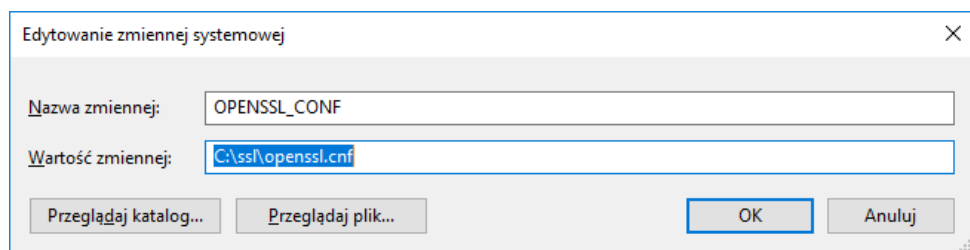
```
setx /m RANDFILE "C:\ssl\rnd"
setx /m OPENSSL_CONF"C:\ssl\openssl.cnf"
```



Rys. 13 Definiowanie zmiennych środowiskowych w wierszu poleceń.

- 2) Dodanie zmiennej środowiskowej OPENSSL_CONF i RANDFILE w systemie operacyjnym. W celu dodania zmiennej środowiskowej w Windows 10 należy wybrać kolejno następujące opcje\zakładki:

Ten Komputer>Właściwości>Zmień Ustawienia>Zaawansowane>Zmienne środowiskowe



Rys.14 Okno edycji zmiennej środowiskowej.

Proszę zwrócić uwagę na wielkość liter – ma znaczenie. Analogicznie tworzymy zmienną RANDFILE (C:\ssl\rnd)

Podczas generowania kluczy zauważyłem, że w użytej kompilacji OpenSSL dla Windows, trzeba podawać pełne ścieżki do plików we/wy. Podobnie mimo, że ścieżkę do RANDFILE umieszcza się w pliku

konfiguracyjnym to i tak program nie bierze jej pod uwagę. Stąd konieczność tworzenia tej zmiennej środowiskowej.

Tworzenie głównego certyfikatu CA

Aby utworzyć główny certyfikat CA w pierwszej kolejności musimy wygenerować klucz prywatny (*keyCA.pem*). Dobrą praktyką jest wykonanie procesu generowania certyfikatów na środowisku hermetycznym, odłączonym od Internetu. Wszystkie klucze muszą pozostać w bezpiecznym miejscu, tak aby nie doszło do ich kompromitacji.

Generacja klucza prywatnego certyfikatu CA

Wygenerujemy 4096 bitowy klucz prywatny przy pomocy algorytmu asymetrycznego RSA. Klucz zostanie dodatkowo zaszyfrowany algorytmem AES-256, przy użyciu wymyślonego przez nas silnego hasła (Zawierającego ≥ 8 znaków, duże i małe litery, co najmniej jedną liczbę i znak specjalny). Alternatywą dla szyfru AES256 może być algorytm DES3. Postąpimy tak zarówno dla głównego certyfikatu jak i pośredniego. Wydając certyfikat dla naszej domeny, można zastosować krótszy (2048 bitowy) klucz prywatny - ze względu na wydajność.

```

OpenSSL> genrsa -aes256 -out c:\ssl\private\keyCA.pem 4096
Generating RSA private key, 4096 bit long modulus
.....++
.....++
e is 65537 (0x10001)
Enter pass phrase for c:\ssl\private\keyCA.pem:
Verifying - Enter pass phrase for c:\ssl\private\keyCA.pem:


```

Generowanie samo-podpisanego certyfikatu CA

Kolejnym krokiem jest utworzenie certyfikatu głównego, zawierającego w sobie klucz publiczny. W tym celu musimy użyć wcześniej utworzonego klucza prywatnego, do którego ścieżkę wskazujemy przez parametr *-key*. Parametr *-config* jest opcjonalny jeżeli zmienna środowiskowa *OPENSSL_CONF* wskazuje na ten sam plik. Certyfikat będzie ważny przez ok. 20 lat (7300 dni) - co ustalmy parametrem *-days*. Podając w parametrze *-extensions* frazę *v3_ca* - każemy tym samym użyć ustawień, które zawarte są w sekcji *[v3_ca]* pliku konfiguracyjnego. W sekcji tej określone są dodatkowe definicje, właściwe dla certyfikatów w formacie x.509. Dzięki nim jesteśmy w stanie narzucać różne ograniczenia.

Przykładowo, w rozszerzenie *basicConstraint* :

- parametr *CA:TRUE* – stanowi o tym, że certyfikat jest fragmentem struktury CA. Dla certyfikatów końcowych (serwera) parametr CA przyjmuje wartość FALSE

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

- parametr *pathLength*:<liczba> określa maksymalną liczbę certyfikatów pośrednich jakie mogą zostać podpisane przy pomocy certyfikatu z tym ograniczeniem.

Więcej informacji dotyczących formatu x.509 uzyskać można z RFC5280 (*Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*) <https://www.ietf.org/rfc/rfc5280.txt>

```
OpenSSL> req -config c:/ssl/openssl.cnf -key c:/ssl/private/keyCa.pem -new -x509 -days 7300 -sha256 -
extensions v3_ca -out c:/ssl/certs/ca.pem
```

Wykonując powyższą komendę, po podaniu tajnego hasła do zaszyfrowanego klucza - zostaniemy poddani serii pytań. Odpowiadamy na nie z myślą, że tworzymy swój własny Urząd Certyfikacji. Informacje tu wprowadzane są czysto formalne. Należy mieć jednak świadomość, że są to dane publiczne, opisujące wystawcę/odbiorcę certyfikatu do którego użytkownicy strony iRap, będą mieli wgląd. Nie należy podszywać się pod firmy trzecie. Można wykorzystać nazwę własnej firmy. Poniżej przykład wypełnienia formularza dla firmy APMAN S.A.

```
Enter pass phrase for c:/ssl/private/keyCa.pem: <podaj hasło którym zabezpieczyłeś klucz>
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Kraj (2 litery) [PL]:PL
Województwo []:Silesia
Miasto []:Katowice
Nazwa organizacji[]:APMAN S.A.
Nazwa jednostki organizacyjnej []:APMAN S.A. Certificate Authority
Nazwa własna (Common name) []:APMAN S.A. CA
Adres E-mail []:
```

Podobnie sprawa wygląda przy tworzeniu certyfikatu pośredniego. Szczególną ostrożność należy jednak zachować przy tworzeniu certyfikatu serwera – w polu „Nazwa własna (Common name)” konieczne jest podanie pełnej nazwy domeny dla której jest wydawany certyfikat. Błędna nazwa certyfikatu jest przez przeglądarki internetowe interpretowana jako błąd.

Utworzony certyfikat CA można sprawdzić poleceniem

```
OpenSSL> x509 -noout -text -in c:/ssl/certs/ca.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      a2:80:fa:a0:5e:65:b4:93
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=PL, ST=Silesia, L=Katowice, O=Kamssoft S.A., CN=KAMSOFT S.A. CA
    Validity
      Not Before: May 11 11:08:31 2017 GMT
      Not After : May 6 11:08:31 2037 GMT
    Subject: C=PL, ST=Silesia, L=Katowice, O=Kamssoft S.A., CN=KAMSOFT S.A. CA
```

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 16
--	-------------------------	------------	--------------	-----------

Subject Public Key Info:
 Public Key Algorithm: rsaEncryption
 Public-Key: (4096 bit)
 Modulus:
 00:de:80:75:99:cb:e9:6f:f5:28:d1:05:1d:e6:b2:
 [...]

Za sprawą parametru `-noout -text` odpowiedź polecenia `x.509` została wydrukowana na ekran konsoli w postaci czytelnego tekstu. Z jej treści można odczytać kto jest wystawcą (Issuer); jaki jest okres obowiązywania certyfikatu; dla kogo wystawiono certyfikat (w naszym przypadku sami sobie go wystawiliśmy); dane klucza publicznego; parametry dodatkowe dla certyfikatów `x.509`.

Po utworzeniu każdego certyfikatu, warto sprawdzić czy dane w nim zawarte są poprawne i czy pokrywają się z ustawieniami pliku konfiguracyjnego.

Generowanie certyfikatu pośredniego.

W tym rozdziale wygenerujemy certyfikat pośredni, po czym połączymy go z certyfikatem nadrzędnym tworząc łańcuch zaufania. W ten sposób utworzymy plik `root.pem`, który finalnie wkleimy do katalogu roboczego `iRap`.

Aby wygenerować certyfikat pośredni CA musimy utworzyć strukturę katalogów odpowiadającą opisowi zawartemu w rozdziale „Przygotowanie przestrzeni roboczej”. Zawartość katalogu `C:/ssl/int` odpowiada konfiguracji certyfikatu głównego. Dodatkowo, oprócz katalogów `certs`, `newcerts` i `private` tworzymy katalog `csr`, który będzie przechowywał nasze żądania wydania certyfikatu - CSR. Należy spreparować plik konfiguracyjny (`c:/ssl/int/openssl.cnf`) na podstawie załącznika B. Podobnie jak wcześniej tworzymy puste pliki `index.txt` i `serial`.

Tym razem wcielamy się w jednostkę organizacyjną podlegającą pod nasz własny Urząd Certyfikacji. W pierwszym kroku generujemy klucz prywatny. Następnie tworzymy żądanie certyfikacji CSR i prosimy aby oddział główny nam go podpisał. Ostatecznie przy pomocy certyfikatu CA (głównego) posłużymy się do utworzenia certyfikatu pośredniego CA.

Generowanie klucza prywatnego pośredniego certyfikatu CA

```
OpenSSL> genrsa -aes256 -out c:/ssl/int/private/keyCA.pem 4096
```

Tworzenie CSR certyfikatu pośredniego CA

Szczególną uwagę należy zwrócić na parametr *-config* dla którego zmieniono ścieżkę certyfikacji. Czekają nas też ta sama seria pytań co przy tworzeniu certyfikatu głównego. Odpowiadamy na nie np. jako Pośredni Urząd Certyfikacji.

Common name / Nazwa własna –wartość ta musi być unikalna. To znaczy, że łańcuch certyfikacji nie może zawierać w sobie dwóch certyfikatów o tej samej nazwie.

```

OpenSSL> req -config c:/ssl/int/openssl.cnf -new -sha256 -key c:/ssl/int/private/keyICA.pem -out
c:/ssl/int/csr/csrICA.pem
Enter pass phrase for c:/ssl/int/private/keyICA.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Kraj (2 litery) [PL]:PL
Województwo []:Silesia
Miasto []:Zabrze
Nazwa organizacji []:APMAN S.A.
Nazwa jednostki organizacyjnej []:APMAN S.A. Intermediate Certificate Authority
Nazwa własna(Common name)* []:APMAN S.A. ICA
Adres E-mail []:apmansa_ica@gmail.com

```

Przy pomocy tak utworzonego CSR możemy w imieniu głównego Urzędu Certyfikacji wygenerować certyfikat pośredni. Aby wrócić do roli głównego Urzędu Certyfikacji, tym razem w parametrze *-config* musimy podać ścieżkę do pliku konfiguracyjnego *c:/ssl/openssl.cnf*. W pliku tym zawarta jest informacja o lokalizacji klucza prywatnego i certyfikatu CA – Urzędu certyfikacji.

```

# The root key and root certificate.
private_key    = $dir/private/keyCA.pem
certificate     = $dir/certs/CA.pem

```

Wykonujemy polecenie „ca” ... jeżeli wystąpił błąd „*error while loading serial numer*” – to spróbuj plik *serial* utworzyć zgodnie z wytycznymi rozdziału „*Przygotowanie przestrzeni roboczej*”

```

OpenSSL> ca -config c:/ssl/openssl.cnf -extensions v3_int_ca -days 3650 -notext -md sha256 -in
c:/ssl/int/csr/csrICA.pem -out c:/ssl/int/certs/ICA.pem
Using configuration from c:/ssl/openssl.cnf
Enter pass phrase for C:/ssl/private/keyCA.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4096 (0x1000)
  Validity
    Not Before: May 12 13:01:41 2017 GMT
    Not After : May 10 13:01:41 2027 GMT
  Subject:
    countryName           = PL
    stateOrProvinceName  = Silesia
    organizationName      = APMAN S.A.

```

```

organizationalUnitName = APMAN S.A. Intermediate CA
commonName             = APMAN S.A. ICA
emailAddress           = apman@gmail.com
X509v3 extensions:
X509v3 Subject Key Identifier:
    38:BB:D7:68:33:F6:B3:14:9C:36:9B:5A:4E:9B:D6:CF:FD:A0:30:43
X509v3 Authority Key Identifier:
    keyid:96:EA:2A:4E:85:8F:85:B1:C0:8D:81:01:03:61:7F:68:73:D9:03:
[...]
```

Jeżeli nie wystąpi żaden błąd. Na ekran wydrukowane zostaną dane certyfikatu. Należy je sprawdzić, zaakceptować okres ważności certyfikatu (klawisz „Y”) i zatwierdzić fakt generowania certyfikatu (klawisz „Y”).

```

[...]
```

Certificate is to be certified until May 10 13:01:41 2027 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

Podczas generowania pośredniego certyfikatu CA - użyto innej sekcji konfiguracji do wprowadzenia rozszerzonych informacji o certyfikacie – [v3_int_ca]. Nałożono w niej ograniczenie,

basicConstraints = critical, CA:true, pathlen:0

kóre w porównaniu do ustawień z certyfikatu głównego zawiera właściwość pathlen:0. Ta różnica spowoduje że nasz nowy certyfikat pośredni będzie służył tylko do tworzenia certyfikatów końcowych.

Aby zweryfikować czy certyfikat został poprawnie wydany przez certyfikat nadrzędny można wykonać polecenie:

```

OpenSSL> verify -CAfile c:/ssl/certs/ca.pem c:/ssl/int/certs/ica.pem
c:/ssl/int/certs/ica.pem: OK
```


Generowanie łańcucha certyfikacji

W ostatnim kroku musimy złączyć certyfikat główny z certyfikatem pośrednim

- 1) Tworzymy plik root.pem.
- 2) Otwieramy plik root.pem w edytorze tekstu (np. notepad++)
- 3) Wklejamy do niego zawartość certyfikatu pośredniego (C:/ssl/int/certs/ica.pem)
- 4) Doklejamy na końcu zawartość certyfikatu głównego (C:/ssl/certs/ca.pem)
- 5) Zapisujemy plik.

Generowanie certyfikatu serwera iRap.

Proces generowania certyfikatu SSL serwera jest podobny do tworzenia certyfikatu pośredniego. Na początku (z perspektywy Klienta) tworzymy klucz prywatny i żądanie wystawienia certyfikatu. Na tym etapie możemy

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

utworzone żądanie, wysłać do Urzędu Certyfikacji, bądź też samemu podpisać żądanie przy pomocy wcześniej przygotowanemu pośredniemu certyfikatowi CA.

W naszym przypadku, sami podpiszemy certyfikat SSL

Tworzymy klucz prywatny – *key.pem*. Bierze on udział w inicjalizacji sesji (*TLS Handshake*) podczas transmisji danych. Dlatego, tym razem zostanie on skrócony do 2048 bitów - ze względów wydajnościowych.

```
OpenSSL> genrsa -aes256 -out c:/ssl/int/private/key.pem 2048
```

Generujemy CSR – żądanie wydania certyfikatu dla naszej domeny www.apman.pl

Podczas tworzenia CSR podajemy dane firmy wnioskującej o wydanie certyfikatu Zamawiany Certyfikat SSL wydawany jest dla konkretnej domeny. Ważne aby w nazwie własnej (parametr CN – ang. *Common name*) podać właściwy adres domeny np.www.apman.pl. Przeglądarki internetowe generują błąd w sytuacji gdy na stronie WWW znajdzie się certyfikat wydany dla innej domeny.

```
OpenSSL> req -reqexts SAN -config c:/ssl/int/openssl.cnf -key c:/ssl/int/private/key.pem -new -sha256 -out c:/ssl/int/csr/csrAPMAN.pem
```

```
Enter pass phrase for c:/ssl/int/private/keyCA.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Kraj (2 litery) [PL]:PL
Wojewodztwo []:Silesia
Miasto []:Katowice
Nazwa organizacji []:APMAN S.A.
Nazwa jednostki organizacyjnej []:APMAN S.A Dział sprzedaży internetowej
Nazwa własna(Common name)* []:www.apman.pl
Adres E-mail []:apman@gmai.com
```


Pozostało sfinalizować proces generowania certyfikatu SSL. Przy pomocy certyfikatu pośredniego CA podpiszemy nasz CSR. Z reguły certyfikat wydawany jest na rok. Zanim okres ważności dobiegnie końca należy „odnowić certyfikat”. W praktyce, sprowadza się to do ponownego przeprowadzenia procesu przeprowadzonego w tym rozdziale. Tworzymy CSR, podpisujemy go sami, albo wysyłamy do Urzędu Certyfikacji. W tym drugim przypadku czasem wykonywana jest ponowna walidacja podmiotu zamawiającego.

Wykonujemy polecenie utworzenia pliku *cert.pem*

```
ca -config c:/ssl/int/openssl.cnf -extensions server_cert -days 364 -notext -md sha256 -in c:/ssl/int/csr/csrAPMAN.pem -out c:/ssl/int/certs/cert.pem
```

Jeżeli nie wystąpi żaden błąd. Na ekran wydrukowane zostaną dane certyfikatu. Należy je sprawdzić i potwierdzić chęć podpisania certyfikatu klawiszem 'Y'. Kolejny wiersz również potwierdzamy klawiszem 'Y'.

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 20
--	-------------------------	------------	--------------	-----------

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

Tym razem, użyto sekcji [server_cert] pliku konfiguracyjnego openssl.cnf, w której zapisane są dodatkowe informacje dla certyfikatu SSL

-extensions server_cert

Analizując konfigurację zauważyć można zmianę parametru *keyUsage* – ustawiono w nim flagę *keyEncipherment*. Co oznacza, że klucz publiczny certyfikatu będzie brał udział w szyfrowaniu kluczy sesji podczas inicjalizacji zabezpieczonego połączenia. Wszystkie parametry są opisane w RFC 5280 o którym wspomniano już w poprzednim rozdziale.

Możliwe jest dodanie alternatywnych nazw dla swojej domeny. W tym celu należy uzupełnić w pliku `c:/ssl/int/openssl.cnf` sekcję *alt_names*

```
[ alt_names ]
DNS.1 = localhost
IP.1 = 127.0.0.1
```

- dodając kolejne wpisy DNS.2 lub IP.2 ...

Rozwiązaniem takie szczególnie będzie przydatne gdy dana apteka nie będzie dysponowała domeną dla serwisu iRap. Można po prostu wskazać w tym miejscu swój zewnętrzny adres IP. W przypadku gdy odwołujemy się do serwisu poprzez adres inny niż określony w certyfikacie SSL - otrzymujemy błąd typu „Nieznany Urząd Certyfikacji” (*Unknown Issuer*).

W wyniku wykonania powyższych ćwiczeń - wygenerowaliśmy wszystkie potrzebne elementy do konfiguracji programu iRAP, Posiadamy pliki *root.pem*, *key.pem*, *cert.pem*, oraz znamy tajne hasło do klucha prywatnego (*key.pem*).

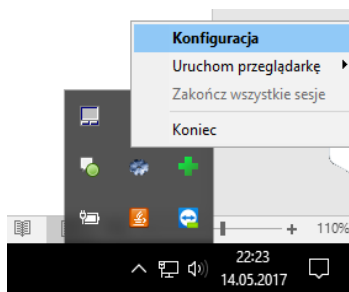
Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 21
--	-------------------------	------------	--------------	-----------

Konfiguracja iRap

UWAGA! Wszelkie zmiany w konfiguracji iRap oraz pliku apman.ini – muszą zostać zakończone ponownym uruchomieniem programu.

Serwer iRap oferuje różne tryby transmisji danych. Dostępne są: połączenie zwykłe, szyfrowane (SSL/TLS) oraz mieszane. Aby połączenie było zabezpieczone, wymagane jest dostarczenie pliku z certyfikatem SSL serwera (*cert.pem*), pliku z certyfikatem głównym (*root.pem*), pliku z kluczem prywatnym (*key.pem*) oraz tajnego hasła zabezpieczającego ten klucz. W pierwszej kolejności należy zmienić hasło w konfiguracji iRap. Następnie pliki umieszczamy w katalogu IRAP (C:\KS\APW\IRAP).

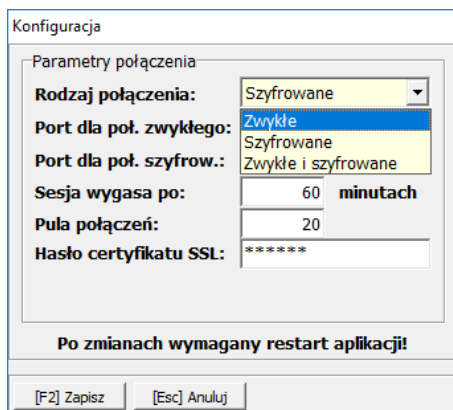
Dostęp do panelu konfiguracyjnego można uzyskać klikając w ikonę programu iRap (zielony krzyż), która znajduje się w zasobniku systemowym (ang. system tray). Alternatywną ścieżką jest wybór opcji *Konfiguracja*, która kryje się w menu podręcznym programu iRap




Rys. 15 Ikona iRap z zasobniku systemowym wraz z opcją konfiguracji iRap dostępną w menu podręcznym.

Jeżeli polityka firmy zakłada restrykcyjne procedury zabezpieczające należy w panelu *Konfiguracja* wybrać wyłącznie połączenie szyfrowane.

W następnych krokach ustawiamy port dla połączenia szyfrowanego oraz hasło certyfikatu SSL do klucza prywatnego.



Rys.16 Panel konfiguracyjny modułu iRap.

	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

Konfiguracja obsługiwanych protokołów SSL/TLS

Jak powszechnie wiadomo starsze wersje SSL\TLS narażone są na różnego rodzaju ataki.

SSL v2 – protokół podatny jest na atak typu DROWN (*Decrypting RSA with Obsolete and Weakened eNcryption*). Aby do niego doszło najpierw atakujący przeprowadza atak typu MITM (*Man In The Middle*), czyli musi znaleźć sposób, aby pośredniczyć w komunikacji między serwerem a klientem. Następnie przy wymuszeniu szyfrowania SSLv2 z kluczami RSA, atakujący wysyła spreparowane pakiety aby odszyfrować komunikację. Opis metody jest dostępny na <https://drownattack.com/>

SSL v3 – protokół nie powinien być używany z powodu ataku typu POODLE (*Padding Oracle On Downgraded Legacy Encryption*). Atakujący musi uzyskać dostęp do komunikacji między serwerem a klientem. Korzystając z protokołu SSL v3 napastnik wysyła serię zapytań, z pozoru nieistotnych, które mają na celu uzyskaniu odpowiedzi. Odpowiedzi te, z kolei posłużą jako próbki na których wykonana zostanie analiza prowadząca do odszyfrowania plików cookie, w których mogą być przechowywane poufne dane. Więcej informacji na stronie - <https://zaufanatrzeciastrona.pl/post/niebezpieczny-pudelek-atakujecie-czyli-wyjasnienie-bledu-w-sslv3/>

TLS 1.0 – Protokół może ulec atakom typu BEAST (Browser Exploit Against SSL/TLS). W tym przypadku ofiara musi uruchomić złośliwy kod JavaScript emitujący znane napastnikowi dane. Informacje te po zakodowaniu przez serwer www, zostają ponownie przechwycone i zdekodowane. Poznając sposób szyfrowania spreparowanych danych atakujący może odszyfrować pozostałe dane. Więcej informacji na

<http://www.computerworld.pl/news/Bestia-grozna-dla-stron-z-SSL,375612.html>

TLS 1.1 oraz TLS 1.2 są protokołami uważanymi za bezpieczne. Chociaż, w obu przypadkach nieznanie są jeszcze żadne ataki, to wersja 1.2 realizuje nowsze algorytmy kryptograficzne. Użycie wyłącznie protokołów TLS 1.2, choć zwiększa poziom bezpieczeństwa może spowoduje, że na niektórych przestarzałych przeglądarkach - strona www iRap nie będzie dostępna.

Na chwilę obecną, ze względu na konieczność kompatybilności modułu iRap z hurtowniami, konieczne jest użycie protokołu TLS 1.0. Z tego powodu **domyślnie iRap jest włączony w trybie zgodności z TLS 1.0/TLS 1.1/ TLS 1.2.**

Administrator serwera iRap ma możliwość zmiany stosowanych protokołów SSL/TLS poprzez edycję pliku konfiguracyjnego apman.ini (znajdującego się w głównym katalogu programu KS-Apteka). W sekcji [OPCJE] należy dodać/zmodyfikować parametr SSL_VERSIONS - z listą obsługiwanych protokołów. Poniżej przykład wyczerpujący wszystkie możliwości:

```
SSL_VERSIONS= SSLv2,SSLv23,SSLv3,TLSv1,TLSv11,TLSv12
```

Konfiguracja Cipher list

Ostatnim elementem konfiguracji jest ustalenie listy dozwolonych zestawień algorytmów szyfrujących. Więcej informacji na temat samego zagadnienia umieszczono w osobnym rozdziale pt. „Lista zestawień algorytmów szyfrujących (cipher list)”. IRap pozwala na modyfikację listy przy pomocy pliku apman.ini – poprzez

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 23
--	-------------------------	------------	--------------	-----------

dodanie w sekcji [OPCJE] parametru SSL_CIPHER_LIST. Domyślna konfiguracja zakłada użycie najsilniejszych dostępnych algorytmów szyfrujących i wymiany kluczy.

SSL_CIPHER_LIST=AES256+EECDH:AES256+EDH

Przyjęte rozwiązanie, ogranicza listę wspieranych urządzeń (przeglądarek internetowych). Dlatego w przypadku gdy, konieczne będzie dopuszczenie mniej restrykcyjnych ustawień, czy to też w celu zwiększenie kompatybilności - można próbę konfiguracji rozpocząć od poniższego zestawienia

SSL_VERSIONS= TLSv10,TLSv11,TLSv12

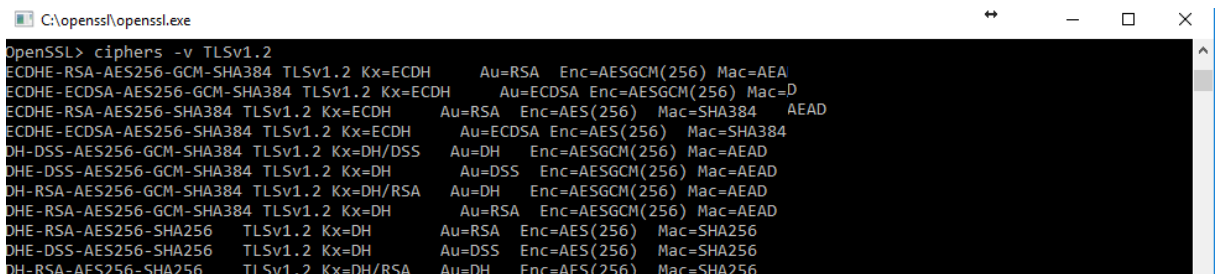
SSL_CIPHER_LIST=ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!MD5:!DSS

Dobór parametrów został opracowany z myślą o zapewnieniu możliwie wysokiego poziomu bezpieczeństwa przy zachowaniu niezbędnej kompatybilności z pozostałymi modułami systemu. Więcej informacji odnośnie stosowania zabezpieczeń w Internecie, można uzyskać w artykule pt. „SSL and TLS Deployment Best Practices” który został opublikowany przez portal SSL Labs na stronie <https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices>

Listę dostępnych w OpenSSL zestawień algorytmów szyfrujących można wypisać na ekran przy pomocy polecenia:

```
OpenSSL> ciphers -v TLSv1.2
```

Parametr dodatkowy TLSv1.2 zawęża zakres cipher list do zestawień obsługiwanych przez zadany protokół. Jego pominięcie spowoduje wypisanie pełnej listy dostępnych zestawień algorytmów szyfrujących.



```

C:\openssl\openssl.exe
OpenSSL> ciphers -v TLSv1.2
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH    Au=RSA  Enc=AESGCM(256) Mac=AEA
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH    Au=ECDSA Enc=AESGCM(256) Mac=D
ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH    Au=RSA  Enc=AES(256) Mac=SHA384 AEAD
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH    Au=ECDSA Enc=AES(256) Mac=SHA384
DH-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH/DSS  Au=DH   Enc=AESGCM(256) Mac=AEAD
DHE-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH     Au=DSS  Enc=AESGCM(256) Mac=AEAD
DH-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH/RSA Au=DH   Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH     Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-SHA256 TLSv1.2 Kx=DH     Au=RSA  Enc=AES(256) Mac=SHA256
DHE-DSS-AES256-SHA256 TLSv1.2 Kx=DH     Au=DSS  Enc=AES(256) Mac=SHA256
DH-RSA-AES256-SHA256 TLSv1.2 Kx=DH/RSA Au=DH   Enc=AES(256) Mac=SHA256

```

Rys. 17 Wykonanie polecenia listującego dostępne zestawy algorytmów szyfrujących

Na podstawie listy można sporządzić wartość parametru SSL_CIPHER_LIST. Wybrane zestawienia oddzielamy znakiem dwukropka. Możliwe jest stosowanie notacji ‘zawierających’.

Przykładowo:

AES256+EECDH

Jest równoznaczne z poniższą listą:

ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA

Co można sprawdzić poleceniem programu Open SSL

OpenSSL> **ciphers** AES256+EECDH

Dodatkowo dozwolone jest stosowanie wykluczenia np. !DSS co spowoduje wykluczenie zestawień zawierających sygnaturę DSS.

Ocena poziomu bezpieczeństwa na serwerze iRap.

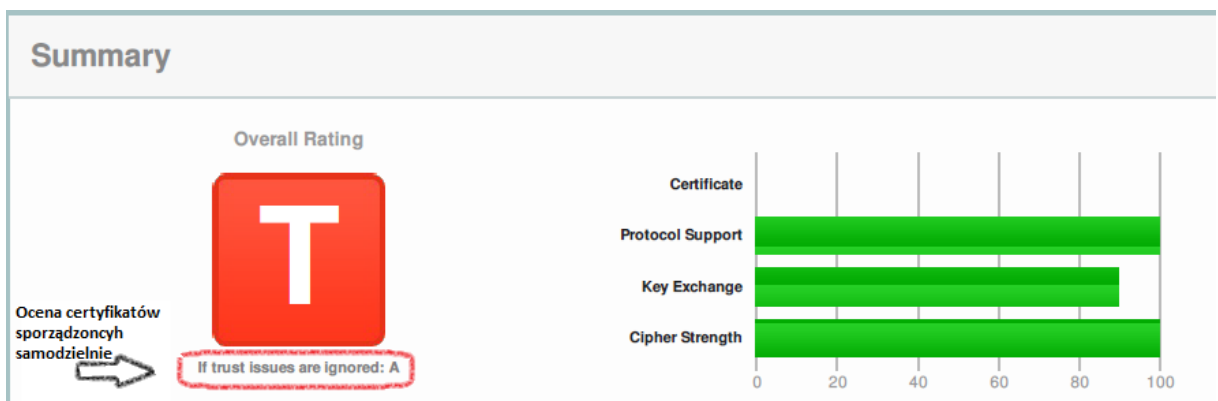
Firma Qualyas Inc, oferuje wszystkim zainteresowanym podmiotom dostęp do niekomercyjnego projektu badawczego SSL Labs. Przedsięwzięcie jest rzetelnym źródłem wiedzy o stosowanych obecnie zabezpieczeniach sieci Internet. Projekt dostarcza wiele narzędzi służących m.in. usprawnianiu zabezpieczeń na serwerach. Jednym z nich jest darmowa usługa internetowa (<https://www.ssllabs.com/ssltest/>), która potrafi przeprowadzić szczegółową analizę konfiguracji dowolnego serwera.




Rys. 18 Usługa internetowa, służąca do oceny zabezpieczeń serwera www. Adres: <https://www.ssllabs.com/ssltest/>

Aby wykonać test, konieczne jest udostępnienie witryny iRap w Internecie z wykorzystaniem nazwy domenowej. W celach testowych skorzystano z portalu *nazwa.pl*, który umożliwił dostęp do darmowych domen. Po zarejestrowaniu domeny, administrator musi jedynie ustawić w konfiguracji „Przekierowanie na zewnętrzny adres IP”.

W Załączniku C: „Raport SSL Labs dla domyślnej konfiguracji serwera iRap” - umieszczono kompletny wynik testów serwera iRap w jego domyślnej konfiguracji. Stronie przyznano wysoką ocenę „A” - przy pominięciu kwestii zaufania certyfikatu głównego. Ocenę „T” uzyskują wszystkie witryny, które nie mają zakupionego certyfikatu SSL w Urzędzie Certyfikacji.



	INSTRUKCJA SERWISOWA				KS-AOW
	Polityka bezpieczeństwa w module iRap				
ISO 9001:2008	Dokument: 2017.05.16	Wydanie: 1		Waga: 90	

Rys. 19 Wynik testu zabezpieczeń wykonanym na portalu SSL Labs.

W przypadku gdy Administrator będzie dokonywał zmian w konfiguracji serwera iRap zalecane jest wykonanie powyżej opisanych testów.

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 26
--	-------------------------	------------	--------------	-----------

Załącznik A. Plik konfiguracyjny głównego certyfikatu CA

Nazwa pliku: C:/ssl/openssl.cnf.

```
# OpenSSL root CA configuration file.
# Copy to `/root/ca/openssl.cnf`.

[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
dir          = C:/ssl
certs        = $dir/certs
#crl_dir     = $dir/crl
new_certs_dir = $dir/newcerts
database     = $dir/index.txt
serial       = $dir/serial
#RANDFILE   = $dir/private/.rand

# The root key and root certificate.
private_key  = $dir/private/keyCA.pem
certificate   = $dir/certs/CA.pem

# SHA-1 is deprecated, so use SHA-2 instead.
default_md   = sha256

name_opt     = ca_default
cert_opt     = ca_default
default_days = 375
preserve     = no
policy       = policy_strict

[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName     = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName      = supplied
emailAddress     = optional

[ req ]
# Options for the `req` tool (`man req`).
default_bits     = 2048
distinguished_name = req_distinguished_name
string_mask      = utf8only

# SHA-1 is deprecated, so use SHA-2 instead.
default_md       = sha256

# Extension to add when the -x509 option is used.
x509_extensions  = v3_ca

[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName      = Kraj (2 litery)
```



stateOrProvinceName = Wojewodztwo
localityName = Miasto
O.organizationName = Nazwa organizacji
organizationalUnitName = Nazwa jednostki organizacyjnej
commonName = Nazwa własna(Common name)*
emailAddress = Adres E-mail

Optionally, specify some defaults.

countryName_default = PL
#stateOrProvinceName_default =
#localityName_default =
#O.organizationName_default =
#organizationalUnitName_default =
#emailAddress_default =

[v3_ca]

Extensions for a typical CA (`man x509v3_config`).

subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[v3_int_ca]

Extensions for a typical intermediate CA (`man x509v3_config`).

subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

Załącznik B. Plik konfiguracyjny pośredniego certyfikatu CA

Nazwa pliku: C:/ssl/int/openssl.cnf.

```
# OpenSSL intermediate CA configuration file.
# Copy to `root/ca/intermediate/openssl.cnf`.

[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
dir          = C:/ssl/int
certs        = $dir/certs
database     = $dir/index.txt
serial       = $dir/serial
new_certs_dir = $dir/newcerts
# RANDFILE   = $dir/private/.rand

# The root key and root certificate.
private_key  = $dir/private/keyICA.pem
certificate   = $dir/certs/ICA.pem

# SHA-1 is deprecated, so use SHA-2 instead.
default_md   = sha256

name_opt     = ca_default
cert_opt     = ca_default
default_days = 375
preserve     = no
policy       = policy_loose

[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName     = optional
stateOrProvinceName = optional
localityName     = optional
organizationName = optional
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional

[ SAN ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = localhost
IP.1  = 127.0.0.1

[ req ]
# Options for the `req` tool (`man req`).
default_bits      = 2048
distinguished_name = req_distinguished_name
string_mask       = utf8only

# SHA-1 is deprecated, so use SHA-2 instead.
default_md        = sha256
```



```
[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName          = Kraj (2 litery)
stateOrProvinceName  = Wojewodztwo
localityName         = Miasto
O.organizationName    = Nazwa organizacji
organizationalUnitName = Nazwa jednostki organizacyjnej
commonName           = Nazwa własna(Common name)*
emailAddress          = Adres E-mail

# Optionally, specify some defaults.
countryName_default   = PL
stateOrProvinceName_default =
localityName_default =
O.organizationName_default =
organizationalUnitName_default =
emailAddress_default =

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
subjectAltName = @alt_names
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

Załącznik C: Raport SSL Labs dla domyślnej konfiguracji serwera iRap

Test wykonano na domyślnych ustawieniach programu iRap:

Protokół TLS 1.0/TLS 1.1/TLS 1.2; Cipher list = AES256+EECDH:AES256+EDH; Połączenie tylko szyfrowane; Certyfikat z pełnym łańcuchem zaufania. Root Ca i Intermediate CA (AES 256) z 4096 bitowym kluczem RSA. Certyfikat serwera oparty na 2048 bitowym kluczu RSA.



[Home](#) [Projects](#) [Qualys.com](#) [Contact](#)

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > www.apman.pl

SSL Report: [www.apman.pl](#) (178.212.44.105)

Assessed on: Mon, 19 Jun 2017 10:18:00 UTC | [Hide](#) | [Clear cache](#)

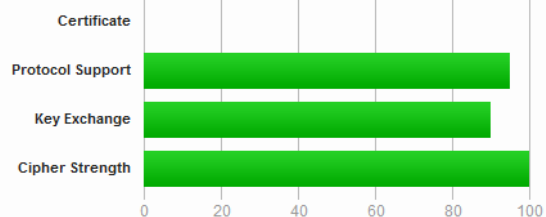
[Scan Another »](#)

Summary

Overall Rating



If trust issues are ignored: A



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This server's certificate is not trusted, see [below](#) for details.

Certificate #1: RSA 2048 bits (SHA256withRSA)



Server Key and Certificate #1

Subject	www.apman.pl Fingerprint SHA256: c82e7f89726c9d991b21604dcf00c8b4894b183b9ddb7784a8d8728fa876e3b4 Pin SHA256: Jx0GMOqSNmFVD0fz73ITVuEd8P+rUA+jK8+u2EkeXA=
Common names	www.apman.pl
Alternative names	-
Valid from	Mon, 19 Jun 2017 09:55:05 UTC
Valid until	Mon, 18 Jun 2018 09:55:05 UTC (expires in 11 months and 29 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	KAMSOFT S.A. ICA
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	No
OCSP Must Staple	No
Revocation information	None

DNS CAA No ([more info](#))

Trusted No **NOT TRUSTED** ([Why?](#))

Additional Certificates (if supplied) [↓](#)

Certificates provided 3 (4554 bytes)

Chain issues Contains anchor

#2

Subject	KAMSOFT S.A. ICA
Fingerprint SHA256:	82c1adfa75cef8a6865c9cca7b033e890d807b8d13e637f603154bc214cc2e9c
Pin SHA256:	1S2Kc+a+F0vxNTDh2eZJFSnw3RBZKxXjpxpMluhlw0=
Valid until	Thu, 17 Jun 2027 09:54:37 UTC (expires in 9 years and 11 months)
Key	RSA 4096 bits (e 65537)
Issuer	KAMSOFT S.A. CA
Signature algorithm	SHA256withRSA

#3

Subject	KAMSOFT S.A. CA Not in trust store
Fingerprint SHA256:	c20c31b33613bab8f9145b52c91303b6a4a918c9a6160777d74360ba3282940b
Pin SHA256:	HhILK96DnZHjzpQre2KxbFpOX7R+1fXXBI0EUkPyfKw=
Valid until	Sun, 14 Jun 2037 09:45:21 UTC (expires in 19 years and 11 months)
Key	RSA 4096 bits (e 65537)
Issuer	KAMSOFT S.A. CA Self-signed
Signature algorithm	SHA256withRSA

Certification Paths [+](#)

[Click here to expand](#)

Configuration

Protocols

TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

Cipher Suites

TLS 1.2 (suites in server-preferred order) [-](#)

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)	ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	ECDH secp256r1 (eq. 3072 bits RSA) FS	256

TLS 1.1 (we could not determine if the server has a preference) [+](#)

TLS 1.0 (we could not determine if the server has a preference) [+](#)

Handshake Simulation

[Android 2.3.7](#) No SNI² Server sent fatal alert: handshake_failure

[Android 4.0.4](#) RSA 2048 (SHA256) TLS 1.0 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS

[Android 4.1.1](#) RSA 2048 (SHA256) TLS 1.0 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS



Android 4.2.2	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Android 4.3	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Android 4.4.2	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Android 5.0.0	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Android 6.0	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Android 7.0	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Baidu Jan 2015	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
BingPreview Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Chrome 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Chrome 51 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Firefox 31.3.0 ESR / Win 7	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Firefox 47 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Firefox 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Firefox 49 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Googlebot Feb 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
IE 6 / XP No FS ¹ No SNI ²			Server closed connection		
IE 7 / Vista	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
IE 8 / XP No FS ¹ No SNI ²			Server sent fatal alert: handshake_failure		
IE 8-10 / Win 7 R	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
IE 11 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
IE 11 / Win 8.1 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
IE 10 / Win Phone 8.0	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
IE 11 / Win Phone 8.1 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
IE 11 / Win Phone 8.1 Update R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
IE 11 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Edge 13 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Edge 13 / Win Phone 10 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Java 6u45 No SNI ²			Server sent fatal alert: handshake_failure		
Java 7u25			Server sent fatal alert: handshake_failure		
Java 8u31			Server sent fatal alert: handshake_failure		
OpenSSL 0.9.8y			Server sent fatal alert: handshake_failure		
OpenSSL 1.0.1i R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
OpenSSL 1.0.2e R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 5.1.9 / OS X 10.6.8	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Safari 6 / iOS 6.0.1	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
Safari 6.0.4 / OS X 10.8.4 R	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDH secp256r1	FS
Safari 7 / iOS 7.1 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
Safari 7 / OS X 10.9 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
Safari 8 / iOS 8.4 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
Safari 8 / OS X 10.10 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDH secp256r1	FS
Safari 9 / iOS 9 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 9 / OS X 10.11 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 10 / iOS 10 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 10 / OS X 10.12 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Apple ATS 9 / iOS 9 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Yahoo Slurp Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
YandexBot Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS





(1) Clients that do not support Forward Secrecy (FS) are excluded when determining support for it.

(2) No support for virtual SSL hosting (SNI). Connects to the default site if the server uses SNI.

(3) Only first connection attempt simulated. Browsers sometimes retry with a lower protocol version.

(R) Denotes a reference browser or client, with which we expect better effective security.

(All) We use defaults, but some platforms do not use their best protocols and features (e.g., Java 6 & 7, older IE).

		Protocol Details
		No, server keys and hostname not seen elsewhere with SSLv2
		(1) For a better understanding of this test, please read this longer explanation
		(2) Key usage data kindly provided by the Censys network search engine; original DROWN test here
		(3) Censys data is only indicative of possible key and certificate reuse; possibly out-of-date and not complete
DROWN		
Secure Renegotiation		Supported
Secure Client-Initiated Renegotiation		Yes
Insecure Client-Initiated Renegotiation		No
BEAST attack		Not mitigated server-side (more info) TLS 1.0: 0x0014
POODLE (SSLv3)		No, SSL 3 not supported (more info)
POODLE (TLS)		No (more info)
Downgrade attack prevention		Yes, TLS_FALLBACK_SCSV supported (more info)
SSL/TLS compression		No
RC4		No
Heartbeat (extension)		Yes
Heartbleed (vulnerability)		No (more info)
Ticketbleed (vulnerability)		No (more info)
OpenSSL CCS vuln. (CVE-2014-0224)		No (more info)
OpenSSL Padding Oracle vuln. (CVE-2016-2107)		No (more info)
Forward Secrecy		Yes (with most browsers) ROBUST (more info)
ALPN		No
NPN		No
Session resumption (caching)		Yes
Session resumption (tickets)		Yes
OCSP stapling		No
Strict Transport Security (HSTS)		No
HSTS Preloading		Not in: Chrome Edge Firefox IE
Public Key Pinning (HPKP)		No (more info)
Public Key Pinning Report-Only		No
Public Key Pinning (Static)		No (more info)
Long handshake intolerance		No
TLS extension intolerance		No
TLS version intolerance		No
Incorrect SNI alerts		No
Uses common DH primes		No, DHE suites not supported
DH public server param (Ys) reuse		No, DHE suites not supported
ECDH public server param reuse		No
Supported EC Named Curves		secp256r1
SSL 2 handshake compatibility		Yes
		HTTP Requests 
		https://www.apman.pl/ (HTTP/1.1 200 OK)
		Miscellaneous
Test date		Mon, 19 Jun 2017 10:15:36 UTC
Test duration		143.686 seconds
HTTP status code		200
HTTP server signature		-
Server hostname		host-178.212.44.185-internet.zabrze.debacom.pl

Załącznik D: Instrukcja skrócona – generowanie certyfikatu SSL i konfiguracja IRAP

- 1) Pobierz strukturę katalogów z <ftp://ftp.kamsoft.pl/pub/KS-APW/2017/Inne/ssl.zip> i rozpakuj, na dysku C: (tak że plik konfiguracyjny będzie dostępny w C:/ssl/openssl.cnf)
- 2) Dodaj zmienne środowiskowe:
Naciśnij klawisz Windows> wpisz z klawiatury CMD > Otwórz jako administrator (prawym klawiszem) wiersz poleceń > Wykonaj polecenia:

```
setx /m RANDFILE "C:\ssl\rnd"
setx /m OPENSSL_CONF "C:\ssl\openssl.cnf"
```

- 3) Utwórz certyfikat CA. Otwórz **C:/ssl/openssl.exe**

Wygeneruj parę kluczy RSA dla CA. Zapamiętaj hasło (1) aby podpisać pośredni certyfikat	genrsa -aes256 -out c:\ssl\private\keyCA.pem 4096
Tworzenie certyfikatu CA ważny przez od 20 lat. Dane wpisujemy jakbyśmy tworzyli własny Urząd certyfikacji. Podaj hasło (1)	req -config c:/ssl/openssl.cnf -key c:/ssl/private/keyCa.pem -new -x509 -days 7300 -sha256 -extensions v3_ca -out c:/ssl/certs/ca.pem
Sprawdzenie certyfikatu CA	x509 -noout -text -in c:/ssl/certs/ca.pem

- 4) Utwórz pośredni certyfikat CA

Wygeneruj parę kluczy RSA dla ICA. Zapamiętaj hasło (2) aby podpisać certyfikat serwera	genrsa -aes256 -out c:/ssl/int/private/keyICA.pem 4096
Tworzenie pośredniego certyfikatu ICA. Dane wpisujemy jakbyśmy byli jednostką podrzędną Urzędu Certyfikacji. Podaj hasło (2)	req -config c:/ssl/int/openssl.cnf -new -sha256 -key c:/ssl/int/private/keyICA.pem -out c:/ssl/int/csr/csrICA.pem
W roli Urzędu certyfikacji podpisujemy certyfikat pośredni. Podaj hasło (1)	ca -config c:/ssl/openssl.cnf -extensions v3_int_ca -days 3650 -notext -md sha256 -in c:/ssl/int/csr/csrICA.pem -out c:/ssl/int/certs/ICA.pem
Sprawdzenie certyfikatu ICA	x509 -noout -text -in c:/ssl/int/certs/ICA.pem

- 5) Utwórz plik **root.pem**, Wklej do niego najpierw certyfikat pośredni a następnie główny.
- 6) Utwórz certyfikat serwera cert.pem,

Wygeneruj parę kluczy RSA. Zapamiętaj hasło (3) - należy je podać w konfiguracji iRap.	genrsa -aes256 -out c:/ssl/int/private/ key.pem 2048
Generujemy żądanie podpisania certyfikatu dla naszej domeny. Podaj hasło (3)	req -reqexts SAN -config c:/ssl/int/openssl.cnf -key c:/ssl/int/private/key.pem -new -sha256 -out c:/ssl/int/csr/csrAPMAN.pem
Podpisujemy certyfikat serwera przy pomocy certyfikatu pośredniego CA. Podaj hasło (2)	ca -config c:/ssl/int/openssl.cnf -extensions server_cert -days 364 -notext -md sha256 -in c:/ssl/int/csr/csrAPMAN.pem -out c:/ssl/int/certs/cert.pem
Sprawdzenie certyfikatu CA	x509 -noout -text -in c:/ssl/int/certs/ cert.pem

- 7) **Zmień hasło w konfiguracji iRap.** Sprawdź czy ustawiono połączenie szyfrowane.
- 8) Zamknij program iRap a następnie podmień pliki w katalogu C:\KS\APW\IRAP na wygenerowane wcześniej: **cert.pem, key.pem i root.pem.**

Tytuł: Polityka bezpieczeństwa w module iRap	Wykonał: Kamil Kotarski	Sprawdził:	Zatwierdził:	Strona 35
--	-------------------------	------------	--------------	-----------