Generowanie certyfikatów za pomocą pakietu OpenSSL

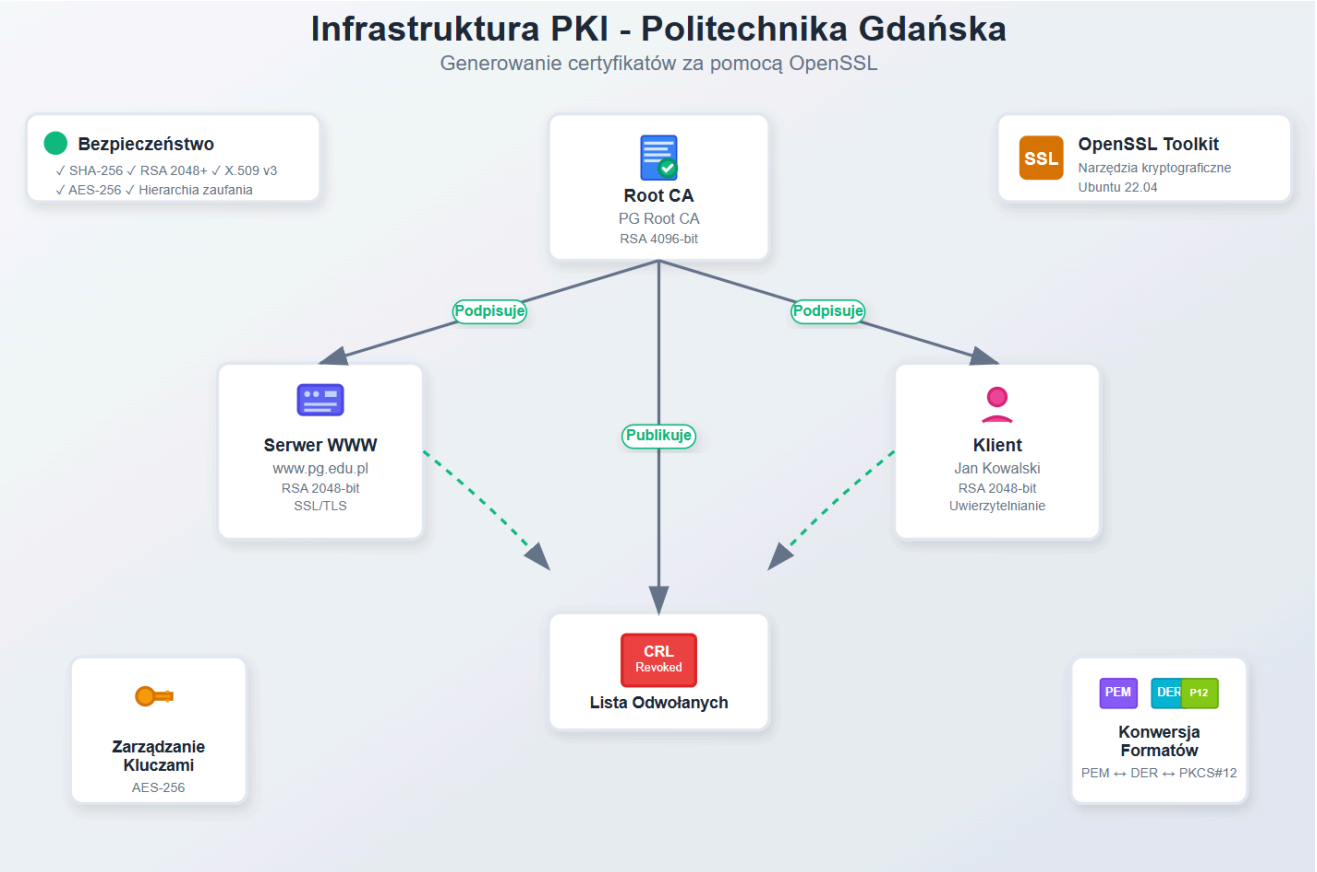
Autorzy: Krzysztof Taraszkiewicz 197796
Jakub Szymczyk
Józef Sztabiński 197890
Jakub Drejka
Przedmiot: Wprowadzenie do Cyberbezpieczeństwa

Spis treści

---

1. Wstęp

Sprawozdanie z projektu dotyczącego generowania certyfikatów za pomocą pakietu OpenSSL wykonanego w ramach przedmiotu Wprowadzenie do Cyberbezpieczeństwa. Przedstawione zostaną w nim kroki niezbędne do utworzenia pełnej infrastruktury PKI (Public Key Infrastructure), generowania certyfikatów dla różnych typów użytkowników oraz praktyczne zastosowania w środowisku serwerowym. Zaprezentowany zostanie proces tworzenia Urzędu Certyfikacji (CA), generowania certyfikatów serwera i klienta, konwersji między formatami oraz weryfikacji certyfikatów. Wszystkie operacje zostały przeprowadzone w systemie Ubuntu 22.04.



2. Infrastruktura klucza publicznego (PKI)

OpenSSL to zestaw narzędzi kryptograficznych służących do zarządzania certyfikatami X.509, kluczami RSA oraz operacjami szyfrowania. Jest to fundamentalne narzędzie w dziedzinie cyberbezpieczeństwa.

Zasady funkcjonowania PKI:

- **Hierarchiczność:** Certyfikaty są organizowane w hierarchicznej strukturze z CA na szczycie
- **Zaufanie:** Całe bezpieczeństwo opiera się na zaufaniu do certyfikatu CA
- **Weryfikowalność:** Każdy certyfikat może zostać zweryfikowany względem CA
- **Odwołalność:** Możliwość unieważnienia certyfikatów przed końcem ich ważności
- **Automatyzacja:** Procesy mogą być zautomatyzowane za pomocą skryptów
- **Standardowość:** Zgodność ze standardami X.509 zapewnia interoperacyjność

Struktura PKI w naszym projekcie:

Politechnika Gdańska Root CA
├── Certyfikat serwera (www.pg.edu.pl)
├── Certyfikat klienta (Jan Kowalski)
└── Lista odwołanych certyfikatów (CRL)

3. Porównanie metod zarządzania certyfikatami

3.1. OpenSSL (Własne CA)

- **Zalety:**
  - Pełna kontrola nad procesem
  - Brak kosztów
  - Dowolne konfiguracje
  - Edukacyjna wartość
- **Wady:**
  - Brak automatycznego zaufania przeglądarek
  - Wymagana ręczna dystrybucja CA
  - Odpowiedzialność za bezpieczeństwo

3.2. Komercyjne CA (DigiCert, GlobalSign)

- **Zalety:**
  - Automatyczne zaufanie przeglądarek
  - Wsparcie techniczne
  - Różne typy walidacji (DV, OV, EV)
  - Długie okresy ważności
- **Wady:**
  - Wysokie koszty
  - Ograniczone możliwości konfiguracji
  - Zależność od zewnętrznego dostawcy

3.3. Let's Encrypt

- **Zalety:**
  - Darmowe certyfikaty
  - Automatyczne odnawianie
  - Zaufanie przeglądarek
  - Protokół ACME
- **Wady:**
  - Krótki okres ważności (90 dni)
  - Tylko certyfikaty DV
  - Limity częstotliwości

3.4. Zarządzanie wewnętrzne

- **Zalety:**
  - Kontrola nad politykami
  - Integracja z systemami IT
  - Możliwość dostosowania

- **Wady:**
    - Wymagana ekspertyza
    - Koszty infrastruktury
    - Odpowiedzialność za bezpieczeństwo

4. Formaty przechowywania certyfikatów

Przy zarządzaniu certyfikatami wymagana jest znajomość różnych formatów przechowywania. Obecnie wspierane są główne formaty:

- **PEM** (Privacy-Enhanced Mail)
- **DER** (Distinguished Encoding Rules)
- **PKCS#12** (Personal Information Exchange)

Format	Kodowanie	Zawartość	Zastosowanie
PEM	Base64	Pojedynczy obiekt	Serwery Unix/Linux
DER	Binarny	Pojedynczy obiekt	Systemy Windows, Java
PKCS#12	Binarny	Klucz + certyfikaty	Eksport/import

4.1. Format PEM

- Najczęściej używany w środowiskach Unix/Linux
- Czytelny format tekstowy z nagłówkami BEGIN/END
- Jeden plik może zawierać wiele certyfikatów (łańcuch)
- Łatwy w edycji i przeglądaniu

4.2. Format DER

- Binarny format zgodny z ASN.1
- Kompaktowy, ale nieczytelny dla człowieka
- Używany w systemach Windows i aplikacjach Java
- Jeden plik = jeden obiekt

4.3. Format PKCS#12

- Kontener na klucze prywatne i certyfikaty
- Chroniony hasłem
- Używany do bezpiecznego transportu kluczy
- Wspierany przez większość przeglądarek

5. Metody weryfikacji certyfikatów

5.1. Weryfikacja łańcucha zaufania

openssl verify -CAfile ca-cert.pem server-cert.pem

5.2. Sprawdzanie dat ważności

- Certyfikaty mają określony okres ważności
- Sprawdzanie przed wygaśnięciem
- Automatyczne monitorowanie

5.3. Weryfikacja podpisów cyfrowych

- Sprawdzanie integralności certyfikatu
- Potwierdzenie autentyczności wystawcy
- Wykrywanie modyfikacji

5.4. Listy odwołanych certyfikatów (CRL)

- Mechanizm unieważniania certyfikatów
- Publikacja przez CA
- Sprawdzanie przed akceptacją

6. Środowiska testowe i produkcyjne

Z uwagi na bezpieczeństwo i potrzeby rozwojowe, zaleca się rozdzielenie środowisk testowych od produkcyjnych.

6.1. Środowisko testowe

- Używane do testowania konfiguracji
- Certyfikaty nie są zaufane przez przeglądarki
- Możliwość eksperymentowania bez ryzyka
- Szybkie generowanie certyfikatów

6.2. Środowisko produkcyjne

- Certyfikaty używane przez końcowych użytkowników
- Wymagana ostrożność w zarządzaniu
- Backup kluczy prywatnych
- Monitoring i alerting

7. Implementacja praktyczna

7.1. Przygotowanie środowiska

Projekt został zrealizowany z wykorzystaniem pakietu OpenSSL. Utworzono pełną strukturę PKI obejmującą urzędy certyfikacji, serwer i klienta. Architektura wygląda następująco:

```
Struktura PKI
│
├── Root CA (Politechnika Gdańska)
│    ├── Klucz prywatny (RSA 4096-bit, szyfrowany)
│    ├── Certyfikat samopodpisany
│    └── Baza certyfikatów (index.txt, serial, crlnumber)
│
├── Serwer WWW (www.pg.edu.pl)
│    ├── Klucz prywatny (RSA 2048-bit)
│    ├── CSR (Certificate Signing Request)
│    └── Certyfikat podpisany przez CA
│
└── Klient (Jan Kowalski)
      ├── Klucz prywatny (RSA 2048-bit)
      ├── CSR
      └── Certyfikat podpisany przez CA
```

7.2. Konfiguracja OpenSSL

Struktura katalogów:

```
mkdir -p
openssl-project/{ca/{private,certs,newcerts,crl},server,client,conversions}
chmod 700 openssl-project/ca/private
touch openssl-project/ca/index.txt
echo 1000 > openssl-project/ca/serial
echo 1000 > openssl-project/ca/crlnumber
```

Skonfigurowano dwa pliki `.cnf`: `openssl_server.cnf` i `openssl_client.cnf`, dopasowane odpowiednio do roli serwera i klienta. Powinne one byc skopiowane katalogu `/ca`

7.3. Generowanie certyfikatu CA

Generowanie klucza prywatnego CA:

```
openssl genrsa -aes256 -passout pass:2137 -out ca/private/ca-key.pem 4096
```

Tworzenie certyfikatu CA:

```
openssl req -config ca/openssl_client.cnf \
      -key ca/private/ca-key.pem \
      -new -x509 -days 7300 -sha256 \
      -extensions v3_ca \
      -out ca/certs/ca-cert.pem \
      -passin pass:2137
```

7.4. Generowanie i podpisywanie certyfikatów

Serwer:

```
openssl genrsa -out server/server-key.pem 2048
```

```
openssl req -config ca/openssl_server.cnf -key server/server-key.pem -new -sha256
-out server/server-csr.pem
```

```
openssl ca -config ca/openssl_server.cnf -extensions server_cert -days 375 \
-notext -md sha256 -in server/server-csr.pem -out server/server-cert.pem \
-passin pass:2137 -batch
```

Klient:

```
openssl genrsa -out client/client-key.pem 2048
```

```
openssl req -config ca/openssl_client.cnf -key client/client-key.pem -new -sha256
-out client/client-csr.pem
```

```
openssl ca -config ca/openssl_client.cnf -extensions client_cert -days 375 \
-notext -md sha256 -in client/client-csr.pem -out client/client-cert.pem \ -passin
pass:2137 -batch
```

Łańcuchy certyfikatów:

```
cat server/server-cert.pem ca/certs/ca-cert.pem > server/server-chain.pem
```

```
cat client/client-cert.pem ca/certs/ca-cert.pem > client/client-chain.pem
```

7.5. Konwersje formatów

Wykorzystując `format_conversion.sh`, wykonano:

- PEM → DER:

```
openssl x509 -outform der -in ca/certs/ca-cert.pem -out conversions/ca-cert.der
```

- DER → PEM:

```
openssl x509 -inform der -in conversions/ca-cert.der -out
conversions/ca-cert-from-der.pem
```

- PEM → PKCS#12:

```
openssl pkcs12 -export -out conversions/server.p12 -inkey server/server-key.pem \
-in server/server-cert.pem -certfile ca/certs/ca-cert.pem \
-name "PG Server Certificate" -passin pass:2137 -passout pass:2137
```

- PEM ↔ DER (klucz RSA):

```
openssl rsa -in server/server-key.pem -outform der -out
conversions/server-key.der

openssl rsa -inform der -in conversions/server-key.der -out
conversions/server-key-from-der.pem
```

7.6. Weryfikacja certyfikatów

W skrypcie `cert_verification.sh`:

Podstawowa weryfikacja:

```
openssl verify -CAfile ca/certs/ca-cert.pem server/server-cert.pem

openssl verify -CAfile ca/certs/ca-cert.pem client/client-cert.pem
```

Weryfikacja łańcucha:

```
openssl verify -CAfile ca/certs/ca-cert.pem -untrusted
server/server-cert.pem server/server-chain.pem
```

Porównanie kluczy publicznych:

```
openssl x509 -in server/server-cert.pem -noout -pubkey > /tmp/pub-from-cert.pem
openssl rsa -in server/server-key.pem -pubout > /tmp/pub-from-key.pem
diff /tmp/pub-from-cert.pem /tmp/pub-from-key.pem
```

Lista CRL:

```
openssl ca -config ca/openssl_server.cnf -gencrl -out ca/crl/ca-crl.pem
-passin pass:2137

openssl crl -in ca/crl/ca-crl.pem -noout -text
```

Odwołanie certyfikatu:

```
openssl ca -config ca/openssl_server.cnf -revoke server/server-cert.pem
-passin pass:2137

openssl verify -CAfile ca/certs/ca-cert.pem -CRLfile ca/crl/ca-crl.pem
server/server-cert.pem
```

7.7. Przykładowe wykorzystanie certyfikatu

1. Podpisywanie i szyfrowanie pliku (cert_use_example_nr1.sh)

- **Podpisywanie:**

```
openssl dgst -sha256 -sign client/client-key.pem -out document.sig
document.txt
```

- **Weryfikacja podpisu:**

  openssl dgst -sha256 -verify <(openssl x509 -in client/client-cert.pem -pubkey -noout) -signature document.sig document.txt

- **Szyfrowanie dla serwera:**

  ```
  openssl rsautl -encrypt -pubin -inkey <(openssl x509 -in
  server/server-cert.pem -pubkey -noout) -in document.txt -out
  document.encrypted
  ```

- **Odszyfrowanie:**

  ```
  openssl rsautl -decrypt -inkey server/server-key.pem -in
  document.encrypted -out document.decrypted
  ```

2. Test połączenia SSL (cert_use_example_nr2.sh)

```
openssl s_client -connect www.pg.edu.pl:443 -CAfile ca/certs/ca-cert.pem

openssl s_client -connect www.pg.edu.pl:443 -CAfile ca/certs/ca-cert.pem \
-cert client/client-cert.pem -key client/client-key.pem
```

- **Wyświetlenie szczegółów certyfikatu SSL serwera:**

```
echo | openssl s_client -connect www.pg.edu.pl:443 2>/dev/null | openssl
x509 -noout -text
```

8. Podsumowanie

8.1 Demonstracja generowania certyfikatu CA

- Utworzenie struktury PKI z hierarchią zaufania
- Generowanie klucza prywatnego CA (RSA 4096-bit z szyfrowaniem AES-256)
- Tworzenie samopodpisanego certyfikatu CA z rozszerzeniami v3

8.2 Demonstracja generowania i podpisywania certyfikatów

- Certyfikat serwera dla domeny www.pg.edu.pl
- Certyfikat klienta dla użytkownika końcowego
- Proces CSR (Certificate Signing Request)
- Podpisywanie przez CA z odpowiednimi rozszerzeniami

8.3 Demonstracja konwersji między formatami

- Konwersje PEM ↔ DER dla certyfikatów i kluczy
- Tworzenie formatów PKCS#12 z hasłem
- Ekstraktowanie komponentów z kontenerów PKCS#12
- Szyfrowanie i deszyfrowanie kluczy prywatnych

8.4 Demonstracja weryfikacji certyfikatu

- Weryfikacja łańcucha zaufania względem CA
- Sprawdzanie okresów ważności certyfikatów
- Weryfikacja podpisów cyfrowych i integralności
- Obsługa list odwołanych certyfikatów (CRL)

8.5 Demonstracja przykładu wykorzystania certyfikatu

- Konfiguracja serwerów HTTPS (nginx, Apache)
- Uwierzytelnianie wzajemne SSL/TLS
- Podpisywanie i weryfikacja dokumentów
- Szyfrowanie plików kluczem publicznym

8.6 Zastosowane technologie:

- **OpenSSL 1.1.1+** - podstawowe narzędzie kryptograficzne
- **RSA 2048/4096-bit** - algorytm klucza asymetrycznego
- **SHA-256** - bezpieczna funkcja skrótu
- **AES-256** - szyfrowanie symetryczne dla ochrony kluczy
- **X.509 v3** - standard certyfikatów cyfrowych

8.7 Aspekty bezpieczeństwa:

Projekt uwzględnia najlepsze praktyki bezpieczeństwa:

- Klucze prywatne CA chronione hasłem i odpowiednimi uprawnieniami (700)
- Użycie silnych algorytmów kryptograficznych odpornych na obecnie znane ataki
- Rozszerzenia certyfikatów zgodne z RFC 5280 zapewniające prawidłowe zastosowanie
- Odpowiednia długość kluczy rekomendowana dla roku 2025
- Hierarchiczna struktura zaufania minimalizująca ryzyko kompromitacji

8.8 Wartość edukacyjna:

Projekt demonstruje kompleksowy proces zarządzania certyfikatami cyfrowymi w środowisku PKI, od teoretycznych podstaw po praktyczne implementacje. Przedstawione rozwiązania mogą być wykorzystane w rzeczywistych scenariuszach wdrożeniowych, zachowując przy tym edukacyjny charakter przez szczegółowe wyjaśnienie każdego kroku.

Infrastruktura PKI oparta na OpenSSL stanowi fundament bezpieczeństwa w środowiskach korporacyjnych i jest niezbędnym elementem wiedzy każdego specjalisty ds. cyberbezpieczeństwa.