Podpisanie i weryfikacja kodu aplikacji w systemie Windows

Martyna Borkowska, Karolina Glaza, Amila Amarasekara

Maj 2025

Wstęp teorytyczny

Podpis cyfrowy aplikacji

sposób zapewnienia, że:

- kod pochodzi od zaufanego wydawcy (autentyczność)
- nie został zmodyfikowany po podpisaniu (integralność)
- może być zaufany przez system operacyjny

Systemy operacyjne coraz częściej wymagają, by aplikacje były podpisane – szczególnie przy dystrybucji przez internet. Nieużycie podpisu może skutkować:

- ostrzeżeniami bezpieczeństwa ("Nieznany wydawca")
- blokadą instalacji przez system

Działanie

1. Wydawca aplikacji generuje parę kluczy: prywatny i publiczny.
2. Certyfikat X.509 (np. od CA jak DigiCert, Sectigo) zawiera klucz publiczny i dane właściciela.
3. Kod jest podpisywany kluczem prywatnym – hash + podpis.
4. System używa klucza publicznego z certyfikatu do weryfikacji podpisu.

Timestamp (znacznik czasu)

To dowód, że podpis był ważny w momencie jego złożenia, nawet jeśli certyfikat wygasł później.

Wymaga połączenia z usługą TSA (Time Stamping Authority), np.:

- http://timestamp.digicert.com
- http://timestamp.sectigo.com

Krok 1: Generowanie własnego Urzędu Certyfikacji (CA)

Na potrzeby projektu utworzono własny Urząd Certyfikacji (CA) za pomocą narzędzia OpenSSL uruchamianego w PowerShellu jako administrator 2 wyniku którego wygenerowano:

- `ca.key` – klucz prywatny CA (należy chronić jak hasło),
- `ca.crt` – certyfikat CA do dystrybucji.

Użyto polecenia:

```
openssl req -x509 -newkey rsa:2048 -keyout ca.key -out ca.crt
-days 365 -addext "basicConstraints=CA:TRUE"
```

Krok 2: Dodanie CA do zaufanych certyfikatów w Windows

Aby system Windows ufał certyfikatowi aplikacji, należało dodać certyfikat CA do magazynu zaufanych głównych urzędów certyfikacji. W tym celu użyto narzędzia `certmgr.msc`, wybierając opcję „Importuj...” i wskazując plik `ca.crt`. Po zakończeniu importu, podpisy wydawane przez ten CA będą traktowane jako zaufane.

Krok 3: Generowanie certyfikatu dla aplikacji

Wygenerowano parę kluczy dla aplikacji oraz żądanie certyfikatu (CSR), następnie podpisano CSR za pomocą CA. W wyniku czego powstały pliki:

- `app.key` – klucz prywatny aplikacji,
- `app.crt` – certyfikat aplikacji podpisany przez CA,
- `ca.srl` – plik numerów seryjnych certyfikatów CA.

```
openssl req -newkey rsa:2048 -keyout app.key -out app.csr -nodes
```

```
openssl x509 -req -in app.csr -CA ca.crt -CAkey ca.key
-CAcreateserial -out app.crt -days 365 -extfile v3.ext
```

Krok 4: Podpisywanie kodu w Visual Studio

Do podpisywania kodu potrzebny był plik `.pfx`, łączący klucz prywatny i certyfikat. W Visual Studio, w ustawieniach projektu, wybrano plik `app.pfx` do podpisywania. Visual Studio oznacza aplikację jako podpisaną, ale Eksplorator Windows może nie pokazać podpisu, jeśli CA nie jest ogólnie zaufane.Generacja pliku:

```
openssl pkcs12 -export -out app.pfx -inkey app.key -in app.crt
```

Krok 5: Ręczne podpisywanie kodu z timestampem

W celu poprawnego podpisania aplikacji ręcznie użyto narzędzia `signtool.exe`, które umożliwia dołączenie znacznika czasu (timestamp). Dzięki dodaniu znacznika czasu podpis pozostanie ważny nawet po wygaśnięciu certyfikatu.

```
signtool sign /f "C:\WINDOWS\System32\app.pfx" /p silnehaslo
/fd SHA256 /tr http://timestamp.digicert.com /td SHA256
"MKA_Demo_App.exe"
```

Krok 6: Weryfikacja podpisu

Sprawdzono poprawność podpisu poprzez właściwości pliku w Eksploratorze Windows (zakładka „Podpisy cyfrowe") oraz za pomocą:

```
signtool verify /v /pa "MKA_Demo_App.exe"
```

Podsumowanie

W ramach projektu zrealizowano proces pełnego podpisywania kodu aplikacji w systemie Windows. Wykonane kroki:

- Certyfikaty:
  Wygenerowano własny Urząd Certyfikacji (CA) oraz certyfikat aplikacji z rozszerzeniami v3 i dodano CA do zaufanych w systemie Windows, aby umożliwić weryfikację podpisów.
- Podpisywanie kodu:
  Skonfigurowano podpisywanie projektu w IDE (Visual Studio) z użyciem certyfikatu .pfx i przeprowadzono ręczne podpisywanie aplikacji przy pomocy signtool.exe.
- Timestamp:
  Zastosowano usługę znakowania czasem (TSA) w celu zapewnienia ważności podpisu mimo ewentualnego wygaśnięcia certyfikatu.
- Weryfikacja:
  Sprawdzono ważność i autentyczność podpisu z poziomu eksploratora Windows oraz narzędzia signtool verify.