LUKI BEZPIECZEŃSTWA?

NIE W MOIM KODZIE ŹRÓDŁOWYM!

Adam Czubak, PhD
PGP aczubak@4crypto.eu
www.linkedin.com/in/adamczubak

Kierownik B+R projektu CyberEva CYBERSECIDENT/489912/IV/NCBR/2021 Uniwersytetu Opolskiego
Kierownik Zespołu Badawczego „Bezpieczeństwo IT" w Instytucie Informatyki Uniwersytetu Opolskiego

NSA CNSS (Committee on National Security Systems) 4011 Recognition
NSA CNSS (Committee on National Security Systems) 4013 Recognition
US Department of Defense (DoD) 8570.01-M Certification

CheckPoint CCSA
CISCO CCNP Security | Firewall Security Specialist | IPS Specialist | VPN Security Specialist | CCNA CyberOps

opole.dev
2023.10.24 Opole

Informacja o Prawach Autorskich

Kontakt z autorem:

Adam Czubak, PhD
Email: aczubak@4crypto.eu
Tel.: +48 691 122 312

Agenda

1. Ot, klasyka…
2. Moduły i biblioteki na których polegamy
3. Apple i SSL
4. Uwierzytelnienie w CyberEva

#1 Ot, klasyka…

Czarna lista grzechów

1. Niepoprawna walidacja danych wejściowych
2. Niezaszyfrowane połączenia lub niezabezpieczone API, słabe mechanizmy uwierzytelniania i autoryzacji
3. Nieaktualne biblioteki (bądź takie z lukami)
4. Niewłaściwe zarządzanie błędami, w tym yświetlanie zbyt szczegółowych informacji o błędach może dostarczyć atakującym użytecznych danych
5. Brak mechanizmów ograniczających dostęp do krytycznych zasobów systemu

Buffer overflow

```cpp
#include <cstdio>
#include <cstring>
#include <iostream>

const char *PASSWORD_FILE = "rictro";

int main()
{
  char input[8];
  char password[8];

  std::sscanf(PASSWORD_FILE, "%s", password);

  std::cout << "Enter password: ";
  std::cin >> input;

  // Debug prints:
  // std::cout << "Address of input: " << &input << "\n";
  // std::cout << "Address of password: " << &password << "\n";
  // std::cout << "Input: " << input << "\n";
  // std::cout << "Password: " << password << "\n";

  if (std::strncmp(password, input, 8) == 0)
    std::cout << "Access granted\n";
  else
    std::cout << "Access denied\n";

  return 0;
}
```

```
1  Enter password: rictro
2  Access granted


1  Enter password: hello
2  Access denied


1  Enter password: sunshinesunshine
2  Access granted
```

Snyk Security Research Team, Buffer overflow attacks in C++: A hands-on guide
https://snyk.io/blog/buffer-overflow-attacks-in-c/

Buffer overflow

input								password							
h	e	l	l	o	\0			r	i	c	t	r	o	\0	

input								password							
s	u	n	s	h	i	n	e	s	u	n	s	h	i	n	e

Snyk Security Research Team, Buffer overflow attacks in C++: A hands-on guide
https://snyk.io/blog/buffer-overflow-attacks-in-c/

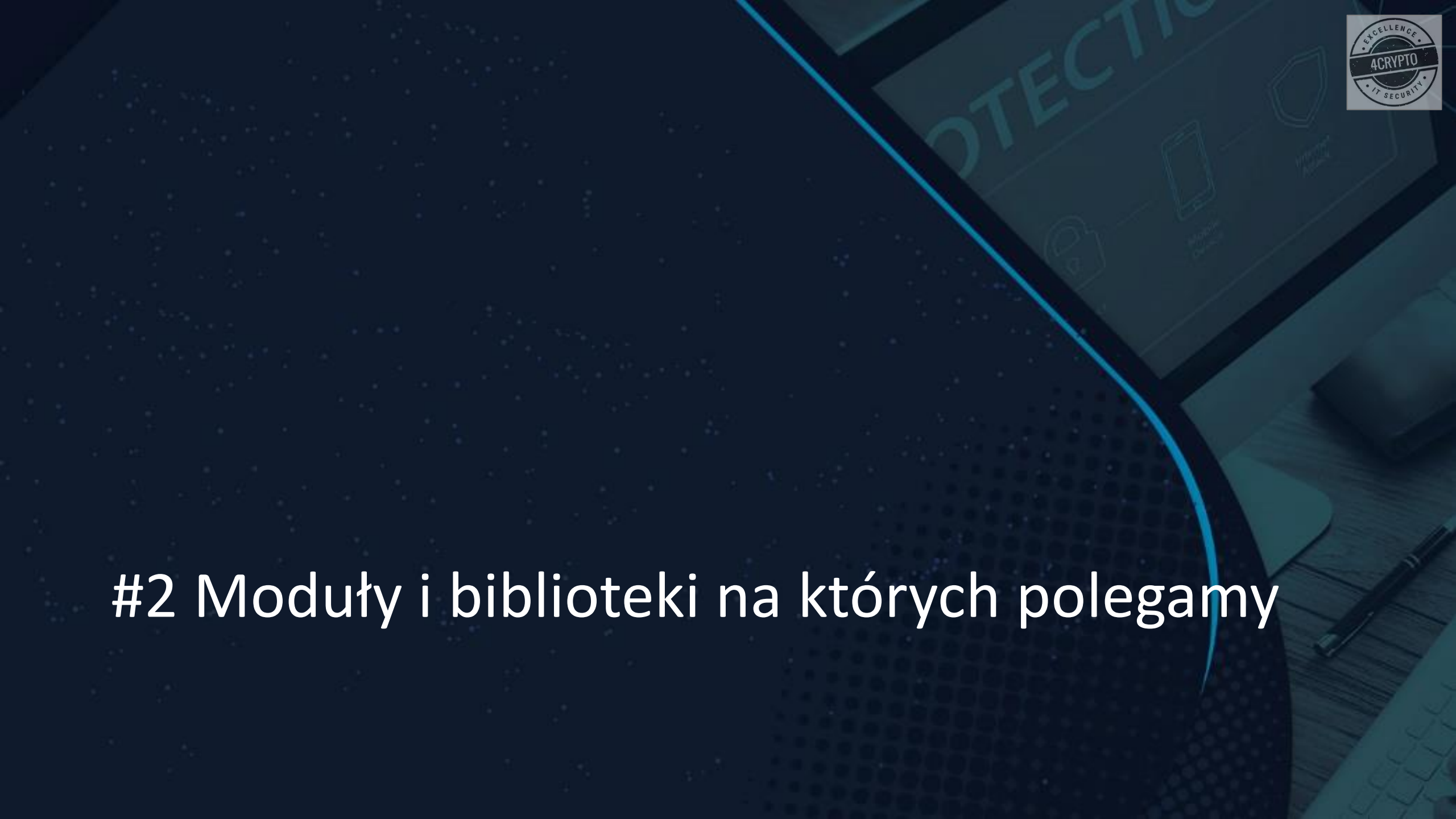
Bezpieczenie jest gdzie?

Buffer overflow jest problemem w językach programowania, które nie zarządzają pamięcią automatycznie i pozwalają na bezpośredni dostęp do adresów pamięci, takich jak C i C++. W językach, które mają wbudowane mechanizmy zarządzania pamięcią i To ryzyko nie występuje.

1. W Java tablice są obiektami, które mają swój własny rozmiar. Próba przekroczenia granic tablicy wywyoła wyjątek ArrayIndexOutOfBoundsException.
2. W Pythonie listy są dynamiczne i mogą zmieniać swój rozmiar. Nie ma również bezpośredniego dostępu do pamięci.
3. C# podobnie jak w Javie, C# ma wbudowane mechanizmy zarządzania pamięcią i kontrolę typów, co minimalizuje ryzyko.
4. JavaScript i inne skryptowe.
5. Rust jest podobny do C/C++, ale ma silne mechanizmy kontroli dostępu do pamięci, które praktycznie eliminują możliwość buffer overflow.

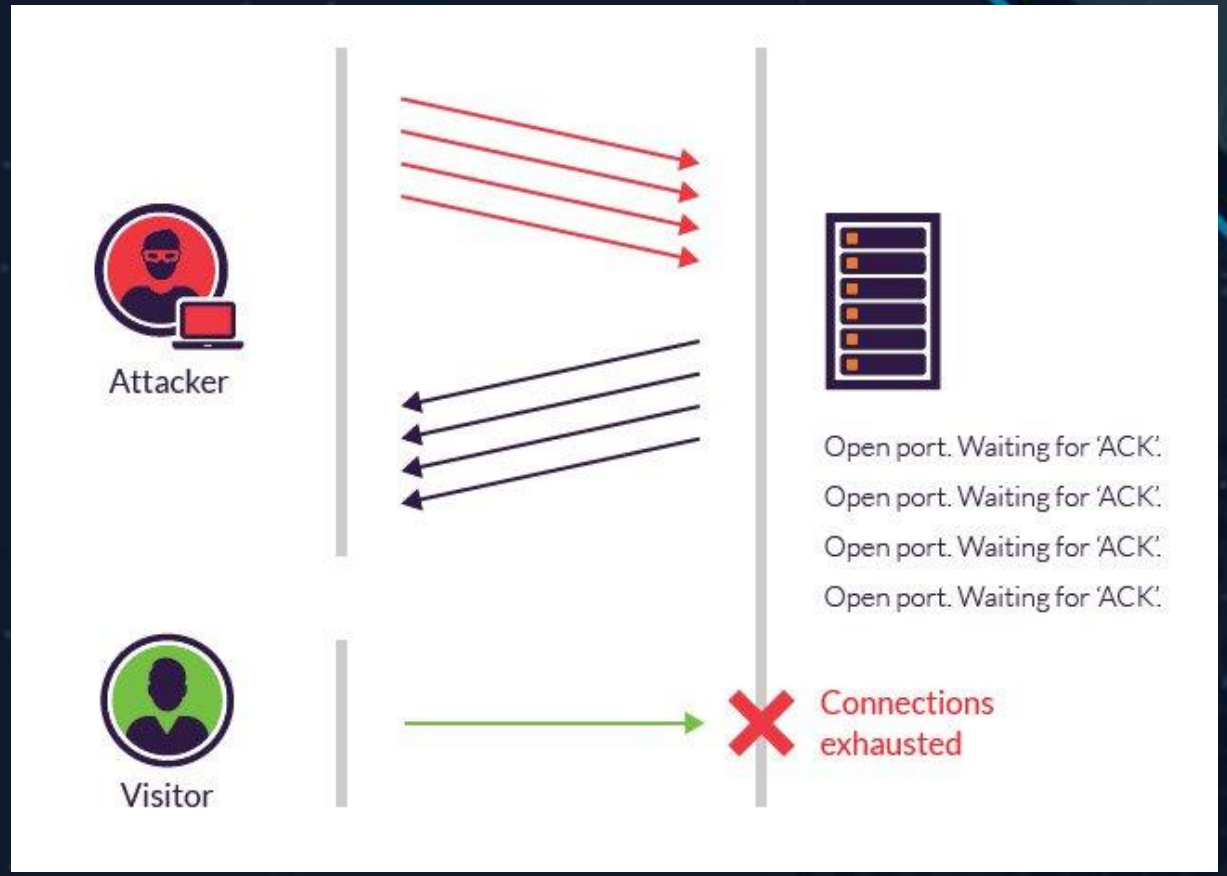
Niepoprawna walidacja danych wejściowych

1. Formularze są już bezpieczne (SQL Injection czy Cross-Site Scripting (XSS))
2. Integracja z innymi systemami stanowi wyzwanie, czyli sanityzacja danych wejścia dotyczy również danych pobieranych automatycznie z innych systemów, a nie tylko od użytkownika.



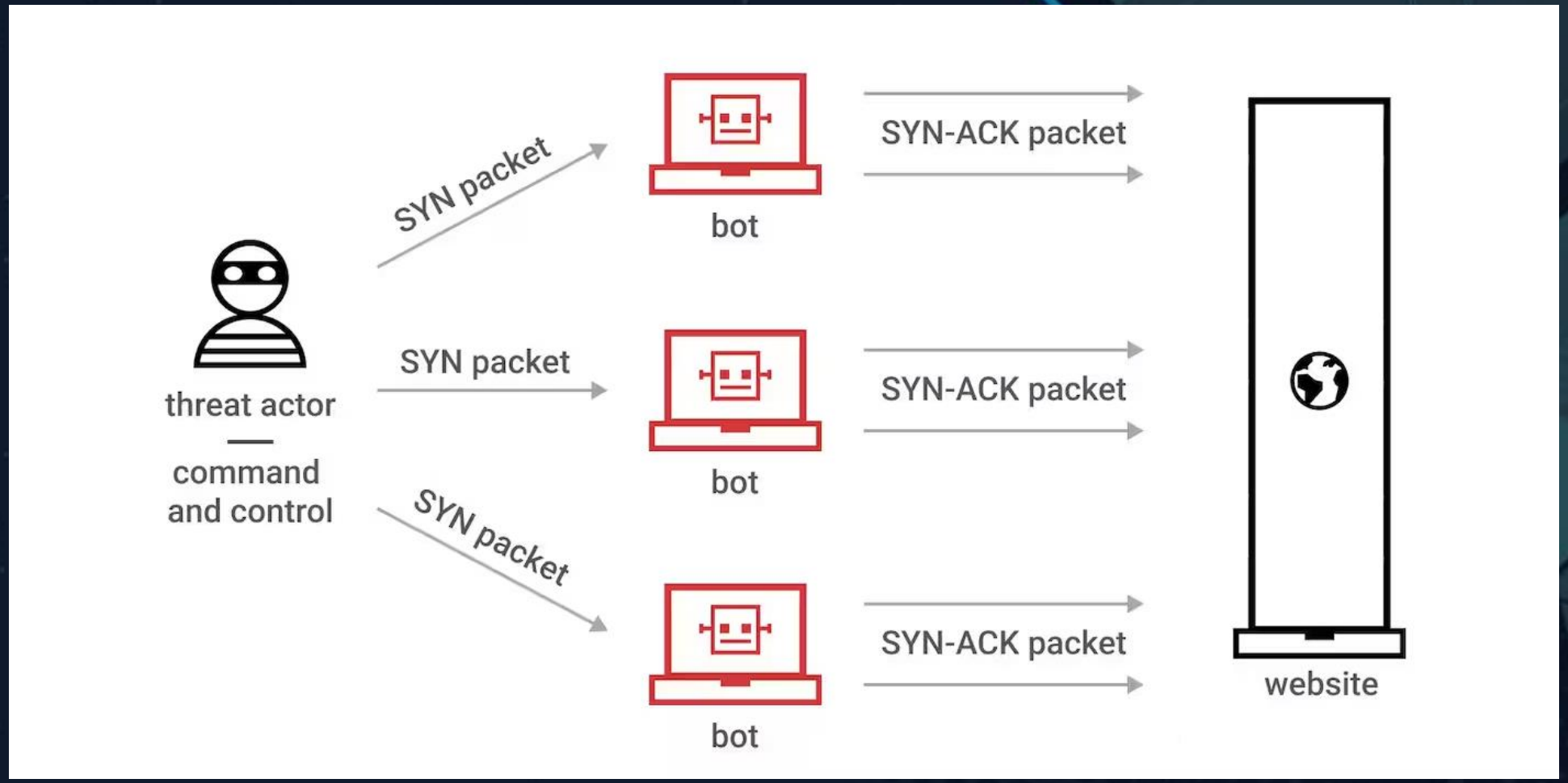
#2 Moduły i biblioteki na których polegamy

Klasyczny DoS 1.0 [SYN-Flood Attack]



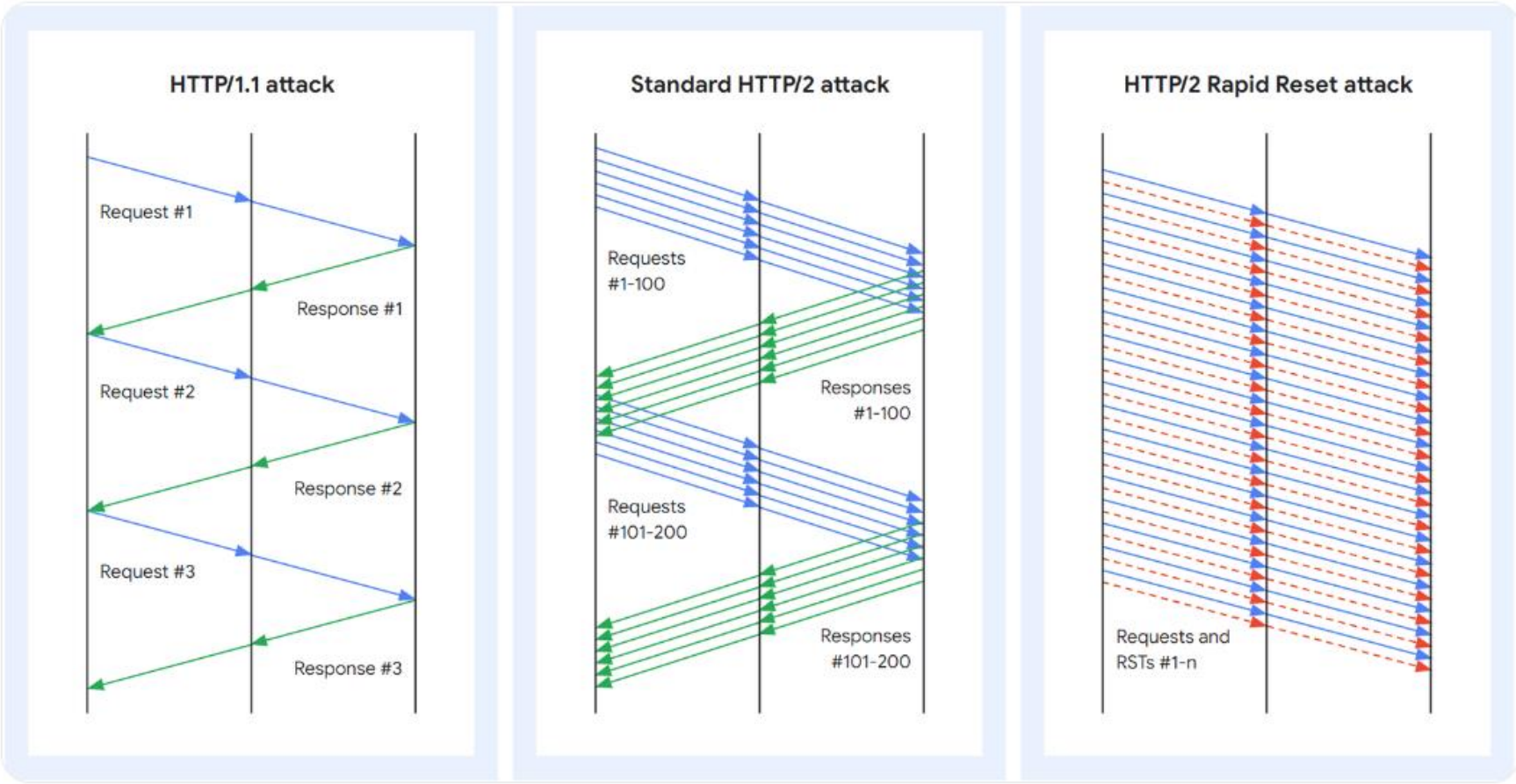
Imperva, TCP SYN Flood
https://www.imperva.com/learn/ddos/syn-flood/

DDoS 1.0 [Bot-Based SYN-Flood Attack]



Akamai, The threat of application-layer attacks
https://www.akamai.com/glossary/what-is-application-layer-ddos-attack

HTTP/2 ‘Rapid Reset’ DDoS 10.X.2023

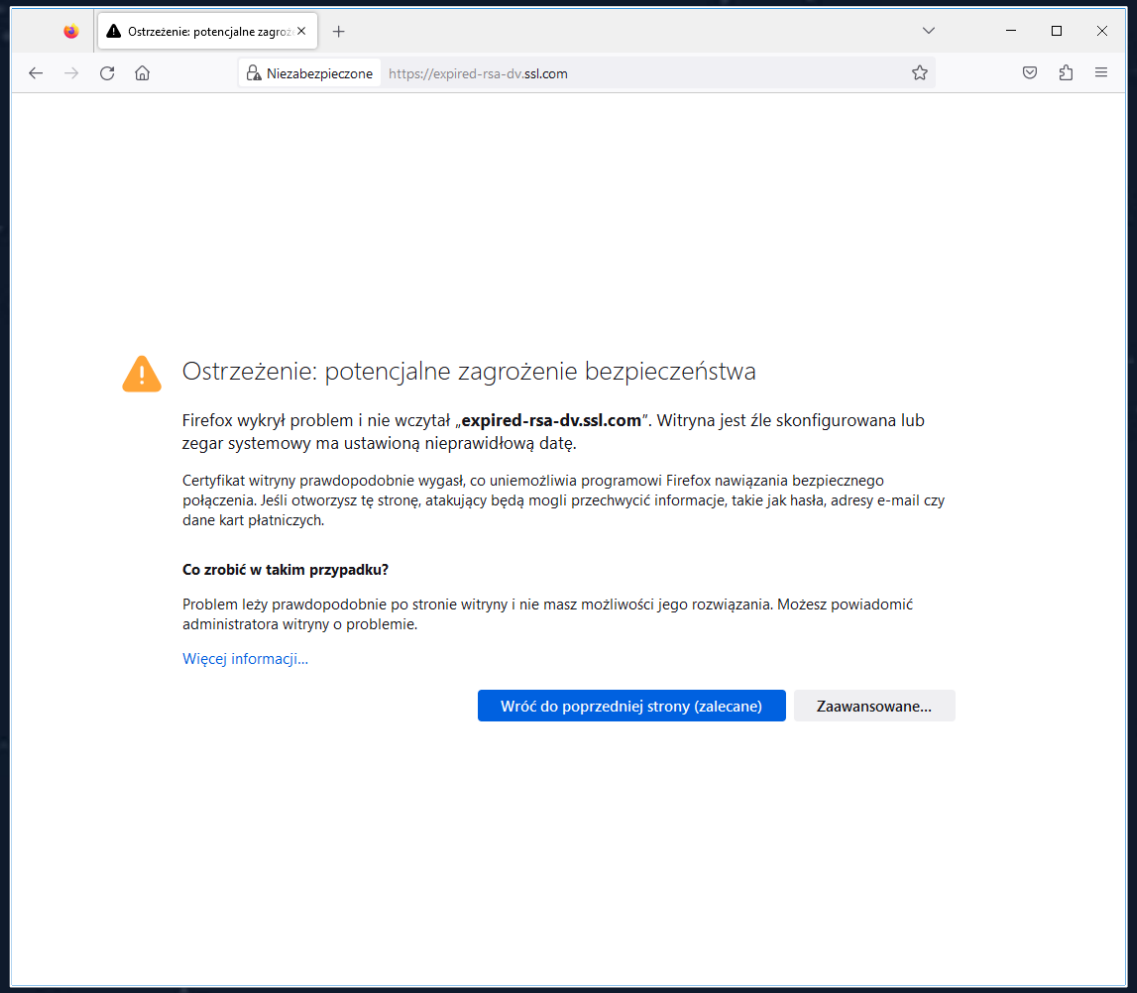


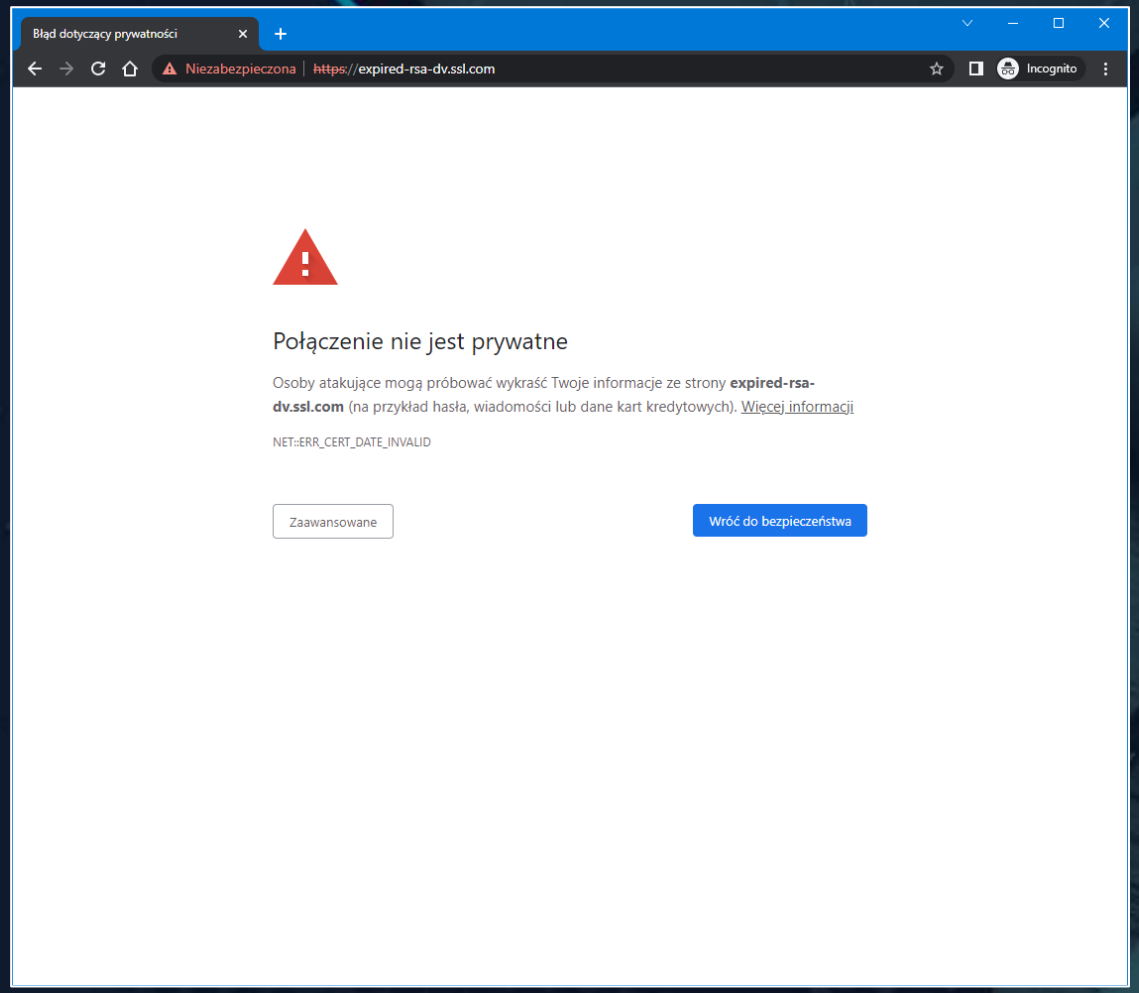
HTTP/1.1 and HTTP/2 request and response pattern

Google, How it works: The novel HTTP/2 ‘Rapid Reset’ DDoS attack
https://cloud.google.com/blog/products/identity-security/how-it-works-the-novel-http2-rapid-reset-ddos-attack

#3 Apple i SSL

Walidacja certyfikatu X.509

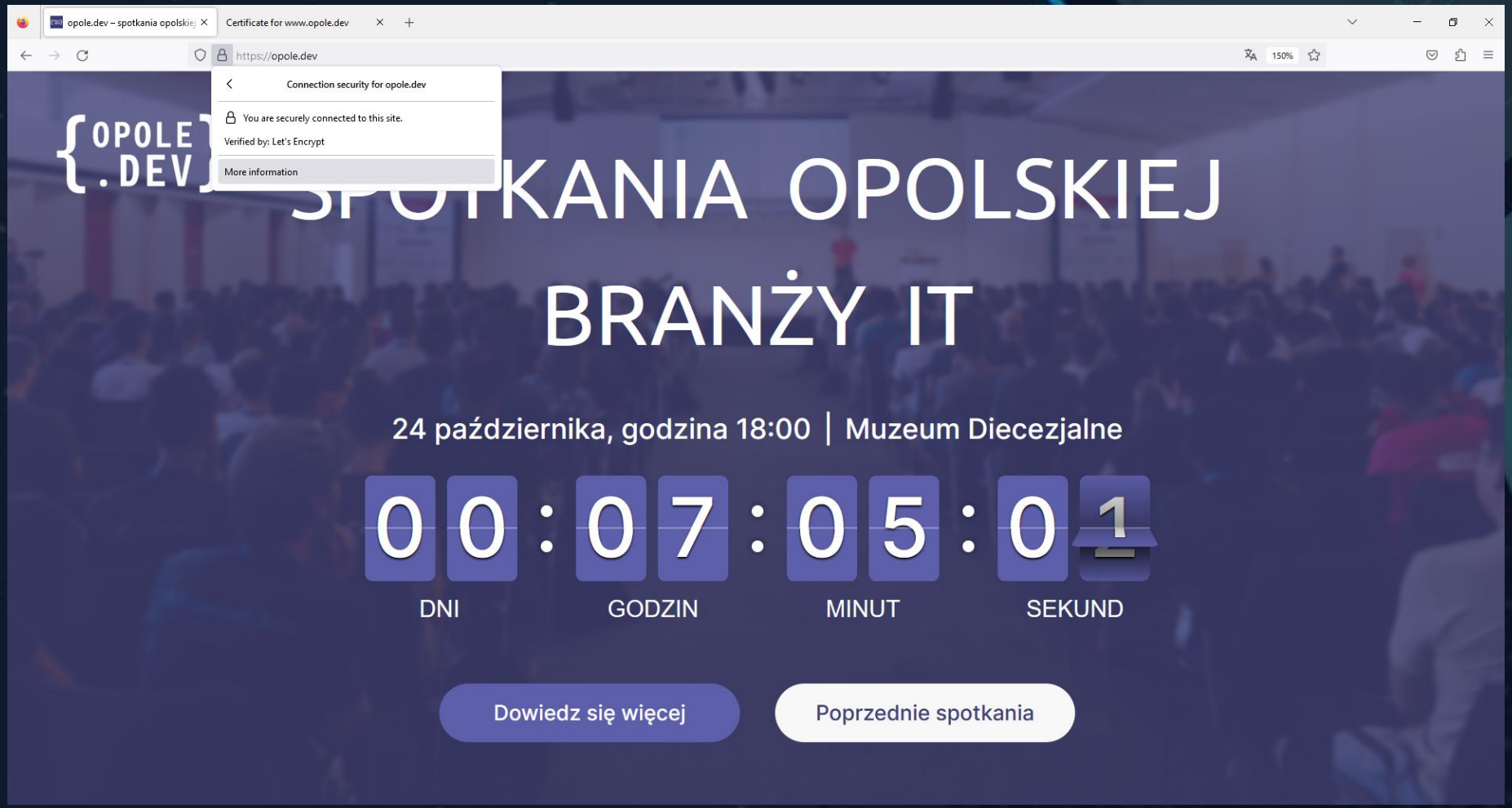




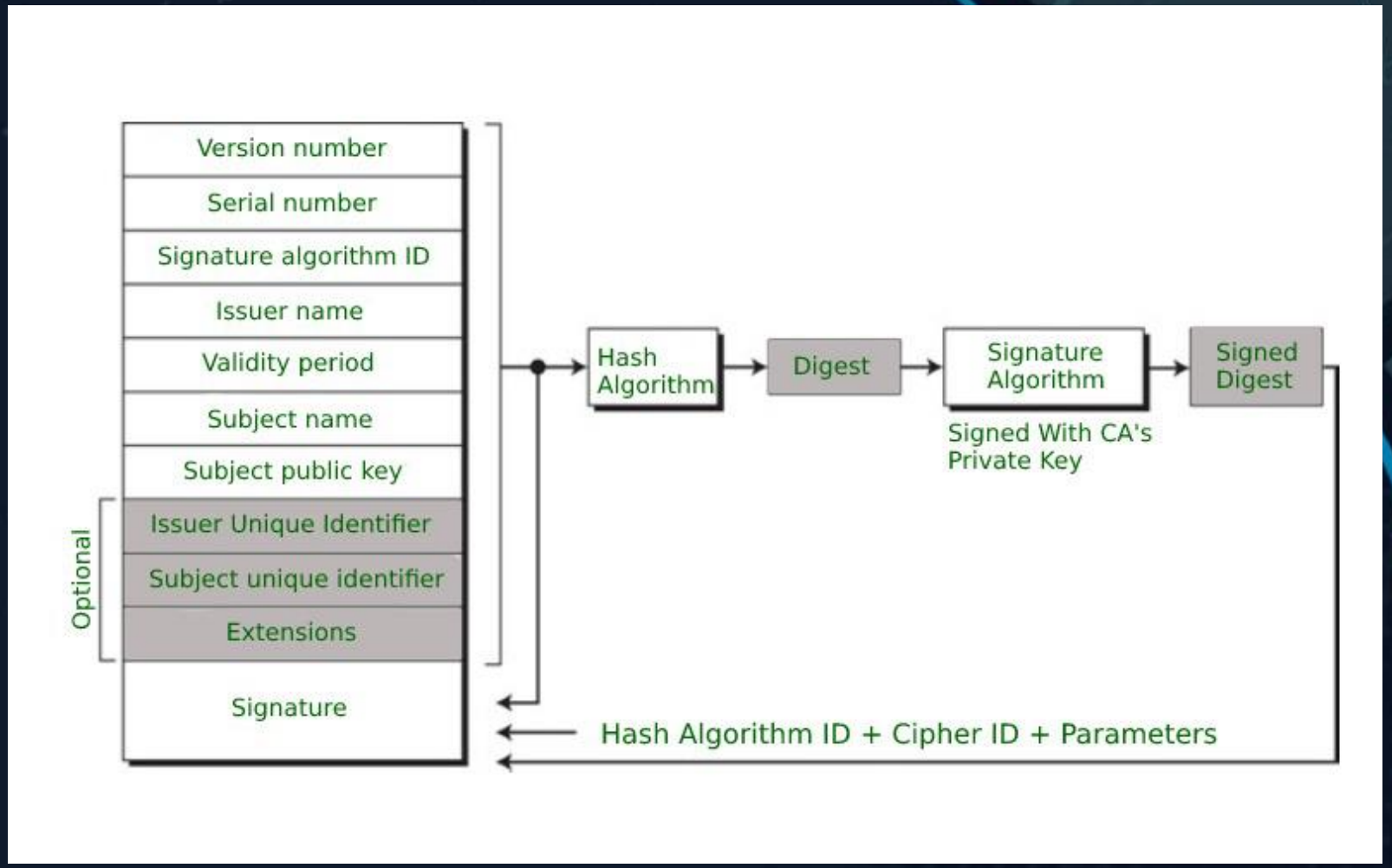
Przykłady błędnych certyfikatów SSL/TLS:
https://expired-rsa-dv.ssl.com/

Certyfikaty SSL/TLS

DEMO



Certyfikaty SSL/TLS



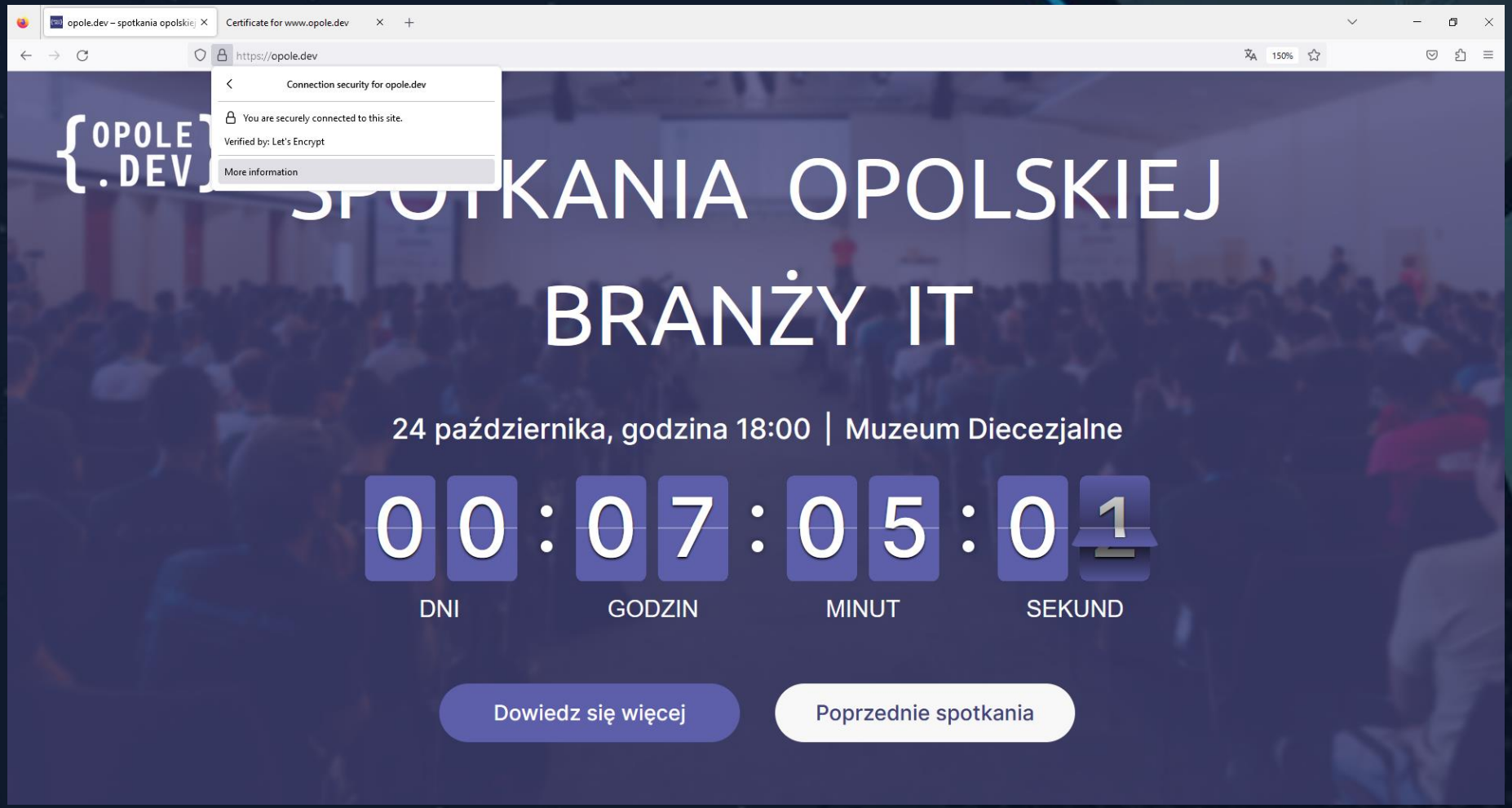
GeeksforGeeks, X.509 Authentication Service
https://www.geeksforgeeks.org/x-509-authentication-service/

Walidacja certyfkatu X.509

1. Nawiązanie połączenia: Strona połączenia nawiązuje połączenie z serwerem webowym i żąda certyfikatu SSL.
2. Pobranie certyfikatu: Serwer przesyła certyfikat SSL do strony połączenia.
3. Sprawdzenie ważności: Strona połączenia weryfikuje, czy certyfikat jest aktualny i nie wygasł.
4. Sprawdzenie wydawcy: Strona połączenia sprawdza, czy certyfikat został wydany przez zaufaną instytucję certyfikującą (CA).
5. Sprawdzenie domeny: Strona połączenia sprawdza, czy certyfikat jest wydany dla domeny, z którą użytkownik próbuje się połączyć.
6. Sprawdzenie CRL i OCSP: Opcjonalnie, strona połączenia może sprawdzić listę odwołanych certyfikatów (CRL) lub użyć protokołu OCSP, aby upewnić się, że certyfikat nie został odwołany.
7. Sprawdzenie algorytmów: Strona połączenia sprawdza, czy algorytmy szyfrujące i podpisu są silne i aktualne.
8. Sprawdzenie łańcucha certyfikatów: W przypadku certyfikatów wielopoziomowych, strona połączenia weryfikuje cały łańcuch certyfikatów aż do głównego certyfikatu korzeniowego.
9. Potwierdzenie weryfikacji: Jeżeli wszystkie kroki weryfikacji zakończą się sukcesem, strona połączenia ustanawia zaszyfrowane połączenie z serwerem.
10. Błąd weryfikacji: W przypadku niepowodzenia któregokolwiek z kroków, strona połączenia wyświetli ostrzeżenie o niebezpiecznym połączeniu.

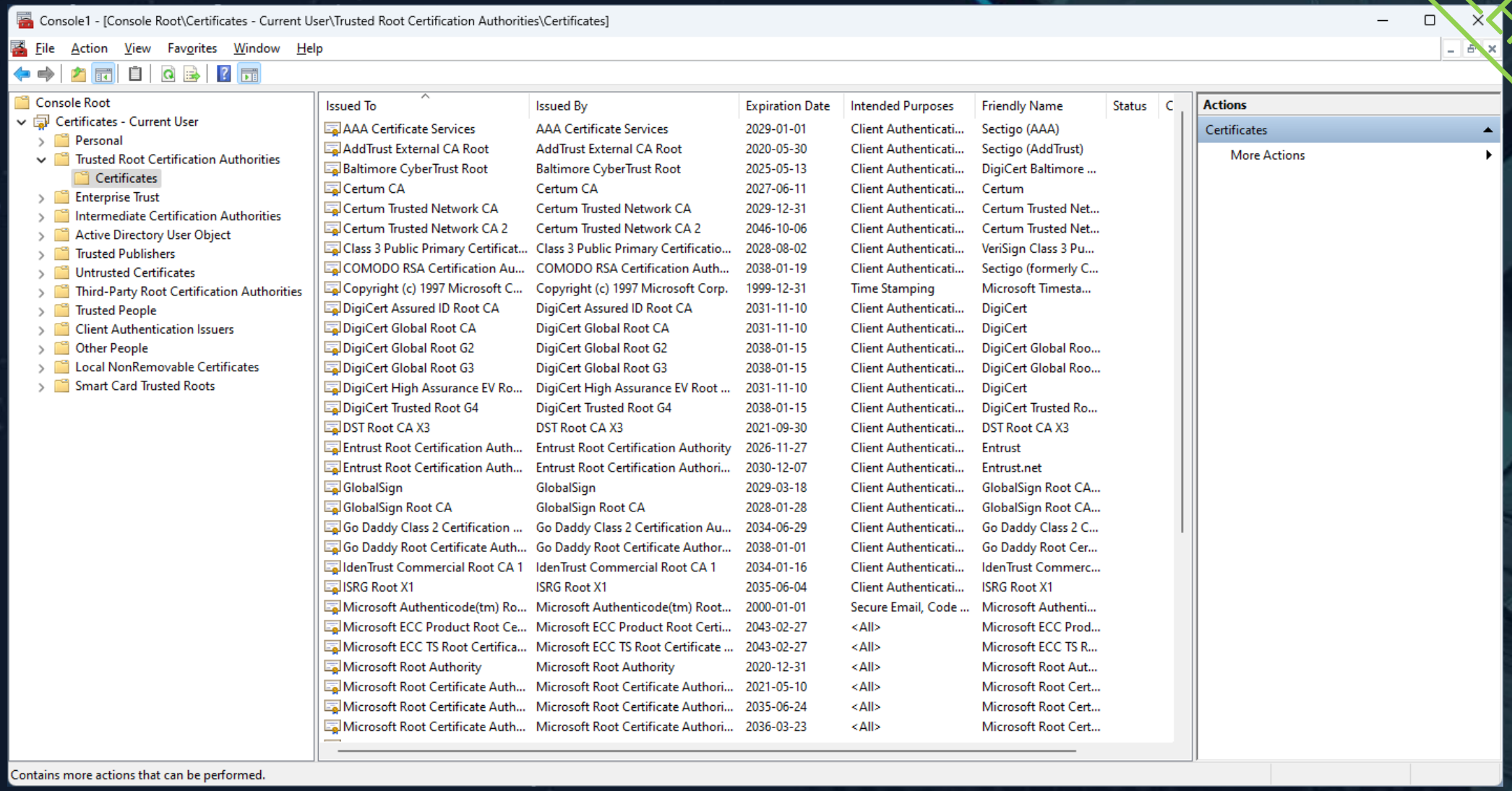
Certyfikaty SSL/TLS

DEMO



HTTPS – CA w MMC

ĆWICZENIE



Apple i SSL

Print subscriptions | Sign in | Search jobs | Search | Europe edition

Support the Guardian

Fund independent journalism with €5 per month

Support us →

The Guardian

News | Opinion | Sport | Culture | Lifestyle | More

World UK Climate crisis Environment Science Global development Football **Tech** Business Obituaries

Apple

Analysis

Apple's SSL iPhone vulnerability: how did it happen, and what next?

Charles Arthur

SSL vulnerability in iPhone, iPad and on Mac OS X appeared in September 2012 - but cause remains mysterious as former staffer calls lack of testing 'shameful'

Charles Arthur, Apple's SSL iPhone vulnerability: how did it happen, and what next? (2014)
https://www.theguardian.com/technology/2014/feb/25/apples-ssl-iphone-vulnerability-how-did-it-happen-and-what-next

Walidacja certyfikatu SSL/TLS

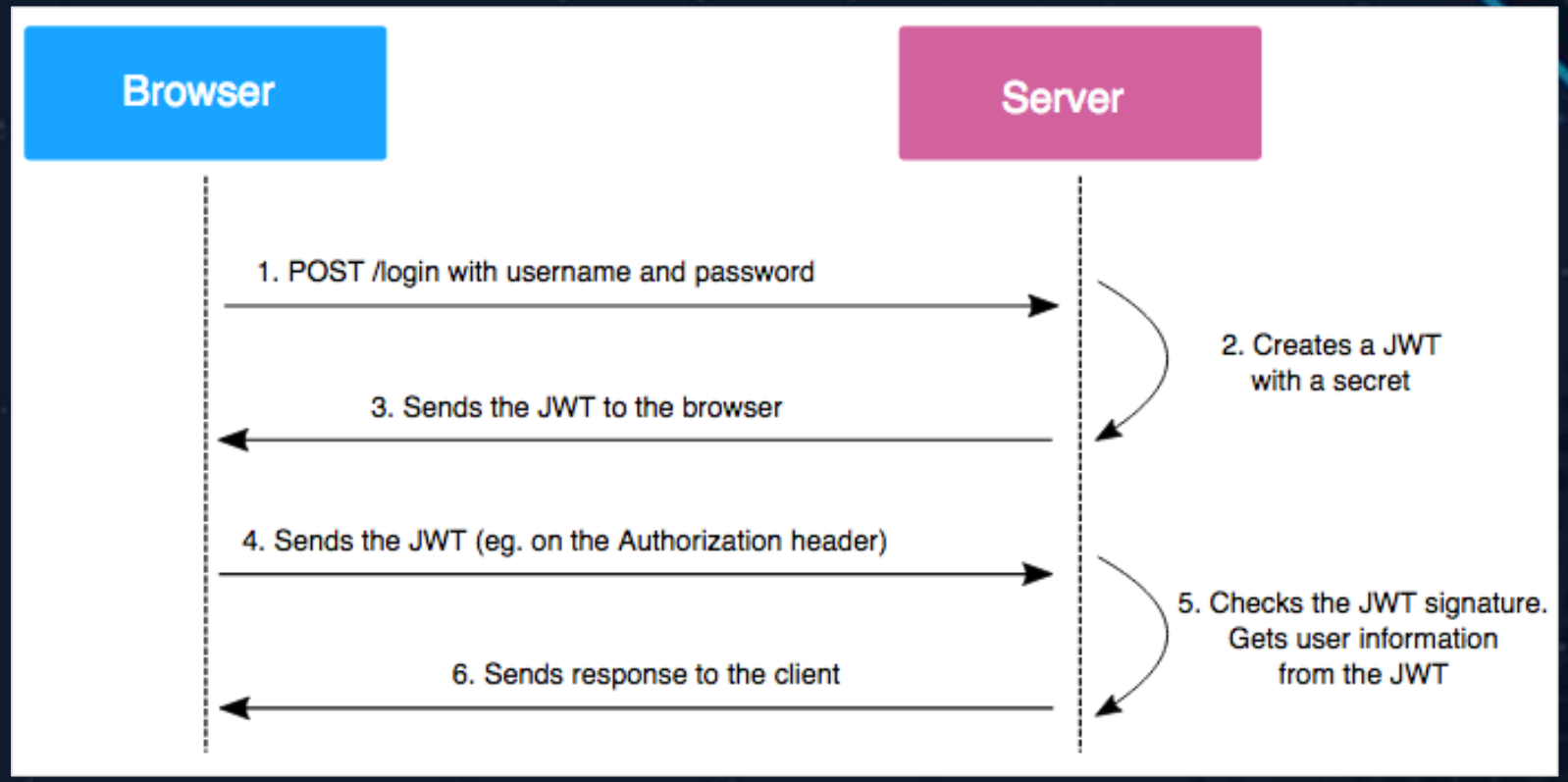
```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

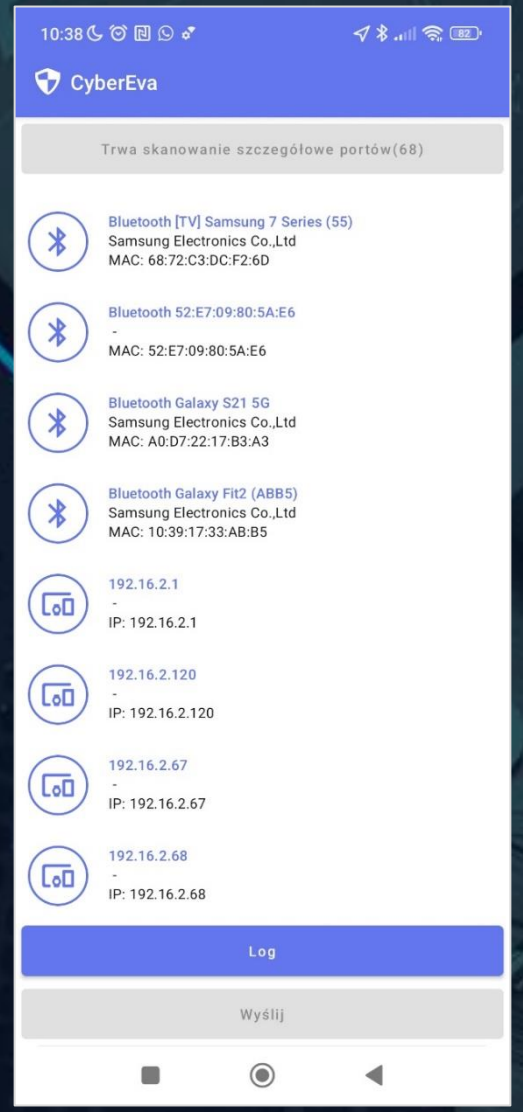
These checks got skipped

Charles Arthur, Apple's SSL iPhone vulnerability: how did it happen, and what next? (2014)
https://www.theguardian.com/technology/2014/feb/25/apples-ssl-iphone-vulnerability-how-did-it-happen-and-what-next

#4 Uwierzytelnienie w CyberEva

Case - JWT


Browser
Server
1. POST /login with username and password
2. Creates a JWT with a secret
3. Sends the JWT to the browser
4. Sends the JWT (eg. on the Authorization header)
5. Checks the JWT signature. Gets user information from the JWT
6. Sends response to the client


10:38
CyberEva
Trwa skanowanie szczegółowe portów(68)
Bluetooth [TV] Samsung 7 Series (55)
Samsung Electronics Co.,Ltd
MAC: 68:72:C3:DC:F2:6D
Bluetooth 52:E7:09:80:5A:E6
-
MAC: 52:E7:09:80:5A:E6
Bluetooth Galaxy S21 5G
Samsung Electronics Co.,Ltd
MAC: A0:D7:22:17:B3:A3
Bluetooth Galaxy Fit2 (ABB5)
Samsung Electronics Co.,Ltd
MAC: 10:39:17:33:AB:B5
192.16.2.1
-
IP: 192.16.2.1
192.16.2.120
-
IP: 192.16.2.120
192.16.2.67
-
IP: 192.16.2.67
192.16.2.68
-
IP: 192.16.2.68
Log
Wyślij

Case – JWT Java Web Tokens

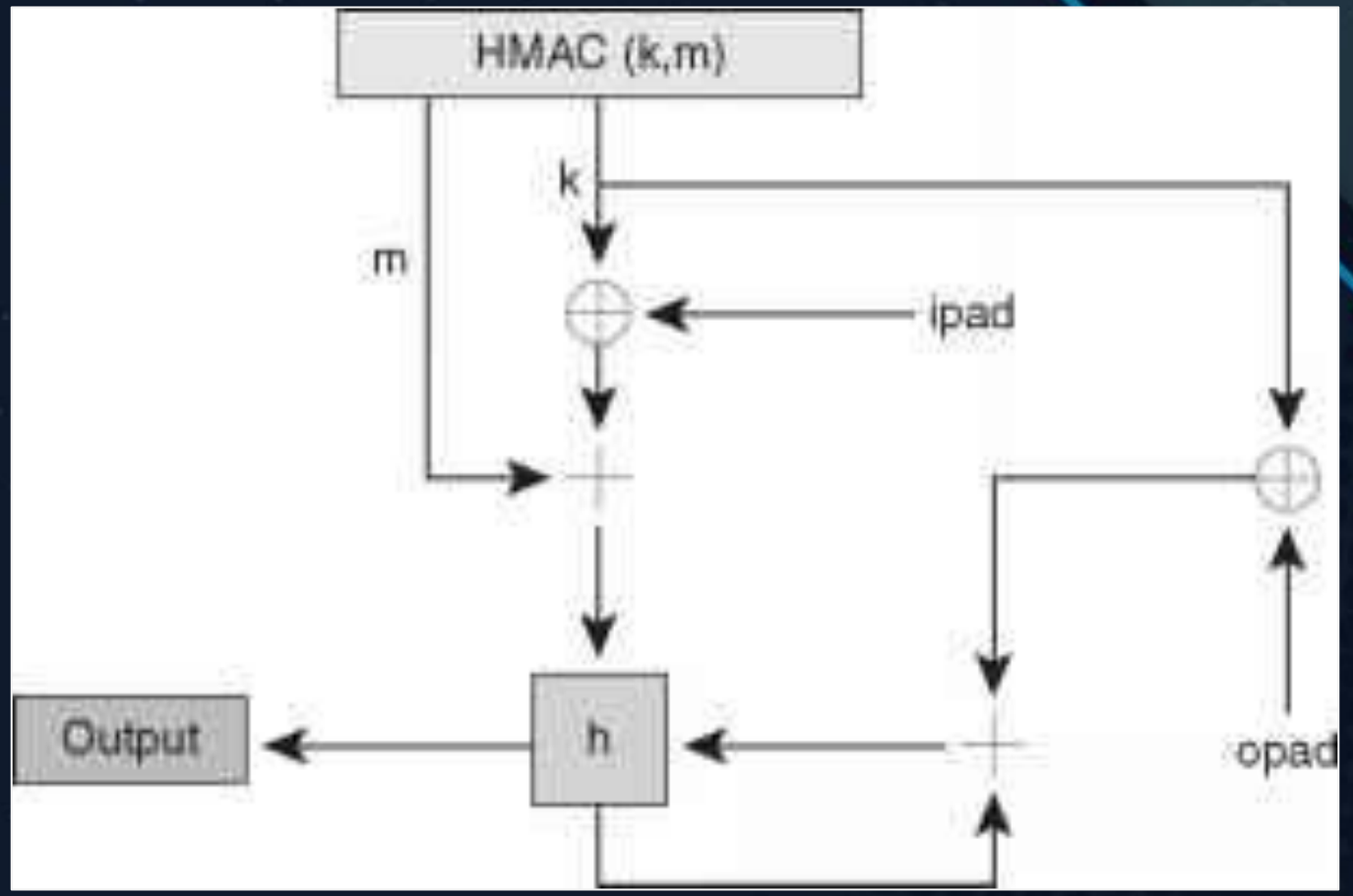
The JWT's Header

```
{
    "alg": "HS256",
    "typ": "JWT"
}
```

The JWT's Data

```
{
    "iat": 1416929109,
    "jti": "aa7f8d0a95c",
    "scopes": [
        "repo",
        "public_repo"
    ]
}
```

Case – JWT Java Web Tokens



Cisco Certified Expert, HMAC Explained
https://www.ccexpert.us/catalyst-switch/hmac-explained.html

Case – JWT Java Web Tokens – RFC7518

3.1. "alg" (Algorithm) Header Parameter Values for JWS

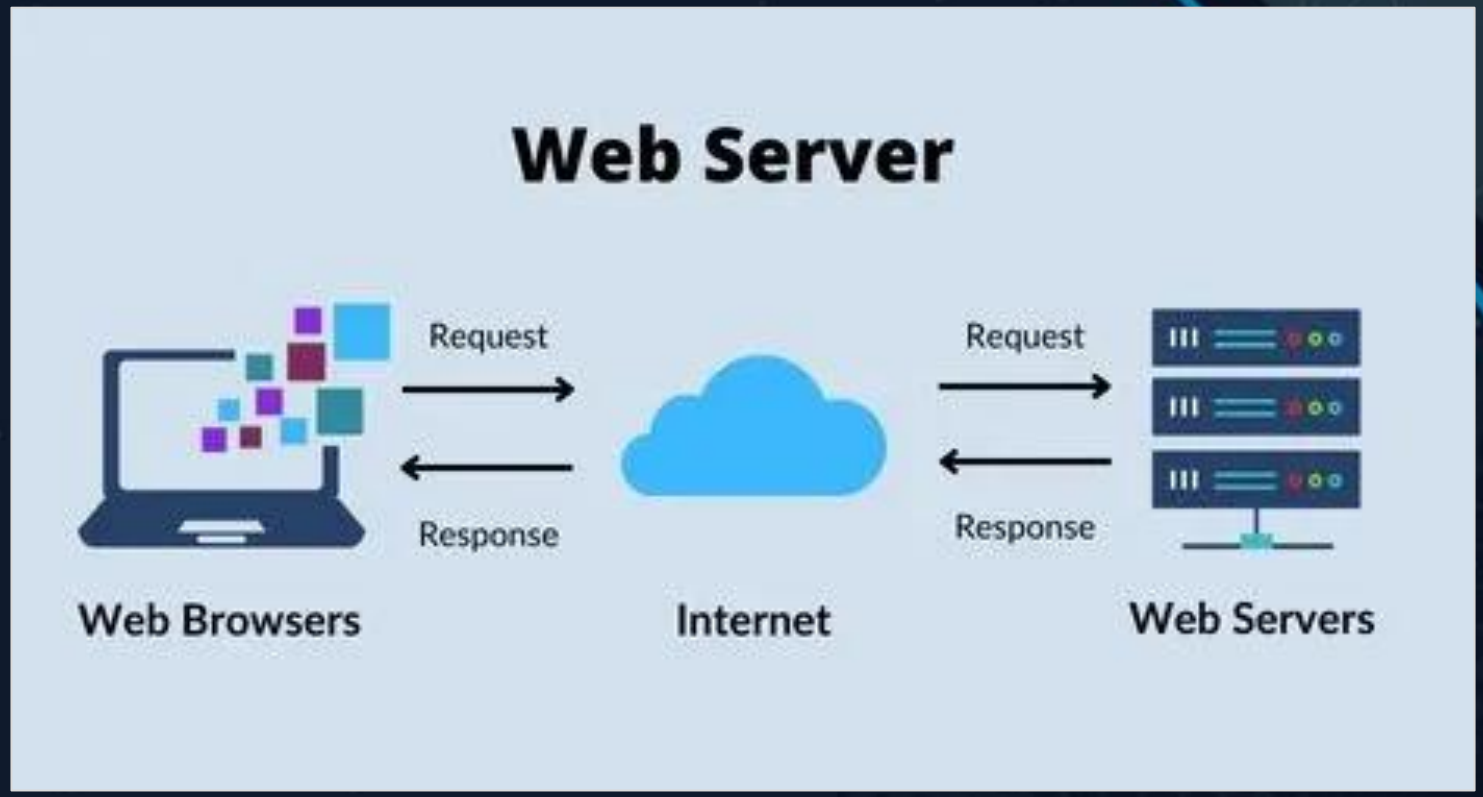
The table below is the set of "alg" (algorithm) Header Parameter values defined by this specification for use with JWS, each of which is explained in more detail in the following sections:

"alg" Param Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

The use of "+" in the Implementation Requirements column indicates that the requirement strength is likely to be increased in a future version of the specification.

RFC 7518 - JSON Web Algorithms (JWA)
https://datatracker.ietf.org/doc/html/rfc7518

Case – JWT – Credential Reuse



Źródła:

1. ENISA – Agencja Unii Europejskiej ds. Cyberbezpieczeństwa
2. CISA – Cybersecurity & Infrastructure Security Agency (U.S. Department of Homeland Security)
3. Washington Post & BBC
4. CERT Polska , NASK
5. Sophos
6. Bleeping Computer
7. Obserwacje własne

UNIWERSYTET OPOLSKI

Zapraszamy na przyszłoroczną

III edycję Konferencji CACS - Conference on Applied Cybersecurity

21-22.X.2024

Park Naukowo-Technologiczny w Opolu

https://cacs.uni.opole.pl