CORS

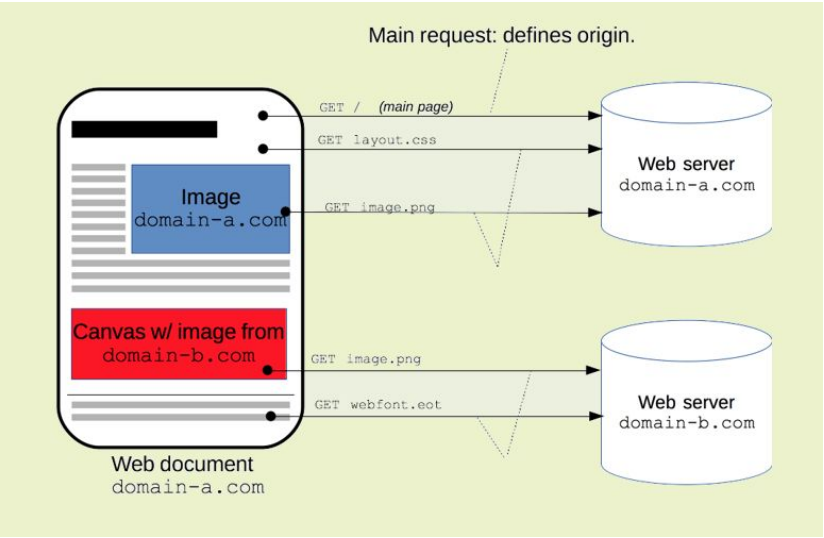
czyli zabezpieczenie przed przesyłaniem danych na wrogi serwer

Aleksandra Mardaus
Jakub Jastrzębski

Same-Origin Policy (SOP)

Zasada tego samego pochodzenia (w skrócie SOP) jest ważną koncepcją w modelu bezpieczeństwa aplikacji internetowych.

Przeglądarka internetowa zezwala skryptom zawartym na jednej stronie internetowej na dostęp do danych na drugiej stronie internetowej, tylko wtedy, gdy obie strony internetowe mają to samo pochodzenie (origin)



Origin

Właściwie to w świecie WWW jednoznacznie definiuje on pojedynczą aplikację.
Origin jest definiowany przez trójkę: **protokół**, np. HTTP czy HTTPS), **host** oraz **port**. Gdy wszystkie trzy elementy naszego *origin* są takie same, uważamy, że są one tego samego pochodzenia.

URL A	URL B	Same origin?
http://softwareskill.pl/a	http://softwareskill.pl/b	TAK
http://softwareskill.pl	https://softwareskill.pl	**NIE (inny protokół http vs https)**
http://softwareskill.pl	http://www.softwareskill.pl	**NIE (inny host)**
http://softwareskill.pl	http://softwareskill.pl:8080	**NIE (inny port)**
http://softwareskill.pl/user	http://softwareskill.pl/admin	TAK
http://softwareskill.pl	http://news.softwareskill.pl	**NIE (inny host)**

Origin

SOP powoduje, że dwie aplikacje charakteryzujące się różnymi originami (więc dwie różne aplikacje) nie mogą używać swoich wzajemnych elementów (ściąganie, osadzanie czy odpytywanie).

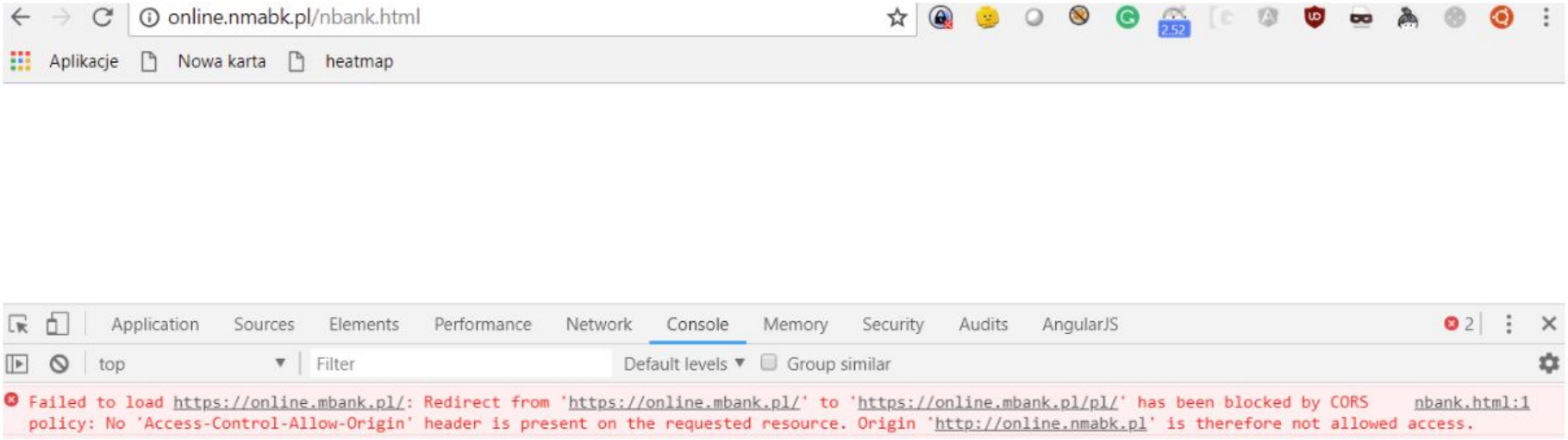
Polityka SOP dotyczy nie tylko dokumentów (DOM), ale rozszerza się także na ciasteczka (ang. Cookies) oraz cały stos kontekstu Javascript.

Żądanie niespełniające SOP

online.nbank.pl/nbank.html

```
<html>
<body>
        <script>
                var req = new XMLHttpRequest();
                req.open('POST', 'https://online.mbank.pl', true);
                req.setRequestHeader('Content-Type','application/json');
                req.send('');
        </script>
</body>
</html>
```

Żądanie niespełniające SOP

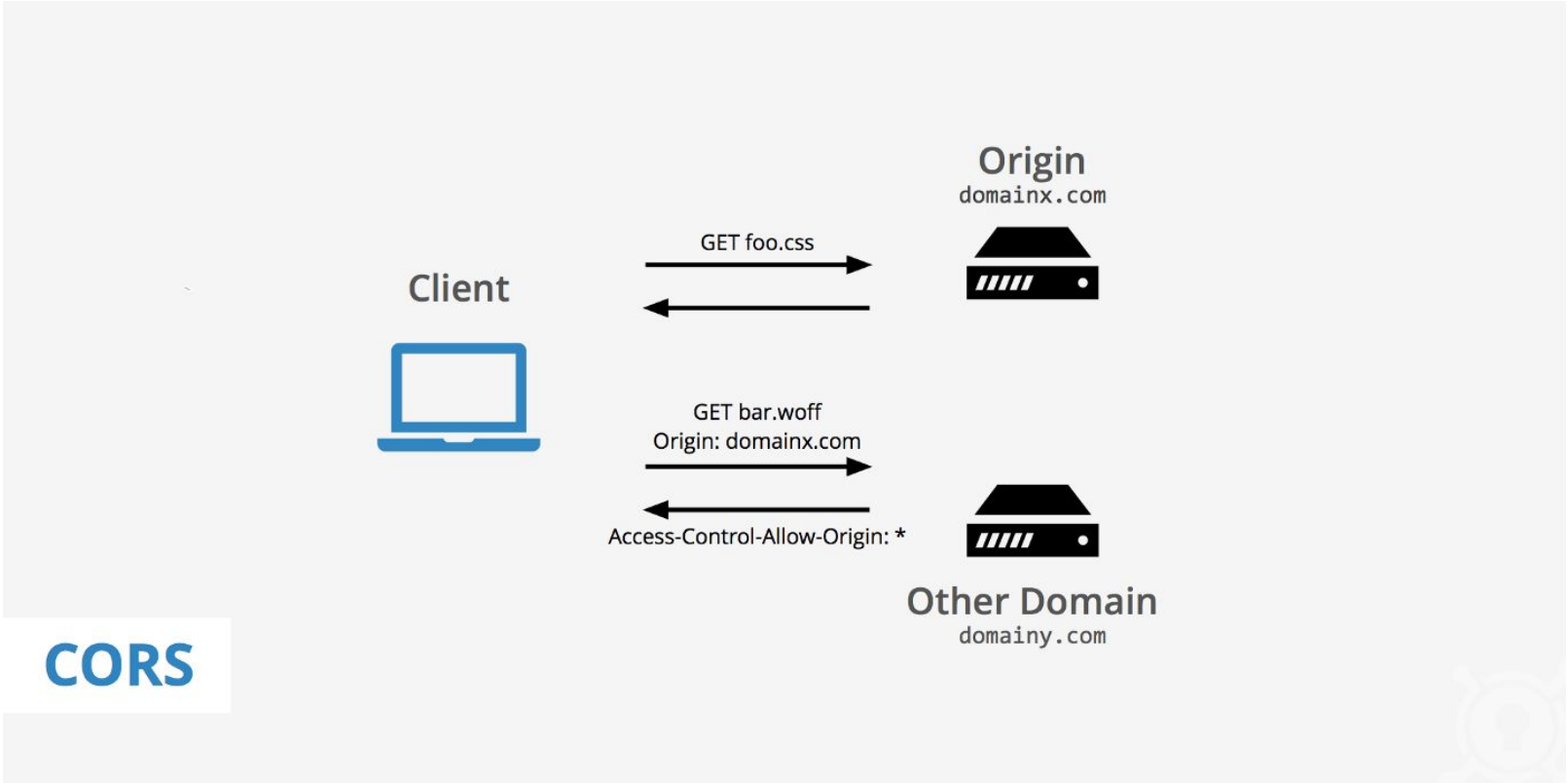


SOP

Gdyby przeglądarka podchodziła do SOP bardzo rygorystycznie, nie byłoby możliwe:

1. Umieszczenie na stronie z *originem A* obrazów, skryptów czy arkuszy CSS z *originu B* (wszystkie usługi typu CDN przestałyby działać)
2. Wywoływanie zapytań HTTP z *originu A* do *originu B* (a więc element <form> nie zostałby wysłany pod inny adres)
3. Odczytywanie i zapisywanie ciasteczek *originu A*, będąc na stronie innego *originu B*

Jak działa CORS?



Jak działa CORS?

Krok 1: żądanie klienta (przeglądarki)

Gdy przeglądarka wysyła żądanie cross-origin, przeglądarka dodaje nagłówek Origin z bieżącym pochodzeniem (protokół, host i port).

Krok 2: odpowiedź serwera

Po stronie serwera, gdy serwer widzi ten nagłówek i chce zezwolić na dostęp, musi dodać nagłówek Access-Control-Allow-Origin do odpowiedzi określającej pochodzenie żądania (lub *, aby zezwolić na dowolne pochodzenie).

Krok 3: przeglądarka otrzymuje odpowiedź

Gdy przeglądarka widzi tę odpowiedź z odpowiednim nagłówkiem Access-Control-Allow-Origin, umożliwia udostępnianie danych odpowiedzi witrynie klienta.

Przykład

W momencie, gdy klient (przeglądarka) próbuje nawiązać połączenie między domenowe, przeglądarka wyśle pakiet z nagłówkiem Origin równym adresowi klienta czyli w tym przypadku “Origin: http://localhost:24018”:

```
Accept:*/*
Accept-Encoding:gzip, deflate, sdch
Accept-Language:en-GB,en;q=0.8,en-US;q=0.6,pl;q=0.4
Cache-Control:max-age=0
Connection:keep-alive
Host:localhost:24523
Origin:http://localhost:24018
Referer:http://localhost:24018/Home/Index
User-Agent:Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/44.0.2403.125 Safari/537.36
```

Przykład

Jeśli domena pokryje się z regułami opisanymi za pomocą EnableCors, wtedy zostanie zwrócony pakiet z nagłówkiem Access-Control-Allow-Origin równym danej domenie tzn.:

```
Access-Control-Allow-Origin:http://localhost:24018
Cache-Control:no-cache
Content-Length:14
Content-Type:application/json; charset=utf-8
Date:Mon, 03 Aug 2015 18:09:57 GMT
Expires:-1
Pragma:no-cache
Server:Microsoft-IIS/10.0
X-AspNet-Version:4.0.30319
X-Powered-By:ASP.NET
X-SourceFiles:=?UTF-8?B?YzpcdXNlcnNccGlvdHJ6XGRvY3VtZW50c1x2aXN1YWwgc3R1ZGlvIDIwMTVcUHJ
vamVjdHNcV2ViQXBwbGljYXRpb24yXFdlYkFwcGxpY2F0aW9uMlxhcGlcVmFsdWVz?=
```

Zezwolenia dla originów, metod i nagłówków

```
CORS_ALLOWED_ORIGINS = [
    "https://example.com",
    "https://sub.example.com",
    "http://localhost:8080",
    "http://127.0.0.1:9000",
]
```

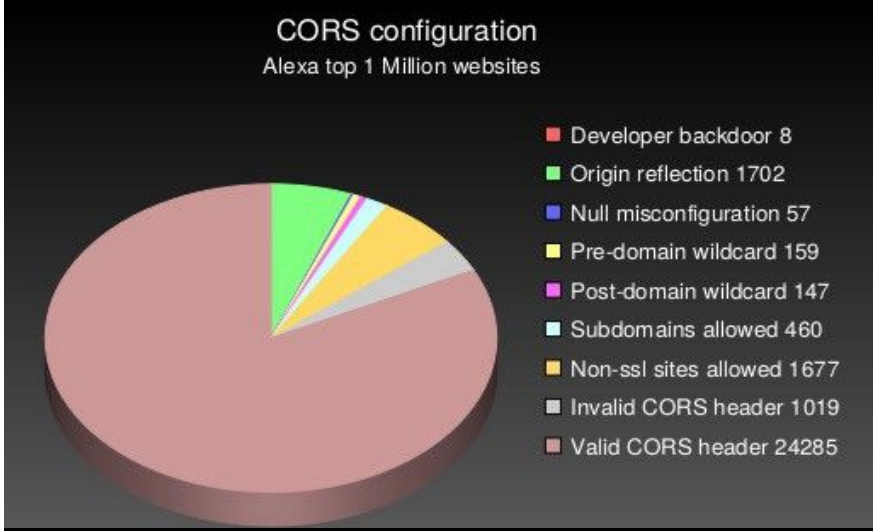
```
CORS_ALLOW_METHODS = [
    "DELETE",
    "GET",
    "OPTIONS",
    "PATCH",
    "POST",
    "PUT",
]
```

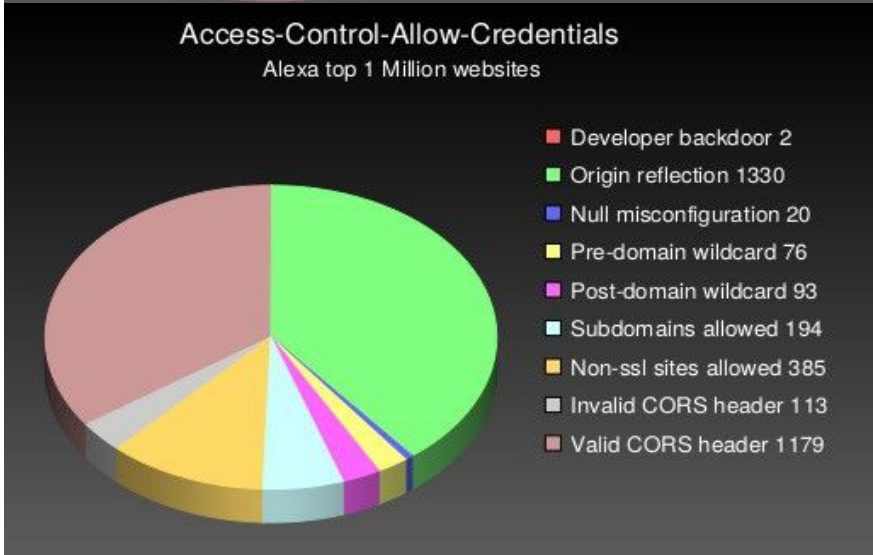
```
CORS_ALLOW_HEADERS = [
    "accept",
    "accept-encoding",
    "authorization",
    "content-type",
    "dnt",
    "origin",
    "user-agent",
    "x-csrftoken",
    "x-requested-with",
]
```

CORS - Statystyki

Nieoficjalne badanie przeprowadzone w czerwcu 2020 r. wykazało, że spośród 1 milionów witryn Alexa, które są najpopularniejsze, tylko 3% (29 514) witryn obsługuje CORS na swojej stronie głównej.

Jak wspomnieliśmy powyżej, aby móc przeprowadzić atak CORS, zasada Access-Control-Allow-Credentials musi być ustawiona na „true”. Przyglądając się witrynom obsługującym zarówno ACAO, jak i ACAC, to samo badanie wykazało, że prawie połowa z nich miała błędne konfiguracje CORS, które mógłby zostać wykorzystane.





How to find CORS ?

Do znalezienia możliwej podatności CORS można użyć dowolnej platformy do wyszukiwania błędów, takiej jak hakerone, bug-crowd, Synack, Open Bug Boun i innych platform.

How to check is vulnerable or not ?

Testowanie automatyczne i manualne:

W testowaniu automatyzacji używamy narzędzia z Git-Hub-a

Jest to skaner błędów konfiguracji CORS: https://github.com/RUB-NDS/CORStest

Do testowania manualnego korzystamy z dwóch narzędzi:

-> Curl

-> Burp Suite

Sprawdzenie podatności z wykorzystaniem pakietu burp

Teraz używamy pakietu burp, aby sprawdzić podatność CORS, wykonaj następujący krok:

1. “spider” domeny i osadzania parametrów.

2. Po tym wyślij żądanie w “repeaterze”

3. Następnie zmień pochodzenie i sprawdź odpowiedź

How can CORS based attacks be prevented ?

Określ dozwolone pochodzenie

Zezwalaj tylko na zaufane witryny

Nie umieszczaj na białej liście „null”

Wdróż odpowiednie polityki bezpieczeństwa po stronie serwera

CORS vulnerability with basic origin reflection

Zadanie 1

Uzyskanie klucza API administratora za pomocą skryptu JavaScript.

Ta witryna ma niezabezpieczoną konfigurację CORS, ponieważ ufa wszystkim źródłom.

Login: wiener

Hasło: peter

Web Security Academy

Korzystając z Burp Suite, sprawdź jaki typ podatności posiada strona. Można to sprawdzić poprzez wysłanie do strony zapytania z dodanym headerem:

Origin: https://example.com

Do znalezionej podatności za pomocą **exploit server** wstrzyknij kod HTML

Działający kod wyślij do ofiary oraz obserwuj consolę logów w celu odnalezienia klucza API administratora

Znaleziony kod zdekoduj przy pomocy Burp Suite

CORS vulnerability with trusted null origin

Zadanie 2

Uzyskanie klucza API administratora za pomocą skryptu JavaScript.

Ta witryna ma niezabezpieczoną konfigurację CORS, ponieważ ufa pochodzeniu „null”

Login: wiener

Hasło: peter

Korzystając z Burp Suite, sprawdź jaki typ podatności posiada strona. Można to sprawdzić poprzez wysłanie do strony zapytania z dodanym headerem:

Origin: null

Do znalezionej podatności za pomocą **exploit server** wstrzyknij kod HTML

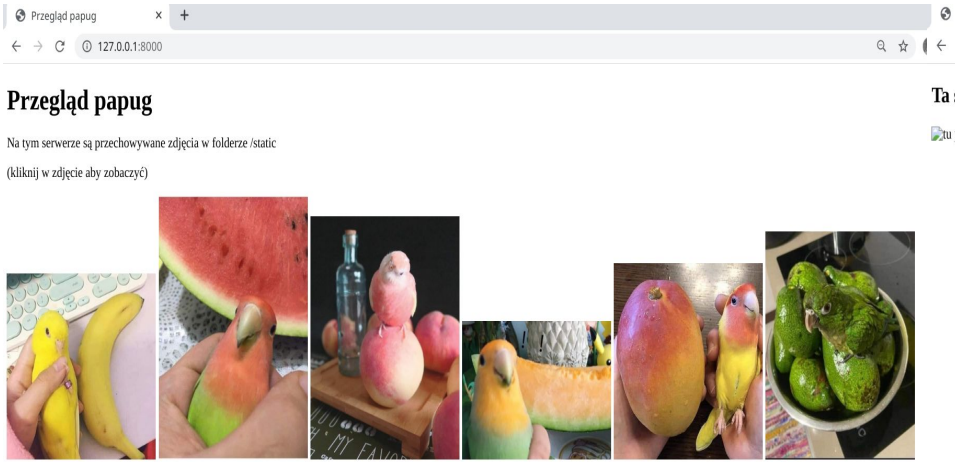
Działający kod wyślij do ofiary oraz obserwuj consolę logów w celu odnalezienia klucza API administratora

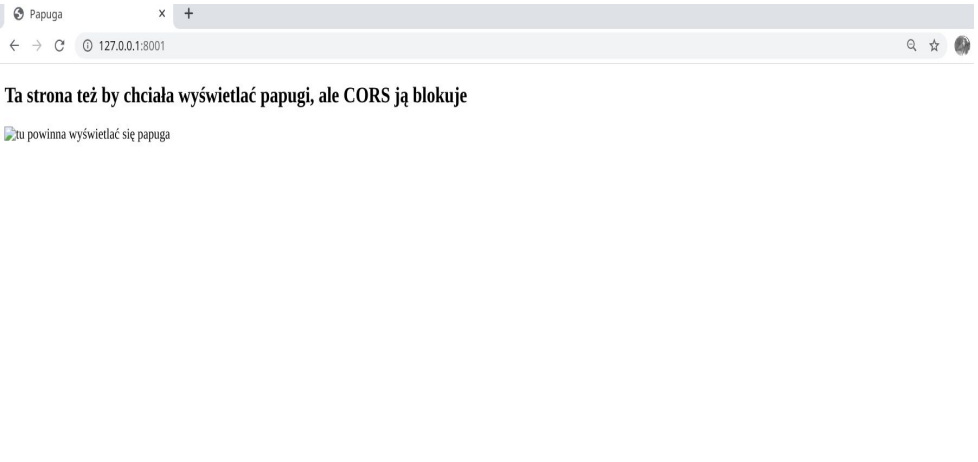
Znaleziony kod zdekoduj przy pomocy Burp Suite

CORS gaining access to images

Zadanie 3

Twoim zadaniem jest umożliwienie serwerowi BAWIM2 dostępu do obrazów z BAWIM.





CORS configuration in Django

Najpierw należy zainstalować django-cors-headers

```
  pip install django-cors-headers
```

W pliku settings.py należy dopisać ‘corsheaders’ w sekcji INSTALLED APPS

```
INSTALLED_APPS = [
   ...
   'corsheaders',
   ...
  ]
```

W pliku settings.py należy dopisać jakim domenom chcemy zezwolić na dostęp

```
MIDDLEWARE = [
  'django.middleware.security.SecurityMiddleware',
  ...
  'django.middleware.clickjacking.XFrameOptionsMiddleware',
  'corsheaders.middleware.CorsMiddleware',
  ]
```

W pliku settings.py należy dopisać jakim domenom chcemy zezwolić na dostęp

```
CORS_ORIGIN_ALLOW_ALL = False
CORS_ORIGIN_WHITELIST = [
  'http://localhost:8000',
  ]
```