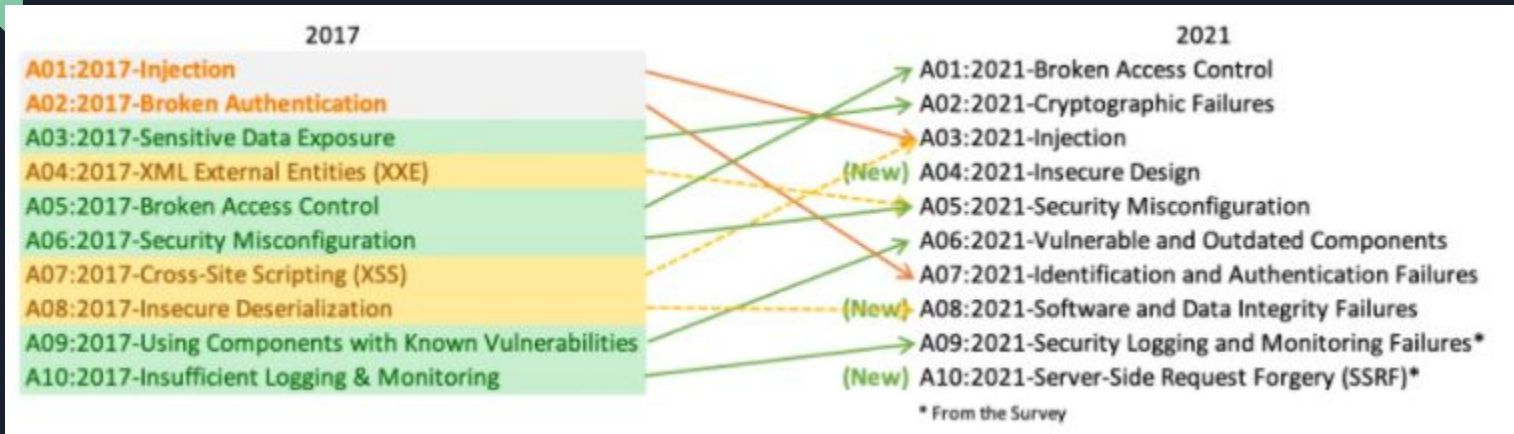




# SSRF

Server Side Request Forgery

# SSRF jako coraz powszechniejszy znaczący problem



A10:2021-Server-Side Request Forgery is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. **This category represents the scenario where the security community members are telling us this is important, even though it's not illustrated in the data at this time.**



# SSRF - co to takiego?

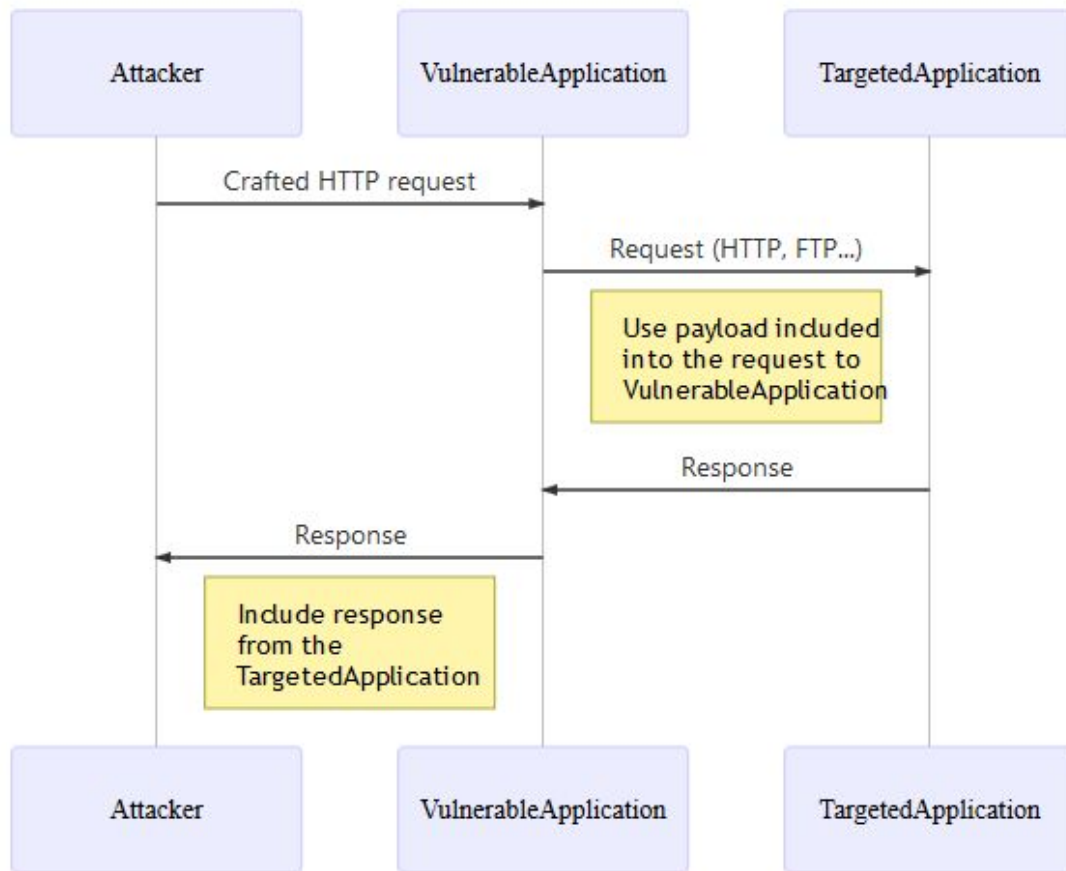
Jest to atak, podczas którego atakujący wykorzystuje serwer aplikacji do wysyłania zapytań HTTP(S)(ale nie tylko) do innych serwerów organizacji lub serwerów zewnętrznych, poprzez specjalnie spreparowane zapytania.

## Blind SSRF

Tak jak SSRF, ale atakujący nie otrzymuje żadnej informacji zwrotnej od aplikacji.

Jest trudniejszy do wykorzystania niż zwykły SSRF, w którym otrzymujemy odpowiedź zwrotną.

## Overview of a SSRF common flow



# Przykład w PHP wrażliwy na SSRF

```
<?php


/**
 * Check if the 'url' GET variable is set
 * Example - http://localhost/?url=http://testphp.vulnweb.com/images/logo.gif
 */
if (isset($_GET['url'])){
    $url = $_GET['url'];

    /**
     * Send a request vulnerable to SSRF since
     * no validation is being done on $url
     * before sending the request
     */
    $image = fopen($url, 'rb');

    /**
     * Send the correct response headers
     */
    header("Content-Type: image/png");

    /**
     * Dump the contents of the image
     */
    fpassthru($image);}
```

Brak sprawdzenia url => możliwość wysłania zapytania do każdej strony w Internecie albo zasobów na serwerze



# Misconfiguration Causes a Leak of One Hundred Million Financial Records

According to a statement by Capital One released on July 19, an unauthorized party gained access to the company's customer data: approximately 106 million individuals in the United States and Canada. Data was stored in Amazon S3 buckets but accessed using Capital One infrastructure. Capital One admits that it was a misconfiguration of the web application firewall that allowed the attacker to acquire suitable credentials and use certain commands to obtain data. However, presumptive evidence suggests that it was due to **Server-Side Request Forgery**.



# Jak się chronić przed SSRF?

- NIGDY NIE UFAJ TEMU CO WPISUJE UŻYTKOWNIK! Oczyszczyć dane wejściowe a następnie sprawdzić ich poprawność.

- Blacklist/Whitelist

- Nigdy nie wysyłać nieprzetworzonej treści odpowiedzi z serwera do klienta.

- Zezwalać tylko na schematy adresów URL, z których korzysta Twoja aplikacja.

- Wyłączyć/blokować nie używane protokoły (ftp://, file://, gopher://)



# Blacklist

Czarna lista, czyli wartości niedopuszczalnych adresów URL np.:

`http://localhost`

`http://127.0.0.1`

Jest to słabe rozwiązanie, ponieważ nie da się zablokować wszystkich możliwych adresów. Każdy adres można przedstawić w postaci:

- szesnastkowej (0x7f.1), ósemkowej (`http://0177.0.0.1`)
- zamkniętych znaków alfanumerycznych
- adresów IPv6 (`http://[0:0:0:0:0:ffff:127.0.0.1]`)
- decymalnej (`http://2130706433`) i oktalnej (`http://0177.0.0.1`) reprezentacji adresu IP (
- za pomocą zmiennych basha (np. przy użyciu curl'a)
- mix powyższych



## 127.0.0.1 lub localhost inaczej

0x7f.0x0.0x0.0x1  
0x7f001  
0x0a0b0c0d7f000001  
2130706433  
383.256.256.257  
0177.0.0.01  
017700000001  
00177.000.0000.000001  
0x7f.0.1  
0x7f.1  
00177.1  
00177.0x0.1  
127.0.0.1  
127.0.0.1  
127.0.0.1

027.00.0.1  
0x7f.0x0.0x0.0x1  
0x7f001  
2130706433  
383.256.256.257  
0177.0.0.01  
00177.000.0000.000001  
[::127.0.0.1]  
[::127.0.0.1%215]  
[::ffff:127.0.0.1]  
[::ffff:127.0.0.1%215]  
0x7f.0.1  
0x7f.1  
00177.1  
00177.0x0.1



Do obejścia blacklisty można użyć innych technik:

- utworzenie przekierowania do localhost'a
- DNS spoofing, rebinding



# Whitelist

Niektóre aplikacje pozwalają jedynie, aby wartości na wejściu zgadzały się ze wzorem, zaczynały się lub zawierały konkretne wartości.

Jest zazwyczaj bardziej rygorystyczna od czarnej listy.

Czasami można obejść poprzez np.:

- podanie usera za pomocą @: **`https://expected-host@evil-host`**
- wskazanie fragmentu adresu URL: **`https://evil-host#expected-host`**



# Zadania

## Zadanie 1.

<https://application.security/free-application-security-training/server-side-request-forgery-in-capital-one>

## Zadania 2 - 4

Na podstawie własnej aplikacji



# Zadanie 2

[Home](#) [Task 1](#) [Task 2](#) [Task 3](#)

## Task 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum semper accumsan velit id tristique. Aliquam eu ipsum elit. Donec ornare nunc et neque lacinia fringilla. In vitae vestibulum ipsum. Cras condimentum mauris ac nulla volutpat mattis. Maecenas vel dictum nisl. Cras diam arcu, pharetra eu tincidunt id, dictum vel lectus. Nulla quis massa at metus lobortis maximus id vel nunc. Aliquam sit amet magna non libero pretium ultrices. Morbi molestie, velit eu lacinia imperdiet, tellus enim facilisis dolor, eget maximus mauris risus id quam. Curabitur fringilla eleifend diam, et pretium justo molestie vel. Nunc euismod sollicitudin elit, cursus venenatis enim sagittis vitae. Phasellus dignissim nunc justo, id vehicula diam viverra id. Ut in turpis enim. Praesent pretium mauris ipsum, nec hendrerit mi commodo cursus.



# Zadanie 3

[Home](#) [Task 1](#) [Task 2](#) [Task 3](#)

## Task 2

### Blacklist

Get image.jpg file from /local/secret directory

Try to get me

# Zadanie 4

[Home](#) [Task 1](#) [Task 2](#) [Task 3](#)

## Task 3

### Whitelist

Try to get access to <http://192.168.100.109/secret/file>

Curl.me

<https://curl.se>

Search for sth on the Internet

Submit



# Bibliografia

<https://owasp.org/Top10/>

[https://cheatsheetseries.owasp.org/cheatsheets/Server Side Request Forgery Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html)

<https://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability/>

<https://sekurak.pl/czym-jest-podatnosc-ssrf-server-side-request-forgery/>

<https://portswigger.net/web-security/ssrf>

<https://www.acunetix.com/blog/web-security-zone/ssrf-misconfiguration-leak-one-hundred-million-financial-records/>

<https://0xn3va.gitbook.io/cheat-sheets/web-application/server-side-request-forgery>

<https://github.com/cujanovic/SSRF-Testing>

<https://github.com/swisskyrepo/PayloadsAllTheThings>