

Bezpieczeństwo Aplikacji WEB

Projekt

Raport 1

Miłosz Jagodziński CBE

242027

stopień magisterski

20.03.2022

termin Wtorek 8:15

Mgr inż. Przemysław Świercz

Gra 1	3
Poziom 1	3
Poziom 2	4
Poziom 3	4
Poziom 4	6
Poziom 5	6
Poziom 6	10
Poziom 7	11
Poziom 8	13
Koniec Gry 1	16
Gra 2	17
Poziom 1	17
Poziom 2	17
Poziom 3	18
Poziom 4	19
Poziom 5	21
Poziom 6	22
Poziom 7	23
Poziom 8	26
Poziom 9	26
Koniec Gry 2	27
Podsumowanie	27
Wnioski	27

Pierwszy projekt polegał na zagraniu w dwie gry programistyczne. Gry te związane były z bezpieczeństwem.

Cel: Łamanie haseł poprzez analizę kodu.

Gra 1

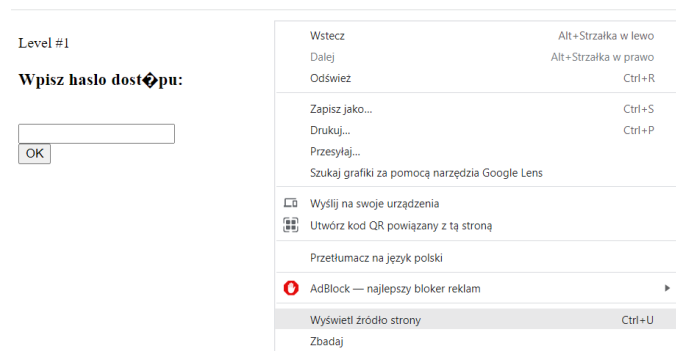
Poziom 1

Wygląd poziomu 1

Level #1

Wpisz hasło dostępu:

Postanowiłem zbadać kod strony. Postępowałem też tak w przypadku kolejnych zadań.



Od razu widać hasło na samym dole badanego kodu. Szukając hasła znalazłem je po `.value=` .

```
55 <script>
56 function sprawdz(){
57   if (document.getElementById('haslo').value=='a jednak umiem czytac') {self.location.href='ok_next.htm';} else {alert('Zle haselko :');}
58 }
59 </script>
60
```

Hasło = “a jednak umiem czytać”.

Level #1

Wpisz hasło dostępu:

Poziom 2

Kod w kolejnym poziomie.

```
1 <HTML>
2 <script src="haselko.js"></script>
3 <script>
4 function sprawdz(){
5   if (document.getElementById('haslo').value==has) {self.location.href=adresik;} else {alert('Nie... to nie to haslo...');}
6 }
7 </script>
8 <br>Level #2
9 <h3>Wpisz haslo dostepu:</h3>
10 <br><input type="text" name="haslo" id="haslo">
11 <br><input type="button" value="OK" onClick="sprawdz()">
12 </HTML>
13
```

Widzimy tu skrypt, który można otworzyć, a więc wszedłem w niego. Poniżej widać jego zawartość:

```
var has='to bylo za proste';
var adresik='formaster.htm';
```

Hasło = “to było za proste”.

Level #2

Wpisz haslo dostepu:

Poziom 3

Kod poziomu 3

```
1 <HTML>
2 <script>
3 function right(e) {
4   if (navigator.appName == 'Netscape' &&
5       (e.which == 3 || e.which == 2))
6     return false;
7   else if (navigator.appName == 'Microsoft Internet Explorer' &&
8             (event.button == 2 || event.button == 3)) {
9     alert('Prawy nie dziala...');
10    return false;
11  }
12  return true;
13 }
14 var dod='unknow';
15 function prawy(tx){
16   var txt=tx;
17   document.onmousedown=right;
18   if (document.layers) window.captureEvents(Event.MOUSEDOWN);
19   window.onmousedown=right;
20 }
21 prawy();
22 var literki='abcdefgh';
23 var ost='';
24 function losuj(){
25   ost=literki.substring(2,4)+'qwe'+dod.substring(3,6);
26 }
27
28 function sprawdz(){
29   losuj();
30   if (document.getElementById('haslo').value==ost) {self.location.href=ost+'.htm';} else {alert('Nie... to nie to haselko...');}
31 }
32 </script>
33 <br>Level #3
34 <h3>Wpisz haslo dostepu:</h3>
35 <br><input type="text" name="haslo" id="haslo">
36 <br><input type="button" value="OK" onClick="sprawdz()">
37 </HTML>
```

W kodzie znajduję się wiele niepotrzebnych znaków. Ja skupiłem się na tych, które są w funkcji losuj. Tyle kodu jest po to, aby wprowadzić małe zamieszanie.

Poniżej ciąg znaków

```
literki = 'abcdefgh'.  
var ost = ' ';  
var dod='unknown';
```

Również znajduję się funkcje losuj gdzie:

```
function losuj() {  
ost=literki.substring(2,4)+'qwe'+dod.substring(3,6);
```

01234567

Jak widać, aby powstał `ost` trzeba z literki `abcdefgh` wziąć ciąg(2,4) = `'cd'`.

Następnie dodać `'qwe'` do `'cde'`. `cde + qwe = cdqwe`

012345

Kolejny krok to trzeba wyciągnąć `dod.substring(3,6)` z `'unknow'` = `'now'`. Dodałem to do poprzedniego `cdqwe + now = cdqwenow`.

Hasło = "cdqwenow".

Level #3

Wpisz hasło dostępu:

Poziom 4

Poniżej znajduję się kod poziomu 4

```
<HTML>  
<script>  
function sprawdz(){  
  zaq=document.getElementById('haslo').value;  
  if (isNaN(zaq)) {alert('Zle haslo!')} else {  
    wynik=(Math.round(6%2)*(258456/2))+(300/4)*2/3+121;  
    if (zaq==wynik) {self.location.href='go'+wynik+'.htm';} else {alert('Zle haslo!')}  
  }  
</script>  
<br>Level #4  
<h3>I co by tu teraz zrobic?</h3>  
<br><input type="text" name="haslo" id="haslo" size=20>&nbsp;<input type="button" value="?" onClick="sprawdz()">  
</HTML>
```

Jak widać jest to zadanie matematyczne.

```
if (isNaN(zaq)) {alert('Zle haslo!')} else {  
  wynik=(Math.round(6%2)*(258456/2))+(300/4)*2/3+121;  
  if (zaq==wynik) {self.location.href='go'+wynik+'.htm';} else {alert('Zle haslo!')}
```

wynik=(Math.round(6%2)*(258456/2))+(300/4)*2/3+121;

Hasłem był wynik powyższego zadania matematycznego. Wynik = 171.
Hasło = "171"

Poziom 5

Poziom 5 również był zadaniem matematycznym. Tylko trochę trudniejszym. Trzeba wpisać cyfrę pomocniczą (b) w odpowiednim czasie (a), aby uzyskać dostęp do kolejnego zadania. Czas oznaczyłem "a, a cyfrę pomocniczą "b".

Level #5

Zamek czasowy

a
57

b

Cyfra pomocnicza:

[wejdź]

Poniżej przedstawiony jest kod levelu 5:

```
1 <HTML>
2 <script>
3 var now = new Date();
4 var seconds = now.getSeconds();
5
6 function czas(){
7 now = new Date();
8 seconds = now.getSeconds();
9 txt.innerHTML=seconds;
10 setTimeout('czas()',1);
11 }
12
13 function sprawdz(){
14 ile=((seconds*(seconds-1))/2)*(document.getElementById('pomoc').value%2);
15 if (ile==861) {self.location.href=seconds+'x.htm'} else {alert('Zle haslo!');}
16 }
17 </script>
18 <br>Level #5
19 <h3>Zamek czasowy</h3>
20 <br><div id="txt"></div>
21 <br>Cyfra pomocnicza: <input type="text" size=3 name="pomoc" id="pomoc"><br>
22 <br><input type="button" value="[wejdź]" onClick="sprawdz()">
23 <script>czas();</script>
24 </HTML>
25
```

Hasłem jest cyfra pomocnicza wpisana w odpowiedniej sekundzie Zamka Czasowego.

```
ile=((seconds*(seconds-1))/2)*(document.getElementById('pomoc')
).value%2);
ile == 861;
```

Warto zwrócić uwagę, że na końcu równania znajduje się działanie modulo 2, a więc każda liczba parzysta będzie miała resztę z dzielenia 0, a każda nieparzysta resztę 1. Liczby parzyste odpadają, ponieważ mnożenie przez 0 zawsze będzie dawało wynik 0.

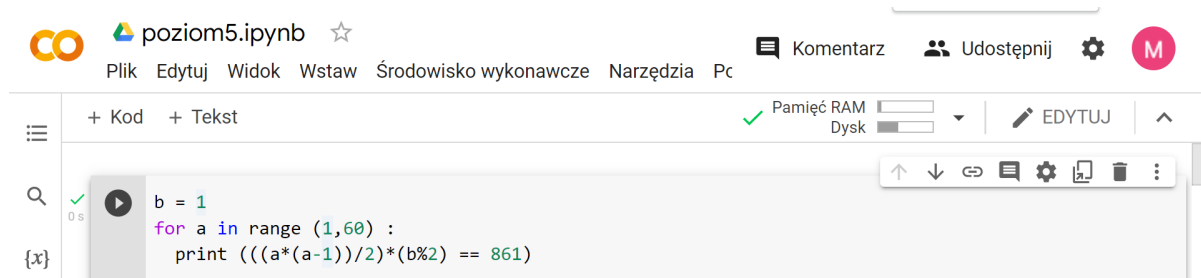
1. Pierwszy wniosek poszukiwana wartość nie jest parzysta tylko jest nieparzysta, ponieważ $2k+1\%2$ dla $k \in (1,2,3,4,...)$ zawsze da nam wynik 1.
2. Drugi wniosek poszukiwana wartość będzie obojętnie jaką liczbą nieparzystą.
3. Trzeci wniosek wartość a znajdują się w przedziale od 1 do 59, ponieważ tyle trwa jedna minuta w Zamku Czasowym.

Ułatwiłem sobie zadanie wykorzystując Colab i stworzyłem w nim proste równanie. Gdzie u mnie:

'a'=seconds Zamek Czasowy

'b'=value Hasło

Język to Python



```
b = 1
for a in range(1,60):
    print(((a*(a-1))/2)*(b%2) == 861)
```

Wyniki wyszedł następujący:

[illegible]

Wrzuciłem wynik do Notepad, aby szybko znaleźć wartość mojego a:

```
1 False
2 False
3 False
4 False
5 False
6 False
7 False
8 False
9 False
10 False
11 False
12 False
13 False
14 False
15 False
16 False
17 False
18 False
19 False
20 False
21 False
22 False
23 False
24 False
25 False
26 False
27 False
28 False
29 False
30 False
31 False
32 False
33 False
34 False
35 False
36 False
37 False
38 False
39 False
40 False
41 False
42 True
43 False
44 False
45 False
46 False
47 False
48 False
49 False
```

Gdzie True występuje przy $a = 42$. Wiemy, w której sekundzie trzeba wpisać cyfrę nieparzystą.

Level #5

Zamek czasowy

42

Cyfra pomocnicza:

Hasło trzeba wpisać w 42 sekundzie Zamka Czasowego i hasło jest obojętnie jaka liczba nieparzysta.

Hasło = $2b+1$, gdzie b należy do $\{1,2,3,4,\dots\}$ i wpisać w 42 sekundzie.

Poziom 6

Poziom 6 polegał na zabawie ze znakami. Musiałem odnaleźć wartość zmiennej `hsx` po kilku działaniach, dzięki temu poszedłem dalej do przedostatniego poziomu 7.

```
1 <HTML>
2 <script>
3 var lit='abcdqepolsrc';
4 function sprawdz(){
5   var licznik=0;
6   var hsx='';
7   var znak='';
8   zaq=document.getElementById('haslo').value;
9   for (i=1; i<=5; i+=2){
10    licznik++;
11    if ((licznik%2)==0) {znak='_';} else {znak='x';}
12    hsx+=lit.substring(i,i+1)+znak;
13  }
14  hsx+=hsx.substring(hsx.length-3,hsx.length);
15  if (zaq==hsx) {self.location.href=hsx+'.htm';} else {alert('Zle haslo!');}
16 }
17 </script>
18 <br>Level #6
19 <h3>Wprowadz haslo:</h3>
20 <br><input type="text" name="haslo" id="haslo">
21 <br><input type="button" value="OK" onClick="sprawdz()">
22 </HTML>
23
```

W kodzie znajdują się następujące zmienne:

```
var lit='abcdqepolsrc';
var licznik=0;
var hsx='';
var znak='';
```

Zauważyłem również pętlę.

```
for (i=1; i<=5; i+=2){
  licznik++;
  if ((licznik%2)==0) {znak='_';} else {znak='x';}
  hsx+=lit.substring(i,i+1)+znak;
}
```

Wiemy, że pętla wykona się w 1, 3 i 5. Dlatego trzeba będzie przejść przez 3 kroki w poziomie 6.

Krok 1, gdzie $i = 1$

Widzimy, że jeżeli nie dostaniemy resztę z dzielenia równą 0 to wtedy znak = `_`, a jeżeli nie równą 0 to znak = `x`.

Dlatego:

licznik = 1 (nieparzysta), więc:

znak = `x`

Mamy **bx**

Krok 2

gdzie $i = 3$
licznik = 2 (parzysta), więc:
znak = _
Mamy **bx**d_

Krok 3

gdzie $i = 5$
licznik = 3 (nieparzysta), więc:
znak = _
Mamy **bx**d_ex

Pętla kończy się po tych trzech krokach. Wychodzi nam:
hsx = bxd_ex (Jest to 6 znaków)

Koniec 3 kroków

Jeszcze trzeba wykonać kolejne działanie:
`hsx+=hsx.substring(hsx.length-3,hsx.length);`

Czyli będziemy wybierać substring[3,6] z 'bx**d_ex**' Gdzie dodałem zabrane **_ex** do naszego bxd_ex. Wynik wyszedł następujący bxd_ex + ex = bxd_ex_ex.
Otrzymałem następujące hasło bxd_ex_ex.

Hasło = "bxd_ex_ex"

Level #6

Wprowadz hasło:

Poziom 7

Kod w zadaniu 7.

```
1 <HTML>
2 <script>
3 function sprawdz(){
4   zaq=document.getElementById('haslo').value;
5   wyn='';
6   for (i=0; i<=zaq.length-1; i++){
7     lx=zaq.charAt(i);
8     ly='';
9     if (lx=='a') {ly='z'}
10    if (lx=='b') {ly='y'}
11    if (lx=='c') {ly='x'}
12    if (lx=='d') {ly='w'}
13    if (lx=='e') {ly='v'}
14    if (lx=='f') {ly='u'}
15    if (lx=='g') {ly='t'}
16    if (lx=='h') {ly='s'}
17    if (lx=='i') {ly='r'}
18    if (lx=='j') {ly='q'}
19    if (lx=='k') {ly='p'}
20    if (lx=='l') {ly='o'}
21    if (lx=='m') {ly='n'}
22    if (lx=='n') {ly='m'}
23    if (lx=='o') {ly='l'}
24    if (lx=='p') {ly='k'}
25    if (lx=='q') {ly='j'}
26    if (lx=='r') {ly='i'}
27    if (lx=='s') {ly='h'}
28    if (lx=='t') {ly='g'}
29    if (lx=='u') {ly='f'}
30    if (lx=='v') {ly='e'}
31    if (lx=='w') {ly='d'}
32    if (lx=='x') {ly='c'}
33    if (lx=='y') {ly='b'}
34    if (lx=='z') {ly='a'}
35    if (lx==' ') {ly='_'}
36    wyn+=ly;
37  }
38  if (wyn=='plxszn_xrv') {self.location.href=wyn+'.htm';} else {alert('Zle haslo!');}
39 }
40 </script>
41 <br>Level #7
42 <h3>Wprowadz haslo:</h3>
43 <br><input type="text" name="haslo" id="haslo">
44 <br><input type="button" value="OK" onClick="sprawdz()">
45 </HTML>
46
```

W kodzie widać pętlę i alfabet. Podejrzewałem, że było to coś podobnego do szyfru Cezara. Każda litera alfabetu ma swój inny niepowtarzalny odpowiednik innej litery.

```
wyn+=ly;
}
if (wyn=='plxszn_xrv') {self.location.href=wyn+'.htm';} else {alert('Zle haslo!');}
}
```

Również zauważyłem `wyn = 'plxszn_xrv'`. Dlatego postanowiłem zamienić litery `ly` na litery `lx`.

'p' na 'k'

'l' na 'o'

'x' na 'c'
's' na 'h'
'z' na 'a'
'n' na 'm'
'_' na ''
'x' na 'c'
'r' na 'i'
'v' na 'e'

Hasło = 'kocham cie'

Level #7

Wprowadz hasło:

Poziom 8

Poziom 8 był ostatnim zadaniem. Poniżej jest kod z tego poziomu, jak widać posiada 19 linii.

```
1 <HTML>
2 <script>
3 var roz='dsabdkgsawqqqlsahdas'; var tmp=roz.substring(2,5)+roz.charAt(12);
4 document.write('<s'+c+'r'+i+'p'+t src="%7A%73%65%64%63%78%2E%6A%73"></s'+c+'r'+i+'p'+t'+>');
5 function sprawdz(){
6   zaq=document.getElementById('haslo').value; wyn=''; alf='qwertyuioplkjhgfdsazxcvbnm';
7   qet=0; for (i=0; i<=10; i+=2){
8     get+=10; wyn+=alf.charAt(qet+i); qet++;}
9   wyn+=eval(ax*bx*cx);
10  if (wyn==zaq) {self.location.href=wyn+'.htm';} else {alert('Zle haslo!');}
11  }
12 </script>
13 <br>Level #8
14 <h3>Wprowadz hasło:</h3>
15 <script src="%70%61%73%73%77%64.js"></script>
16 <br><input type="text" name="haslo" id="haslo">
17 <br><input type="button" value="OK" onClick="sprawdz()">
18 </HTML>
19
```

Zwróciłem uwagę na poniższy skrypt. Jednak była to podpucha.

```
<h3>Wprowadz hasło:</h3>
<script src="%70%61%73%73%77%64.js"></script>
<br><input type="text" name="haslo" id="haslo">
```

```
// -----  
// Niespodzianka!  
// Tu nie ma hasla...  
// szukaj dalej...  
// -----
```

Kolejnym rzeczą, która przykuła moją uwagę był ciąg znaków
`src="%7A%73%65%64%63%78%2E%6A%73"`

Okazało się, że był to ciąg znaków szesnastkowych. Przetłumaczyłem go w konwerterze na znaki ASCII.

<https://www.binaryhexconverter.com/hex-to-ascii-text-converter>

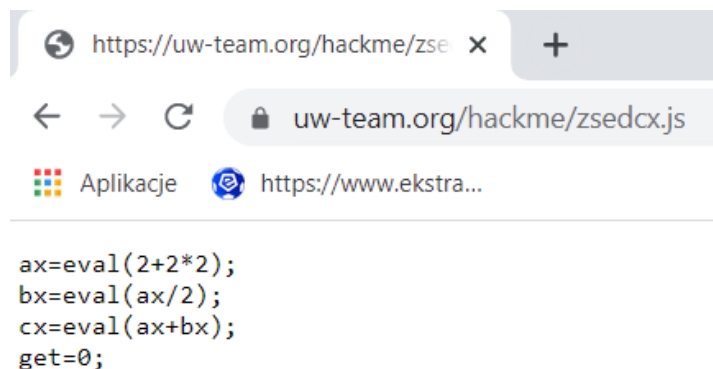
Hex to Ascii (String) Converter

To use this **hex to string converter**, type a hex value like 6C 6F 76 65 and into the left field below and hit the Convert button. You will get the according string.

Facebook Twitter

Hexadecimal Value	Ascii (String)
<input type="text" value="7A73656463782E6A73"/>	<input type="text" value="zsedcx.js"/>
<input type="button" value="Convert"/>	
swap conversion: Ascii Text To Hexadecimal Converter	

Otrzymałem następującą informację zsedcx.js. Postanowiłem to przeszukać.



Dzięki temu uzyskałem nowe zmienne:

```
ax=eval(2+2*2);  
bx=eval(ax/2);  
cx=eval(ax+bx);  
get=0;
```

Następnie wypisałem sobie kolejne zmienne:

```
var roz='dsabdkgsawqqqlsahdas';  
wyn='';  
alf='qwertyuioplkjhgfdsazxcvbnm';
```

Również zbadalem pętle.

```
get=0; for (i=0; i<=10; i+=2){  
get+=10; wyn+=alf.charAt(get+i); get++;}
```

Pętla ta wykona się 6 razy. Dla $i = 0, 2, 4, 6, 8, 10$.

Tak jak w poprzednich zdaniach podzieliłem sobie na kroki.

Krok 1

$i = 0$

$wyn += alf.charAt(get+i)$

Bierzemy 1 znak z alf, gdzie równa się **q**.

Krok 2

$i = 2$

$wyn += alf.charAt(get+i)$

Bierzemy 3 znak z alf, gdzie równa się **r**.

Krok 3

$i = 4$

$wyn += alf.charAt(get+i)$

Bierzemy 6 znak z alf, gdzie równa się **u**.

Krok 4

$i = 6$

$wyn += alf.charAt(get+i)$

Bierzemy 9 znak z alf, gdzie równa się **p**.

Krok 5

$i = 8$

$wyn += alf.charAt(get+i)$

Bierzemy 12 znak z alf, gdzie równa się **j**.

Krok 6

$i = 10$

$wyn += alf.charAt(get+i)$

Bierzemy 15 znak z alf, gdzie równa się **f**.

Podsumowując wykonywano “skoki” o 3 znaki w ciągu alf. Dzięki temu uzyskałem **grupjf**

Pamiętać trzeba było jeszcze o $wyn += eval(ax*bx*cx);$.

Wyliczyłem następujące działania.

$ax = eval(2+2*2); \quad = 6$

$bx = eval(ax/2); \quad = 3$

$cx = eval(ax+bx); \quad = 9$

$wyn = ax*bx*cx = 162$

Otrzymany wynik 162 dołożyłem do otrzymanego wcześniej ciągu znaków grupjf. Dzięki temu uzyskałem hasło.

Hasło = "grupjf162"

Level #8

Wprowadz hasło:

Koniec Gry 1

Po przejściu ostatniego ósmego poziomu, otrzymałem następujący komunikat i gratulację. Wnioski z wykonywanych zadań zostały przedstawione na samym końcu raportu.



You win!

Gratulacje!

Właśnie przeszedłeś grę Hackme 1.0 by Unknow!

Gratuluje cierpliwości :)

Gra 2

Gra numer dwa była bardzo podobna do poprzedniej gry, składała się natomiast z 9 poziomów.

Poziom 1

Od razu po rozpoczęciu poziomu 1, postanowiłem zbadać kod, tak jak robiłem to we wcześniejszych zadaniach.

Zadanie to było bardzo proste, ponieważ hasło od razu zauważyłem, ponieważ input oznacza, że musi być na wejściu wartość `value="text"`.

```
1 <html><head><title>Hackme 2.0 - by Unknow</title></head><body text="white" bgcolor="black" link="yellow"
  vlink="yellow" alink="yellow">
2 <script>
3 function spr(){
4   if (document.getElementById('formularz').value==document.getElementById('haslo').value){
5     self.location.href=document.getElementById('haslo').value+'.htm'; } else {alert('Nie, to nie to haselko :(');}
6   }
7 </script>
8 <h3>Hackme 2.0 - level #1</h3>
9 Podaj haselko: <input type="password" name="haslo" id="haslo"><input value="text" name="formularz" id="formularz"
  type="hidden"><input type="button" onClick="spr()" value="Go!">
10 </body></html>
```

Hasło = "text"



Poziom 2

Poniżej znajdują się kod poziomu 2. Jak widać znowu trzeba zmienić hex na ASCII.

```
1 <html><head><title>Hackme 2.0 - by Unknow</title></head><body text="white" bgcolor="black"
  link="yellow" vlink="yellow" alink="yellow">
2 <script>
3 function spr(){
4   if (document.getElementById('haslo').value==unescape('%62%61%6E%61%6C%6E%65')) {
5     self.location=document.getElementById('haslo').value+'.htm'; } else { alert('Zle haslo!'); }
6   }
7 </script>
8 <h3>Hackme 2.0 - level #2</h3>
9 Podaj haslo: <input type="password" name="haslo" id="haslo"> <input type="button" onClick="spr()"
  value="Break me!">
10 </body></html>
```

Ponownie wykorzystałem konwerter na znaki ASCII.

Hexadecimal Value

Ascii (String)

Convert

swap conversion: [Ascii Text To Hexadecimal Converter](#)

Dzięki temu odnalazłem hasło.

Hasło = “banalne”

Poziom 3

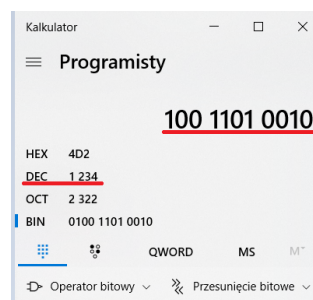
Poziom 3 był również prosty. Do rozwiązania tego zadania użyłem kalkulatora w komputerze. Poniżej widać kod.

```

1 <html><head><title>Hackme 2.0 - by Unknow</title></head><body text="white" bgcolor="black"
2 link="yellow" vlink="yellow" alink="yellow">
3 <script>
4 function binary(liczba) {
5   return liczba.toString(2);
6 }
7 function spr(){
8   if (binary(parseInt(document.getElementById('haslo').value))==10011010010) {
9     self.location=document.getElementById('haslo').value+'.htm'; } else { alert('Zle! \nPodstawy
10 matematyki sie klaniaja :)');}
11 }
12 </script>
13 <h3>Hackme 2.0 - level #3</h3>
14 Podaj haselko: <input type="password" name="haslo" id="haslo"> <input type="button" onClick="spr()"
15 value="Click me baby!">
16 </body></html>

```

Wystarczyło zamienić liczbę binarną 10011010010 na liczbę dziesiętną 1234. Dzięki temu uzyskałem rozwiązanie poziomu 3.



Hasło = “1234”

Po wpisaniu hasła wystąpił następujący komunikat.

Komunikat ze strony uw-team.org

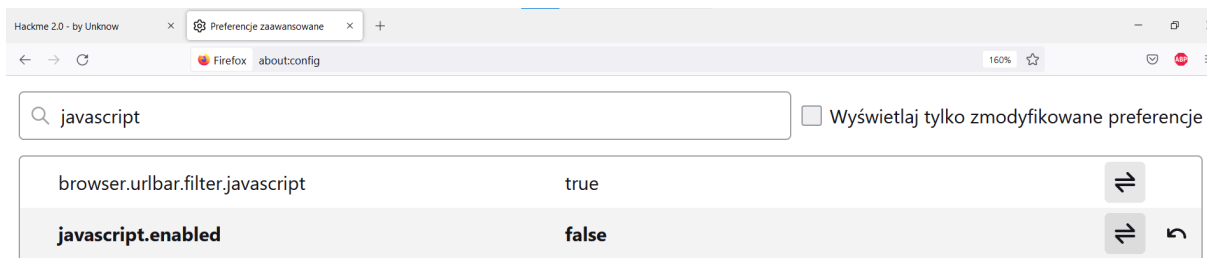
podaj hasło:

wpisz X aby zatrzymać skrypt

OK

Anuluj

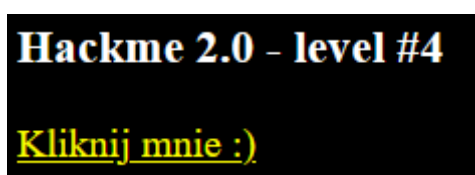
Trzeba było wyłączyć javascript, aby przejść dalej.



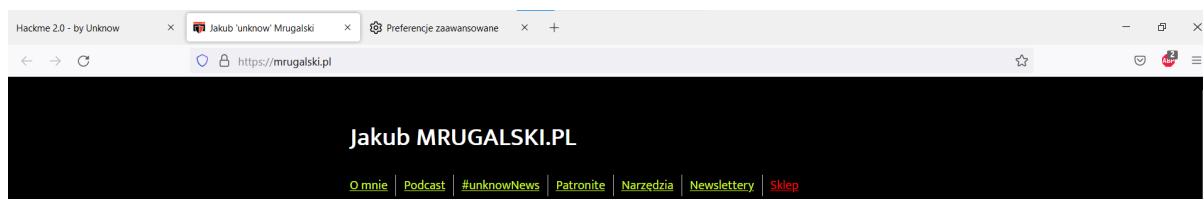
Po wyłączeniu javascript, mogłem przystąpić do następnego zadania.

Poziom 4

Poziom 4 wyglądał następująco.



Próbowałem wejść w “Kliknij mnie :)”, ale wyskoczyła mi strona pokazana poniżej.



Dlatego postanowiłem zbadać lepiej kod strony, więc otrzymałem poniższe informacje.



Zauważyłem poniższy skrypt.

```
<script>
  haslo=''; cos=parseInt(unescape('%32%35%38')); while ((haslo!=cos.toString(16)) && (haslo!='X')){ haslo=prompt('podaj
  haslo:\nwpisz X aby zatrzymać skrypt',''); } if (haslo==cos.toString(16)) { self.location=haslo+'.php';} else
  {self.location='http://www.uw-team.org/';}
</script>
```

Wypisałem sobie wszystkie zmienne.
haslo=' ' pusta zmienna

`cos=parseInt(unescape('%32%35%38'));` Znowu użyłem konwertera na ASCII

Również zwróciłem uwagę na pętlę.

```
while ((haslo!=cos.toString(16)) && (haslo!='X')){  
haslo=prompt('podaj haslo:\nwpisz X aby zatrzymac skrypt','');  
}  
if (haslo==cos.toString(16)) { self.location=haslo+'.php';}  
else {self.location='http://www.uw-team.org/';}
```

Tak jak wspomniałem wcześniej, trzeba było użyć konwertera. Dlatego z %32%35%38 otrzymałem 258.

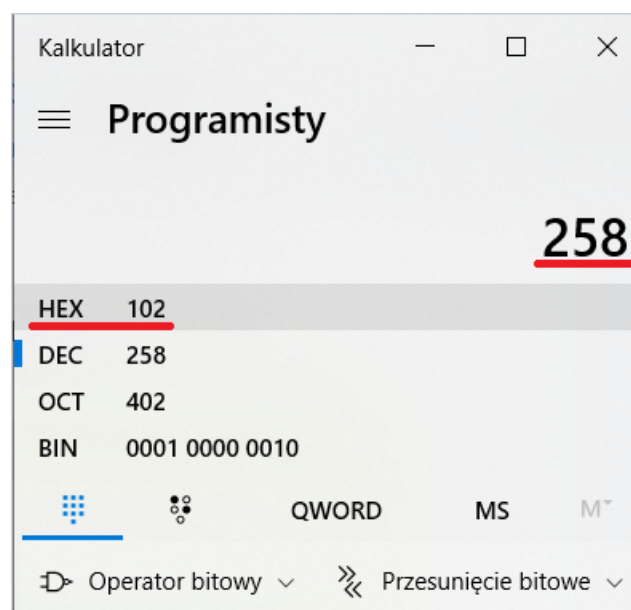
Ascii Text To Hexadecimal Converter'." data-bbox="125 296 862 410"/>

Hexadecimal Value	Ascii (String)
323538	258

Convert

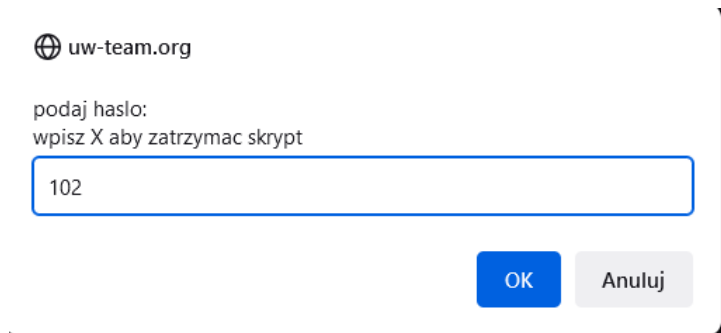
swap conversion: [Ascii Text To Hexadecimal Converter](#)

Następnie liczbę 258 zamieniłem na hex 102.



Kolejno trzeba było z powrotem włączyć javascript, aby móc wpisać hasło.

Hasło = "102"



Poziom 5

Poziom 5 został napisany w PHP, dlatego nie można było już podejrzec kodu. Trzeba było ugryźć to zadanie od innej strony. Poniżej wygląd poziomu 5.

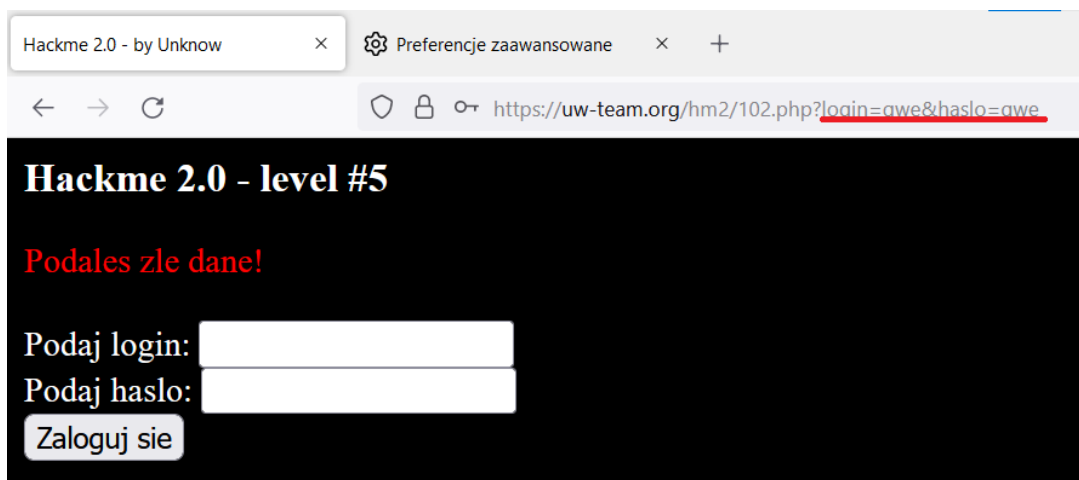
Hackme 2.0 - level #5

Podaj login:
Podaj haslo:

Ten etap napisany jest w PHP, wiec nie mozesz podgladnac jego zrodla.
Oto zrodlo skryptu:

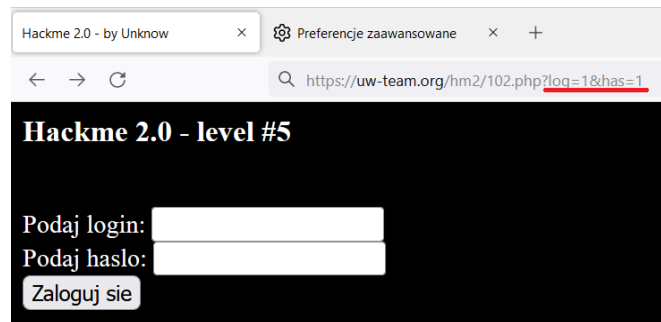
```
if (!isset($haslo)) {$haslo='';}  
if (!isset($login)) {$login='';}  
if ($haslo=="tu jest haslo") {$has=1;}  
if ($login=="tu jest login") {$log=1;}  
if (($has==1) && ($log==1)) { laduj nastepny level } else { powroc do tej strony }
```

Spróbowałem wpisać losowe hasło.

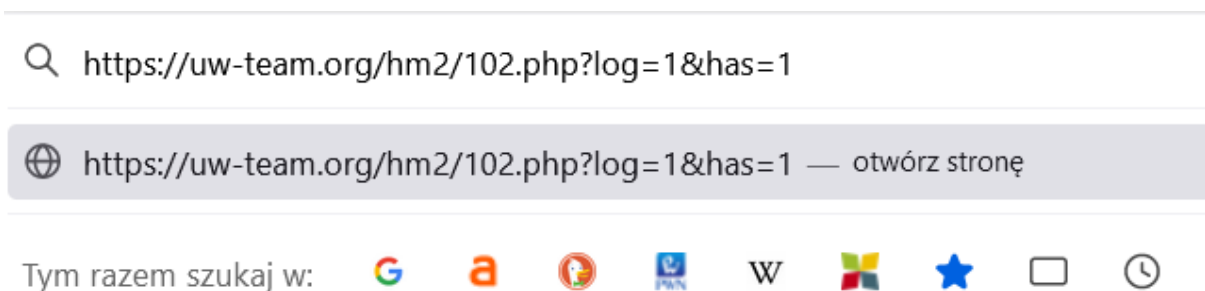


Jak widać wszystko zależy od wpisania loginu i hasła, ponieważ zostają to pobrane przez GET i wysłane do zmiennych.

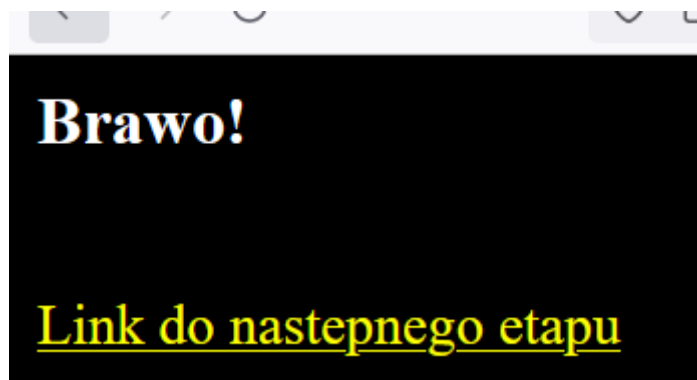
Po przeanalizowaniu poniższego skryptu, stwierdziłem, że trzeba zmienić link strony, aby przejść do kolejnego poziomu. Ostatnia linijka skryptu zawierała informacje umożliwiające przejście dalej. Trzeba było dopisać na końcu linku `?log=1&has=1`.



Następnie kliknąłem otwórz stronę



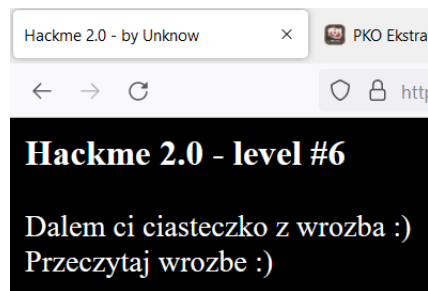
Udało mi się przejść do kolejnego etapu.



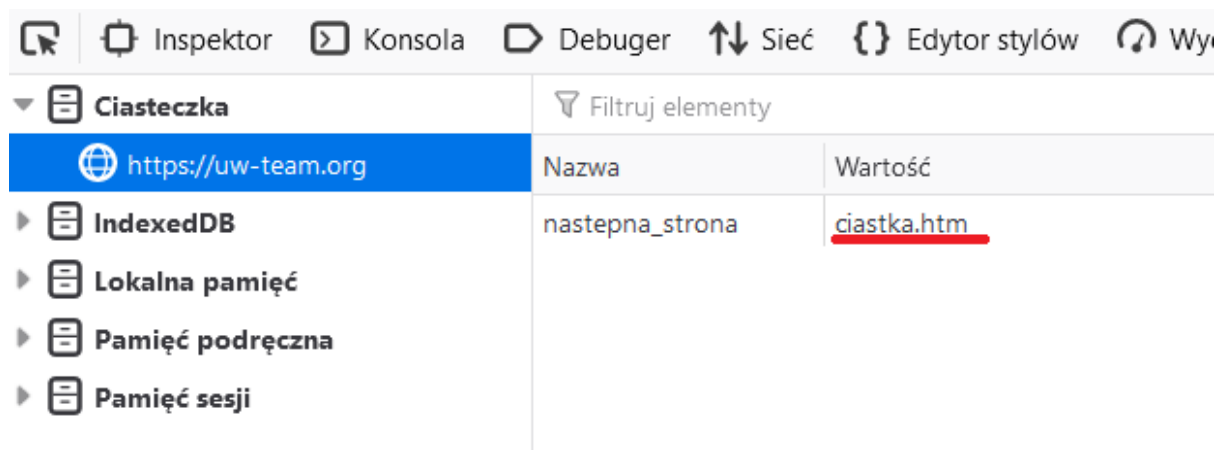
Wynik = wpisanie "<https://uw-team.org/hm2/102.php?log=1&has=1>" i otworzenie tej strony

Poziom 6

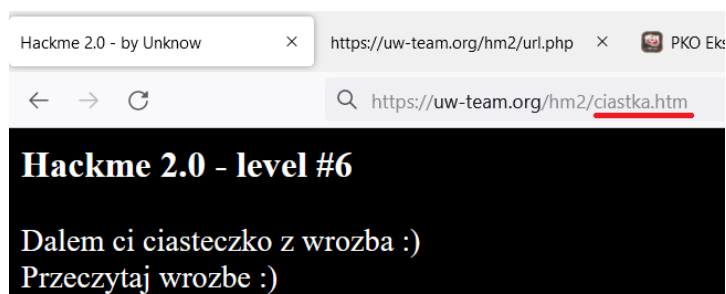
Na dole pokazany jest poziom 6. Przeczytałem wskazówkę i postanowiłem znaleźć rozwiązanie poprzez zbadanie cookies (ciasteczek).



Zbadałem dane, gdzie moją uwagę przykuły następujące podkreślone informacje.



Dlatego postanowiłem tak jak w poprzednim zadaniu wpisać inny link dodając `ciastka.htm` do obecnego linku.

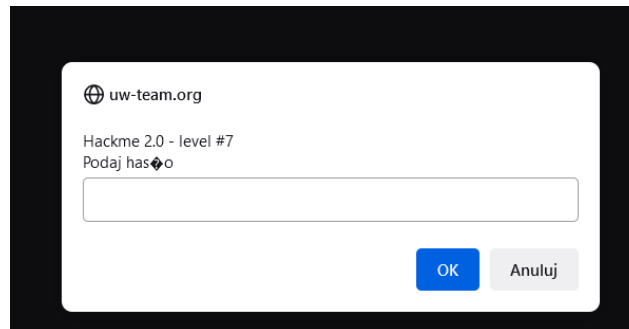


Udało się znaleźć rozwiązanie 6 zadania.

wynik = wpisanie <https://uw-team.org/hm2/ciastka.htm> i otworenie

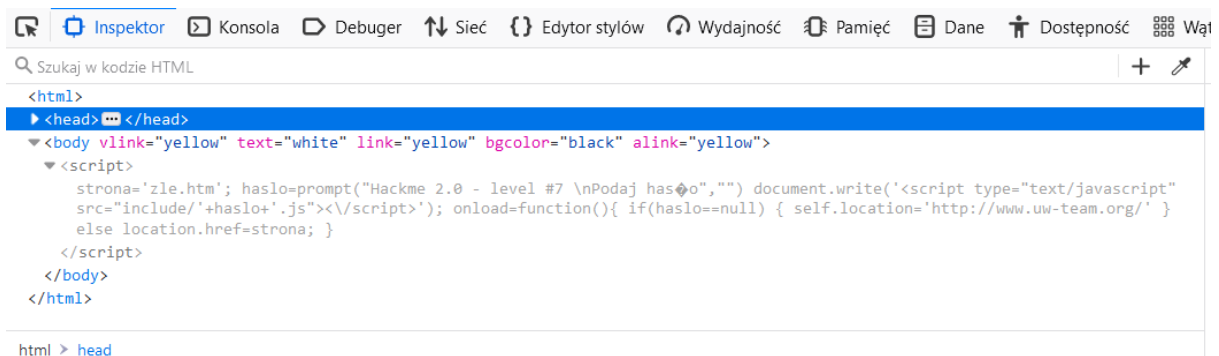
Poziom 7

Po rozwiązaniu poprzedniego zadania od razu zostałem przekierowany do zadania 7. Tak wyglądało zadanie 7.



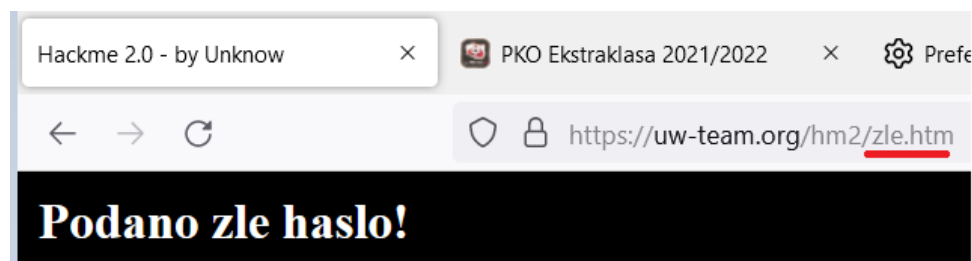
Zacząłem badać kod, jednak nie mogłem. Dlatego ponownie wyłączyłem javascript. Po wykonaniu tego zadania mogłem zacząć badanie.

Moją uwagę zwrócił poniższy skrypt:



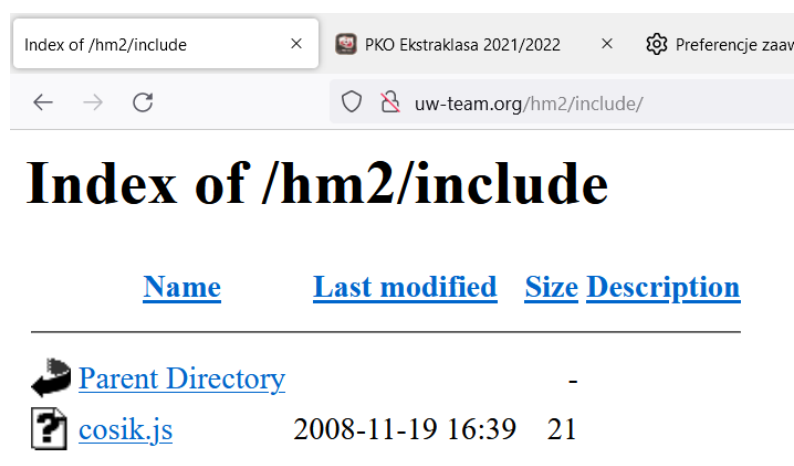
```
strona='zle.htm';
haslo=prompt("Hackme 2.0 - level #7 \nPodaj hasło", "")
document.write('<script type="text/javascript"
src="include/'+haslo+'.js"></script>');
onload=function(){
if(haslo==null) { self.location='http://www.uw-team.org/' }
else location.href=strona; }
```

Dopisałem zle.htm zamiast ciastka.htm.



Dzięki temu zdobyłem następującą informację:

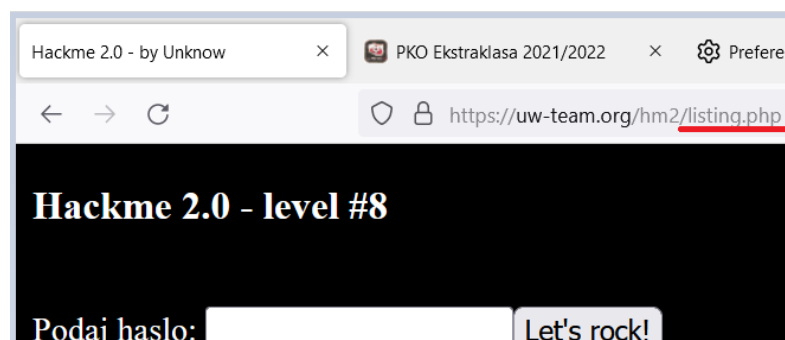
Następnie zacząłem szukać podatności serwera Apache. Postanowiłem wykorzystać do tego katalog `include`, który został znaleziony we wcześniejszym skrypcie. Poniżej widać zawartość katalogu.



Następnie otworzyłem `cosik.js` i zobaczyłem jego zawartość.

```
strona='listing.php';
```

Wpisałem `listing.php` i przeszedłem do 8 poziomu.



Wynik = wpisanie <https://uw-team.org/hm2/listing.php> i otworzenie

Poziom 8

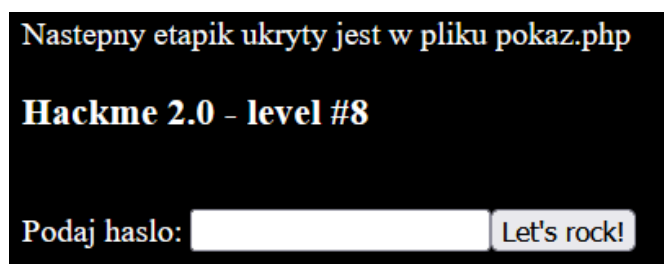
W poziomie 8 trzeba było wyłączyć javascript. Poniżej widać kod. Zbadałem poniższy kod i odczytałem wiadomość w czerwonym prostokącie tzn. hasło.

```
<div id="ukryte" style="display:none">
<font color="black">h</font><font color="black">a</font><font color="black">s</font><font color="black">l</font><font
color="black">e</font><font color="black">m</font><font color="black"> d</font>
<font color="black">o</font><font color="black"> t</font><font color="black">e</font><font color="black">g</font><font color="black">o
</font><font color="black">e</font><font color="black">t</font><font color="black">a</font><font color="black">p</font><font
color="black">u</font><font color="black"> j</font>
<font color="black">e</font><font color="black">s</font><font color="black">t</font><font color="black"> s</font><font
color="black">l</font><font color="black">o</font><font color="black">w</font><font color="black">o </font><font color="black">k</font><font
color="black">x</font><font color="black">n</font><font color="black">x</font><font color="black">g</font><font color="black">x</font><font
color="black">n</font><font color="black">x</font><font color="black">a</font></div>
</body></html>
```

Wiadomość była następująca:

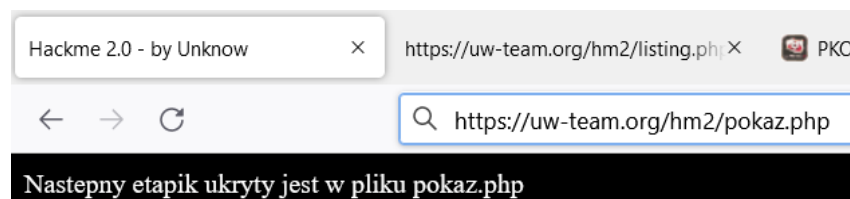
„hasłem do tego etapu jest słowo kxnxgxnx”

Hasło = “kxnxgxnx”

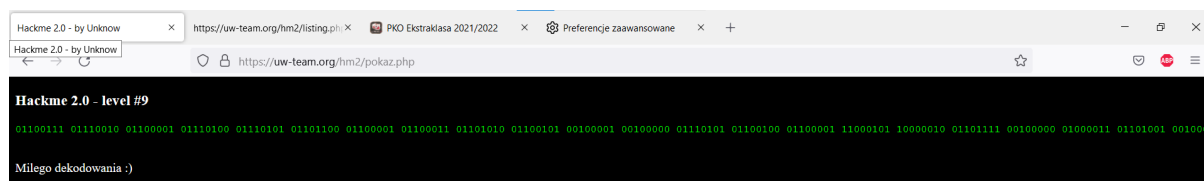


Poziom 9

Przed rozpoczęciem poziomu 9 zgodnie ze zdobytą wcześniej wskazówką dopisałem pokaz.php, abym mógł zacząć ostatni poziom.



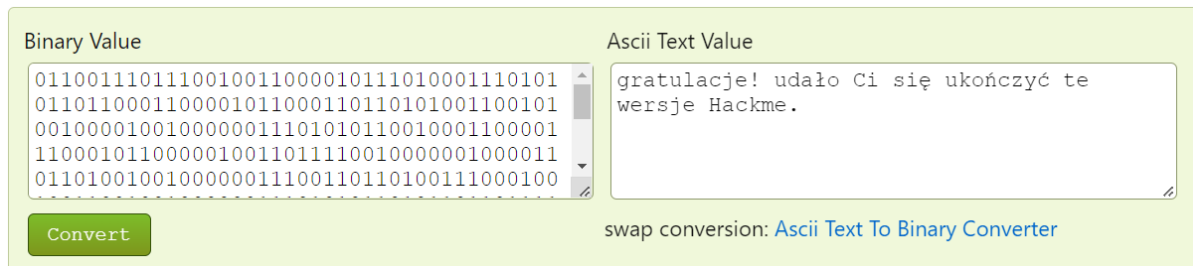
Zobaczyłem następujący wygląd ostatniego zadania.



Poniżej widać następujący kod binarny, który zamieniłem w kalkulatorze.

```
01100111 01110010 01100001 01110100 01110101 01101100 01100001 01100011
01101010 01100101 00100001 00100000 01110101 01100100 01100001 11000101
10000010 01101111 00100000 01000011 01101001 00100000 01110011 01101001
11000100 10011001 00100000 01110101 01101011 01101111 11000101 10000100
01100011 01111010 01111001 11000100 10000111 00100000 01110100 01100101
00100000 01110111 01100101 01110010 01110011 01101010 01100101 00100000
01001000 01100001 01100011 01101011 01101101 01100101 00101110
```

Otrzymałem następującą informację:



Binary Value

Ascii Text Value

gratulacje! udało Ci się ukończyć tę wersję Hackme.

Convert

swap conversion: [Ascii Text To Binary Converter](#)

Wynik = “gratulacje! udało Ci się ukończyć tę wersję Hackme.”

Tak zakończyłem ostatnie zadanie.

Koniec Gry 2

Podsumowanie

W niektórych zadaniach hasła były łatwo widoczne w kodzie strony lub umieszczone w skryptach. Taki problem rozwiązuje się poprzez ukrycie tego lub zaciemnienie. Nie może być widoczne. Również duża część zadań polegała na matematycznej wiedzy. Wystarczyło tylko rozwiązać zadania matematyczne. Oprócz tego potrzebna była wiedza programistyczna, ponieważ trzeba było przeanalizować pętle, aby zdobyć hasło. Także trzeba było wykonać operację na znakach. Do zadań wykorzystywano również kalkulatory lub konwertery na znaki ASCII. W niektórych poziomach trzeba było w pewnym momencie wyłączyć javascript. Hasła zdobywano również poprzez analizę PHP. Oprócz tego hasło było ukryte w ciasteczkach.

Wnioski

- Nie pokazywanie haseł w kodzie strony.
- Nie umieszczanie haseł w ciasteczkach.
- Nie zostawianie łatwego dostępu do katalogów strony, gdzie mogą być umieszczone poufne informacje.
- Trzeba uważać na podatności Apache
- Nie umożliwianie uzyskania hasła poprzez wysłanie odpowiedniego żądania przez atakującego i otrzymania zwrotu GET.
- Nie zostawianie haseł w skryptach.