

KIERUNEK: CYBERBEZPIECZEŃSTWO

Bezpieczeństwo aplikacji webowych Projekt

Raport 1: Zadanie rozgrzewkowe

Autor:
Łukasz Borowiecki

PROWADZĄCY PRACĘ:
Mgr inż. Przemysław Świercz

Spis treści

Wstęp.....	3
1. Cel ćwiczenia	3
2. Pierwsza gra programistyczna.....	3
2.1. Rozwiązanie pierwszego poziomu	4
2.2. Rozwiązanie drugiego poziomu	5
2.3. Rozwiązanie trzeciego poziomu	5
2.4. Rozwiązanie czwartego poziomu.....	9
2.5. Rozwiązanie piątego poziomu	10
2.6. Rozwiązanie szóstego poziomu	12
2.7. Rozwiązanie siódmego poziomu	14
2.8. Rozwiązanie ósmego poziomu	16
3. Druga gra programistyczna	18
3.1. Rozwiązanie pierwszego poziomu	19
3.2. Rozwiązanie drugiego poziomu	20
3.3. Rozwiązanie trzeciego poziomu	20
3.4. Rozwiązanie czwartego poziomu.....	21
3.5. Rozwiązanie piątego poziomu	22
3.6. Rozwiązanie szóstego poziomu	24
3.7. Rozwiązanie siódmego poziomu	25
3.8. Rozwiązanie ósmego poziomu	26
3.9. Rozwiązanie dziewiątego poziomu.....	29
Spis ilustracji	31

Wstęp

Gry programistyczne uwzględnione w tym raporcie zostały stworzone przez UW-TEAM i można je znaleźć na ich oficjalnej stronie <https://uw-team.org/>. Należy zaznaczyć, że opisane zostały tylko dwie pierwsze gry z serii HACKME.

W przygotowanym dokumencie znajdują się rozwiązania poziomów wraz z przedstawieniem toku rozumowania, który doprowadził do znalezienia poprawnego hasła.

1. Cel ćwiczenia

Głównym celem ćwiczenia było przejście wszystkich poziomów w dwóch grach programistycznych oraz sporządzenia raportu z przejścia obu gier. Do rozwiązania każdego z poziomów należy podać odpowiednie hasło. Dozwolone było korzystanie z kodu źródłowego, kalkulatora, kartki oraz ołówka.

2. Pierwsza gra programistyczna

Pierwsza gra programistyczna znajdowała się pod adresem: <https://uw-team.org/hackme/>

Hackme

Masz przed sobą prostą grę programistyczną. Polega ona na łamaniu kolejnych następujących po sobie haseł. Nie zgaduj haseł, zerkaj do kodu! Bądź uczciwy i nie zmieniaj kodu hackme lokalnie! wolno ci używać tylko kalkulatora, ołówka i twojej głowy!

[\[level 1\]](#)

Rysunek 2.1. Strona główna pierwszej gry HACKME

Aby rozwiązać każdy z poziomów, po przejściu na konkretny poziom był sprawdzany kod źródłowy strony. Żeby go wyświetlić należało skorzystać ze skrótu klawiszowego **CTRL + U** lub kliknąć na stronę **prawym przyciskiem myszy**, a następnie z listy rozwijanej wybrać **Wyświetl źródło strony**. Inną opcją było wybranie opcji **„zbadaj”** po kliknięciu **prawym przyciskiem myszy** na stronę lub przy pomocy skrótu klawiszowego **CTRL + SHIFT + I**. Używając tej opcji najbardziej będą nas interesować zakładki **Elements** oraz **Sources**.

2.1. Rozwiązanie pierwszego poziomu

Kod pierwszego poziomu zawierał gotowe rozwiązanie, które wystarczyło wpisać jako hasło i przejść do kolejnego poziomu. Po zatwierdzeniu hasła przez użytkownika przyciskiem „ok” wywoływana jest funkcja „sprawdz()”, w której wartość hasła porównywana jest do „a jednak umiem czytać”.

```
1 <HTML>
2 <script>
3 function sprawdz(){
4   if (document.getElementById('haslo').value=='i am too lame') {self.location.href="zaq.htm";} else {alert('Zle haselko :')};
5 }
6 </script>
7 <br>Level #1
8 <h3>Wpisz haslo dostępu:</h3>
9 <br><input type="text" name="haslo" id="haslo">
10 <br><input type="button" value="OK" onClick="sprawdz()">
11 </HTML>
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
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
```

Rysunek 2.1.1. Kod źródłowy pierwszego poziomu gry HACKME 1.0

Level #1

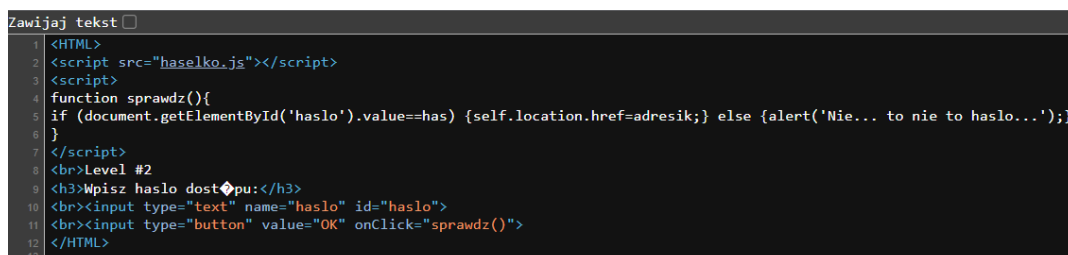
Wpisz hasło dostępu:

Rysunek 2.1.2. Rozwiązanie pierwszego poziomu gry HACKME 1.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: *a jednak umiem czytać*.

2.2. Rozwiązanie drugiego poziomu

Poprawną wartość do wpisania znajdziemy w podanym pliku „haselko.js”, które należy otworzyć. Wartość zmiennej „has” to poprawne hasło.



```
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 hasło dostępu:</h3>
10 <br><input type="text" name="haslo" id="haslo">
11 <br><input type="button" value="OK" onClick="sprawdz()">
12 </HTML>
```

Rysunek 2.2.1. Kod źródłowy drugiego poziomu gry HACKME 1.0

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

Rysunek 2.2.2. Dodatkowa zawartość dodana do drugiego poziomu gry HACKME 1.0

Level #2

Wpisz hasło dostępu:

OK

Rysunek 2.2.3. Rozwiązanie drugiego poziomu gry HACKME 1.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: **to było za proste**.

2.3. Rozwiązanie trzeciego poziomu

Funkcja „sprawdz()” jest skonstruowana w dokładnie taki sam sposób jak poprzednio. Wartością która jest porównywana jest zmienna „ost”. Jest ona również wartością wykorzystywaną w adresie strony dla kolejnego poziomu gry.

```

Zawijaj tekst
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 działa...');
10    return false;
11  }
12  return true;
13 }
14 var dod='unknow';
15 function prawy(txx){
16   var txt=txx;
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)+'que'+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 hasło dostępu:</h3>
35 <br><input type="text" name="haslo" id="haslo">
36 <br><input type="button" value="OK" onClick="sprawdz()">
37 </HTML>
38

```

Rysunek 2.3.1. Kod źródłowy trzeciego poziomu gry HACKME 1.0

Najważniejszą częścią tego poziomu jest rozszyfrowanie jaka wartość kryje się pod zmienną „ost”. Zaczniemy od tego, że początkowo zmienna ta jest pusta. Jej wartość jest zmieniona przez funkcję losuj, która jest wywołana na samym początku funkcji „sprawdz()”.

```

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

```

Zmienna dod					
0	1	2	3	4	5
u	n	k	n	o	w

Zmienna literki							
0	1	2	3	4	5	6	7
a	b	c	d	e	f	g	h

Funkcja „substring” dla zmiennej „literki”, gdzie miejscem startowym jest znak o indeksie 2, a miejscem końcowym znak o indeksie 4. Wartość jaka zostanie zwrócona przez tą funkcję jest ciąg znaków „cd”.

ost = 'cd' + 'que' + dod.substring(3,6);

Ten sam schemat należy przeprowadzić ze zmienną „dod”. W tym przypadku pozycja startowa ustawiona jest na znak o indeksie 3, a znakiem końcowym jest znak o indeksie 6. Zatem wartość jaka zostanie zwrócona przez tą funkcję jest ciąg znaków „now”.

ost = 'cd' + 'que' + 'now';

Level #3

Wpisz hasło dostępu:

Rysunek 2.3.2. Rozwiązanie trzeciego poziomu gry HACKME 1.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: *cdqwenow*.

2.4. Rozwiązanie czwartego poziomu

Jeśli wartość pola tekstowego gdzie użytkownik podaje hasło jest równe wartości przypisanej do zmiennej „wynik” to możemy przejść do kolejnego poziomu gry. Oczywiście jeśli wynik nie będzie się zgadzał użytkownik dostanie powiadomienie o tym, że podał złe hasło.

```
Zawijaj tekst ☐
1 <HTML>
2 <script>
3 function sprawdz(){
4   zaq=document.getElementById('haslo').value;
5   if (isNaN(zaq)) {alert('Zle haslo!')} else {
6     wynik=(Math.round(6%2)*(258456/2))+(300/4)*2/3+121;
7     if (zaq==wynik) {self.location.href='go'+wynik+'.htm';} else {alert('Zle haslo!')}
8   }
9 }
10 </script>
11 <br>Level #4
12 <h3>I co by tu teraz zrobic?</h3>
13 <br><input type="text" name="haslo" id="haslo" size=20>&nbsp;<input type="button" value="?" onClick="sprawdz()">
14 </HTML>
```

Rysunek 2.4.1 Kod źródłowy czwartego poziomu gry HACKME 1.0

$$\text{wynik} = 0 + (300/4) * 2/3 + 121 = 0 + 75 * 2/3 + 121 = 0 + 50 + 121 = 171$$

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: **171**.

Level #4

I co by tu teraz zrobic?

Rysunek 2.4.2. Rozwiązanie czwartego poziomu gry HACKME 1.0

2.5. Rozwiązanie piątego poziomu

Na tym poziomie mamy zabawę z czasem.

```
Zawijaj tekst ☐
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="[wejdz]" onClick="sprawdz()">
23 <script>czas();</script>
24 </HTML>
25
```

Rysunek 2.5.1 Kod źródłowy piątego poziomu gry HACKME 1.0

Funkcja `sprawdz()` zaczyna się od policzenia wartości dla zmiennej „ile”. Następnie sprawdzana jest instrukcja warunkowa, gdzie wartość zmiennej „ile” przyrównywana jest do wartości **861**. Jeśli warunek jest spełniony użytkownik zostaje przekierowany na następny poziom.

$$ile = (seconds * (seconds - 1)) / 2 * pomoc \% 2$$

$$861 = (seconds * (seconds - 1)) / 2$$

$$1722 = seconds * (seconds - 1)$$


```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout<<"HACKEME Poziom 5\n\n";
8     cout<<"Rozwiązanie: ";
9     for (int i=30; i<60; i++)
10     {
11         int result = (i*(i-1))/2;
12         if(result == 861) cout<<i;
13     }
14 }
15
```

HACKEME Poziom 5
Rozwiązanie: 42
...Program finished with exit code 0
Press ENTER to exit console.

Rysunek 2.5.2. Pomocniczy skrypt napisany w C++

Level #5

Zamek czasowy

42

Cyfra pomocnicza:

Rysunek 2.5.3 Rozwiązanie piątego poziomu gry HACKME 1.0

Zatem poprawną cyfrą pomocniczą jaką należało wpisać aby przejść na kolejny poziom była **liczba nieparzysta** (w moim przypadku była to cyfra 1) i wcisnąć przycisk **[wejdz]** kiedy licznik wskazywał **42** sekundę.

2.6. Rozwiązanie szóstego poziomu

Zaraz po zdefiniowaniu zmiennych, mamy fragment kodu z pętlą for. Wartość parametru *i* jest zwiększana o 2 oczka, co oznacza że *pętla wykona się 3 razy* (dla wartości 1,3 oraz 5).

Wartości zmiennych zdefiniowanych przed pętlą for			
<i>lit</i> = 'abcdqepolsrc'	<i>lit</i> = 'abcdqepolsrc'	<i>lit</i> = 'abcdqepolsrc'	<i>lit</i> = 'abcdqepolsrc'
<i>licznik</i> = 0	<i>licznik</i> = 0	<i>licznik</i> = 1	<i>licznik</i> = 2
<i>hsx</i> = ' '	<i>hsx</i> = ' '	<i>hsx</i> = 'bx'	<i>hsx</i> = 'bxd_'
<i>znak</i> = ' '	<i>znak</i> = ' '	<i>znak</i> = 'x'	<i>znak</i> = '_'
Wykonanie operacji w pętli for			
<i>for</i> (<i>i</i> =1 ; <i>i</i> <=5 ; <i>i</i> +=2)	dla <i>i</i> = 1	dla <i>i</i> = 1	dla <i>i</i> = 1
<i>licznik</i> ++;	<i>licznik</i> = 1;	<i>licznik</i> = 1;	<i>licznik</i> = 1;
<i>if</i> (<i>licznik</i> % 2 == 0)	(1%2 = 1)	(2%2 = 0)	(3%2 = 1)
<i>znak</i> = '_ ';	--> false	--> true	--> false
<i>znak</i> = 'x ';	<i>znak</i> = 'x ';	<i>znak</i> = '_ ';	<i>znak</i> = 'x ';
<i>hsx</i> += <i>lit</i> .substring(<i>i</i> , <i>i</i> +1) + <i>znak</i> ;	<i>lit</i> .substring(1,2) + <i>x</i> <i>hsx</i> += 'b' + 'x'	<i>lit</i> .substring(3,4) + _ <i>hsx</i> += 'd' + '_'	<i>lit</i> .substring(5,6) + <i>x</i> <i>hsx</i> += 'e' + 'x'

Zawijaj tekst ☐

```

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

```

Rysunek 2.6.1 Kod źródłowy szóstego poziomu gry HACKME 1.

Oznacza to, że funkcja *length* wywołana dla zmiennej *hsx* zwróci wartość 6. Jest to ilość znaków występująca w *string* przypisanym do zmiennej *hsx*, już po wykonaniu pętli *for*.

Zmienna <i>hsx</i> po pętli <i>for</i>					
0	1	2	3	4	5
b	x	d	_	e	x

Skoro już znamy wartość zwróconą przez funkcję *length*, możemy rozważyć ostatnią operację na zmiennej *hsx*. Do obecnej wartości zostanie dodany ciąg znaków ze zmiennej *hsx*, gdzie miejscem startowym jest 3 pozycja (*length - 3*), a pozycją końcową będzie pozycja 6. Oznacza to, że zostanie dodany ciąg trzech ostatnich znaków zmiennej *hsx*.

Level #6

Wprowadz haslo:

Rysunek 2.6.2 Rozwiązanie szóstego poziomu gry HACKME 1.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: *bx_d_ex_ex*.

2.7. Rozwiązanie siódmego poziomu

Ostatnią liniijką funkcji *sprawdz()* jest instrukcja warunkowa, z której dowiadujemy się że wartość zmiennej *wyn* musi być równa „*plxszn_xrv*”.

Zawijaj tekst ☐

```
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
```

Rysunek 2.7.1 Kod źródłowy siódmego poziomu gry HACKME 1.0

Ponieważ zmienna *wyn* musi mieć wartość „*plxszn_xrv*”, aby przejść na kolejny poziom gry, a jest ona tworzona przez dodanie wartości *ly*, która jest tworzona na zasadzie zmiany konkretnej wartości na inną zdefiniowaną w bloku z instrukcjami warunkowymi. Zatem wystarczy odczytać te wartości w odwrotnej kolejności, co pozwoli nam wpisać odpowiednie hasło.

Odszyfrowanie wartości zmiennej wyn		
$ly = p$	$lf (lx == 'k') (ly = 'p')$	$lx = k$
$ly = l$	$lf (lx == 'o') (ly = 'l')$	$lx = o$
$ly = x$	$lf (lx == 'c') (ly = 'x')$	$lx = c$
$ly = s$	$lf (lx == 'h') (ly = 's')$	$lx = h$
$ly = z$	$lf (lx == 'a') (ly = 'z')$	$lx = a$
$ly = n$	$lf (lx == 'm') (ly = 'n')$	$lx = m$
$ly = _$	$lf (lx == ' ') (ly = ' _ ')$	$lx = ' '$
$ly = x$	$lf (lx == 'c') (ly = 'x')$	$lx = c$
$ly = r$	$lf (lx == 'i') (ly = 'r')$	$lx = i$
$ly = v$	$lf (lx == 'e') (ly = 'v')$	$lx = e$

Level #7

Wprowadz hasło:

Rysunek 2.7.2. Rozwiązanie siódmego poziomu gry HACKME 1.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było:
kocham cie.

2.8. Rozwiązanie ósmego poziomu

```
Zawijaj tekst ☐
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
```

Rysunek 2.8.1 Kod źródłowy ósmego poziomu gry HACKME 1.0

Zanim przejdziemy do kolejnej linijki kodu warto wspomnieć, że do kodu załączony jest w sposób jawny plik o nazwie passwd.js. Sam plik nie zawiera nic ciekawego, co zostało pokazane na *rysunku 2.8.2*.

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

Rysunek 2.8.2. Dodatkowa zawartość dołączona do ósmego poziomu gry HACKME 1.0

W czwartej linijce kodu mamy załączony w nietypowy sposób dodatkowy plik. Dużo lepiej widać to kiedy patrzymy na kod zamieszczony w zakładce zbadaj element.

```
<script src="%7A%73%65%64%63%78%2E%6A%73"></script>
```

W samym pliku mamy zawarte zmienne, które będą nam potrzebne do znalezienia poprawnego hasła i ukończenia gry.

```

<html>
  <head>
    <script>...</script>
    ... <script src="%7A%73%65%64%63%78%2E%6A%73"></script> == $0
    <link id="chromealerabat-link" rel="stylesheet" type="text/css" href="chrome-extension://dacdinoicboceafielngnmjjplncljhj/content.css">
  </head>
  <body>
    <br>
    "Level #8 "
    <h3>Wprowadz hasło:</h3>
    <script src="%70%61%73%73%77%64.js"></script>
    <br>
    <input type="text" name="haslo" id="haslo">
    <br>
    <input type="button" value="OK" onclick="sprawdz()">

```

Rysunek 2.8.3. Kod wyświetlony po zbadaniu elementu na stronie

```

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

```

Rysunek 2.8.4. Ukryta zawartość dodana do ósmego poziomu gry HACKME 1.0

$$ax = (2 + 2 * 2) = 6$$

$$bx = (ax / 2) = 6 / 2 = 3$$

$$cx = (ax + bx) = 6 + 3 = 9$$

Ponieważ warunkiem ograniczającym pętlę jest wartość mniejsza lub równa dziesięć, **pętla wykona się dokładnie sześć razy**. Po wykonaniu skończeniu się pętli *for*, wartość zmiennej **wyn** jest równa ciągowi znaków w postaci „**grupjf**”.

Przykład: eval („ x=1; y=10; document.write(x*y) ”); WYNIK: 10

Przykład: consloe.log(eval (new String (‘ 2 + 2 ’))); WYNIK: 2 + 2

Wykonanie operacji w pętli for			
Zmienna alf = ‘qwertyuioplkjhgfdsazxcvbnm’			
	wyn += alf.charAt (qet + i)	wyn = ‘ ‘	qet = 0
dla i = 0	wyn += alf.charAt (0)	wyn = ‘ q ’	qet = 1
dla i = 2	wyn += alf.charAt (3)	wyn = ‘ qr ’	qet = 2
dla i = 4	wyn += alf.charAt (6)	wyn = ‘ qru ’	qet = 3

<code>dla i = 6</code>	<code>wyn += alf.charAt (9)</code>	<code>wyn = ' grup '</code>	<code>qet = 4</code>
<code>dla i = 8</code>	<code>wyn += alf.charAt (12)</code>	<code>wyn = ' grupj '</code>	<code>qet = 5</code>
<code>dla i = 10</code>	<code>wyn += alf.charAt (15)</code>	<code>wyn = ' grupjf '</code>	<code>qet = 6</code>

Level #8

Wprowadz hasło:

Rysunek 2.8.5. Rozwiązanie ósmego poziomu gry HACKME 1.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: **grupjf162**. Po zatwierdzeniu hasła jesteśmy przekierowani na stronę o adresie: <https://uw-team.org/hackme/grupjf162.htm>, gdzie dostajemy informację o ukończeniu gry.

You win!

Gratulacje!
Właśnie przeszedłeś grę Hackme 1.0 by Unknow!
Gratuluje cierpliwości :)

Rysunek 2.8.6. Komunikat powiadamiający o ukończeniu gry HACKME 1.0

3. Druga gra programistyczna

Kolejna odsłona gry już nie skupia się tak mocno na samych aspektach programowania oraz dobrego rozumienia prostych linijek kodu.

Hackme 2.0

Witaj w drugiej odsłonie mojej gry o nazwie "Hackme".
Gra ma na celu sprawdzenie twoich umiejętności programistycznych,
jak i umiejętności zwykłego 'kombinowania'.
Gre przechodzisz wpisując odpowiednie hasła do każdego levelu.
W tej grze jest wielka zmiana: wszystkie chwytaki dozwolone!
[masz ochotę to oszukać - o ile dasz radę... hehehe...]

Uwaga:
Nie przechodź gry metodą bruteforce, nie zgaduj haseł, zerkaj do kodu i rusz głową!
Proszę o nie przysyłanie mi na maila głupich pytań w stylu 'Podaj mi hasło do pierwszego levelu!'

...: Unknow :..

[Rozpocznij gre](#)

Rysunek 3.1. Strona główna drugiej gry HACKME

3.1. Rozwiązanie pierwszego poziomu

```
<html>
  <head>...</head>
  <body text="white" bgcolor="black" link="yellow" vlink="yellow" alink="yellow"> == $0
    <script>
      function spr(){
        if (document.getElementById('formularz').value==document.getElementById('haslo').value){
          self.location.href=document.getElementById('haslo').value+'.htm'; } else {alert('Nie, to
          nie to haselko :(');}
        }
      }
    </script>
    <h3>Hackme 2.0 - level #1</h3>
    " 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!">
  </body>
</html>
```

Rysunek 3.1.1. Kod źródłowy pierwszego poziomu gry HACKME 2.0

Jak możemy sprawdzić obiekt o nazwie „*formularz*” i takim samym id ma wartość „*text*” i jest typu ukrytego.

Hackme 2.0 - level #1

Podaj haselko:

Rysunek 3.1.2. Rozwiązanie pierwszego poziomu gry HACKME 2.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: *text*.

3.2. Rozwiązanie drugiego poziomu

W kolejnym poziomie wartość podaną jako argument funkcji **unescape** musimy **zamienić z systemu szesnastkowego na odpowiadającemu mu znakowi ASCII**.

```
<html>
  <head>...</head>
  <body text="white" bgcolor="black" link="yellow" vlink="yellow" alink="yellow"> == $0
    <script>
      function spr(){
        if (document.getElementById('haslo').value==unescape('%62%61%6E%61%6C%6E%65')) {
          self.location=document.getElementById('haslo').value+'.htm'; } else { alert('Zle
          haslo!'); }
        }
      }
    </script>
    <h3>Hackme 2.0 - level #2</h3>
    " Podaj haslo: "
    <input type="password" name="haslo" id="haslo">
    <input type="button" onclick="spr()" value="Break me!">
  </body>
</html>
```

Rysunek 3.2.1. Kod źródłowy pierwszego poziomu gry HACKME 2.0

Poszczególne znaki są między sobą oddzielone przez znak „%”. Zamiana znaków została zaprezentowana w tabeli poniżej.

Zmienna HEX na ASCII							
HEX	62	61	6E	61	6C	6E	65
ASCII	b	a	n	a	l	n	e

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: **banalne**.

3.3. Rozwiązanie trzeciego poziomu

Na tym poziomie **wartość hasła została zapisana w systemie binarnym**, co oznacza to, że aby uzyskać hasło dostępu musimy zmienić podaną wartość z systemu binarnego na dziesiętny.

```

<html>
  <head>...</head>
  <body text="white" bgcolor="black" link="yellow" vlink="yellow" alink="yellow"> == $0
    <script>
      function binary(liczba) {
        return liczba.toString(2);
      }
      function spr(){
        if (binary(parseInt(document.getElementById('haslo').value))!=10011010010) {
          self.location=document.getElementById('haslo').value+'.htm'; } else { alert('Zle!
            \nPodstawy matematyki sie klaniaja :)');}
        }
      }
    </script>
    <h3>Hackme 2.0 - level #3</h3>
    " Podaj haselko: "
    <input type="password" name="haslo" id="haslo">
    <input type="button" onclick="spr()" value="Click me baby!">
  
```

Rysunek 3.3.1. Kod źródłowy trzeciego poziomu gry HACKME 2.0

Zmienna systemu 2 na 10											
potęga	10	9	8	7	6	5	4	3	2	1	0
liczba	1	0	0	1	1	0	1	0	0	1	0
$1024 + 0 + 0 + 128 + 64 + 0 + 16 + 0 + 0 + 2 + 0 = 1234$											

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: **1234**.

3.4. Rozwiązanie czwartego poziomu

Kolejny poziom był niestandardowy. Użytkownik musiał podać hasło, nie jak do tej pory na stworzonej stronie, tylko w okienku z wyświetlonym przez stronę komunikatem.

Rysunek 3.4.1 Czwarty poziom gry HACKME 2.0

Aby wyświetlić kod należało skorzystać ze skrótu klawiszowego **CTRL + U**. Z linii kodu poniżej możemy wyczytać, że użytkownik musi podać tę wartość po przeliczeniu na system szesnastkowy.

```
Zawijaj tekst ☐
1 <!-- FIGE Z MAKIEM DOSTANIESZ, A NIE HASLO! //-->
2
3
4
5
6
7
8
9
```

Rysunek 3.4.2. Kod źródłowy czwartego poziomu gry HACKME 2.0 cz.1

```
123
124
125 <html><head><title>Hackme 2.0 - by Unknow</title></head><body text="white" bgcolor="black" link="yellow" vlink="yellow" alink="yellow">
126 <script>
127 haslo='';
128 cos=parseInt(unescape('%32%35%38'));
129 while ((haslo!=cos.toString(16)) && (haslo!='X')){
130 haslo=prompt('podaj haslo:\nwpisz X aby zatrzymac skrypt','');
131 }
132 if (haslo==cos.toString(16)) { self.location=haslo+'.php'; } else {self.location='http://www.uw-team.org/';}
133 </script>
134 <h3>Hackme 2.0 - level #4</h3>
135 <a href="hehehe.htm">Kliknij mnie :)</a>
136 </body></html>
```

Rysunek 3.4.3. Kod źródłowy czwartego poziomu gry HACKME 2.0 cz.2

Zmienna HEX na ASCII			
HEX	32	35	38
ASCII	2	5	8

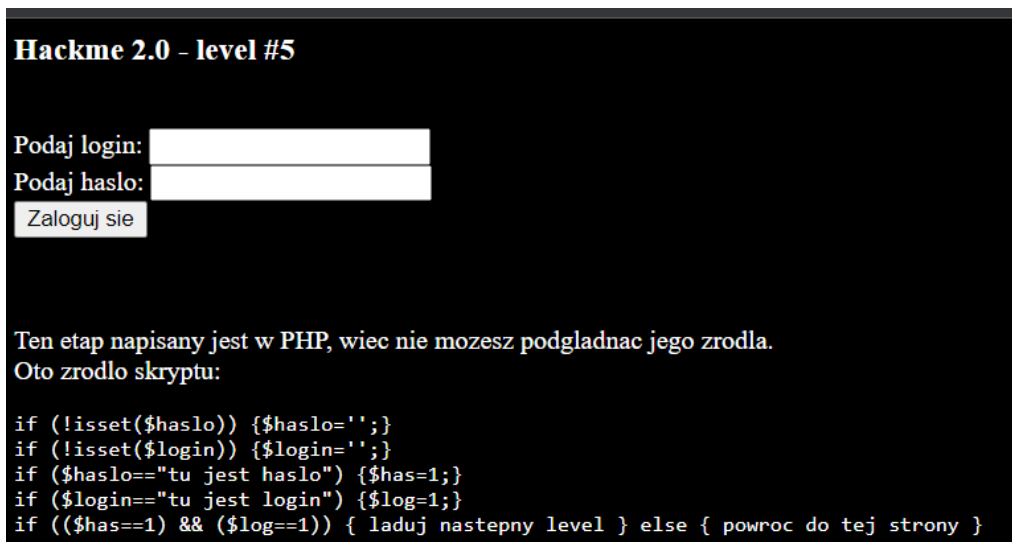
Zamiana liczby 258 na podany system została przedstawiona w tabeli poniżej.

Zmienna systemu 10 na 16 liczby 256			
potęga	2	1	0
liczba	1	0	2
256 + 0 + 2 = 258			

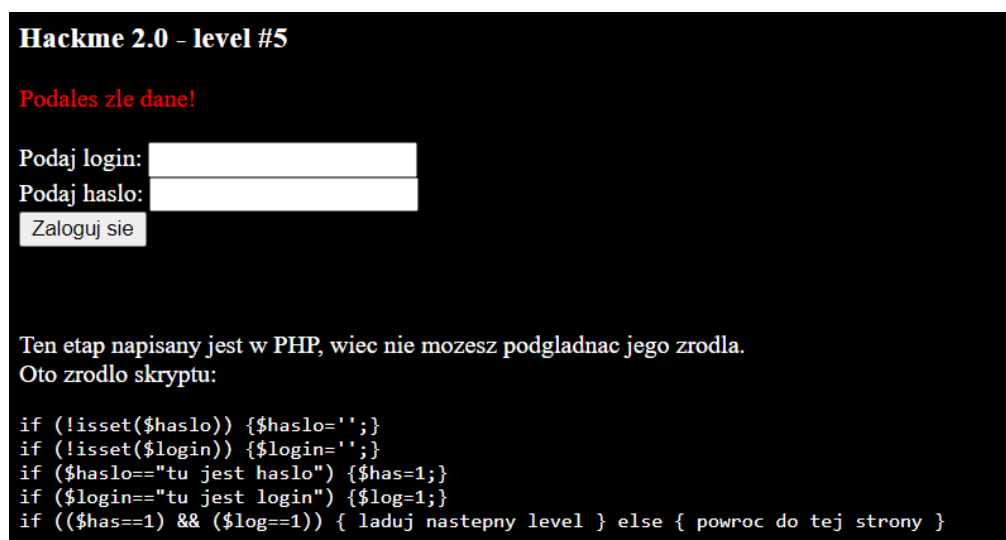
Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: **102**.

3.5. Rozwiązanie piątego poziomu

Podczas rozwiązywania tego poziomu nie mamy dostępu do kodu źródłowego, ponieważ skrypt napisano w PHP. Autor udostępnił nam część kodu, jako wskazówkę do rozwiązywania poziomu..

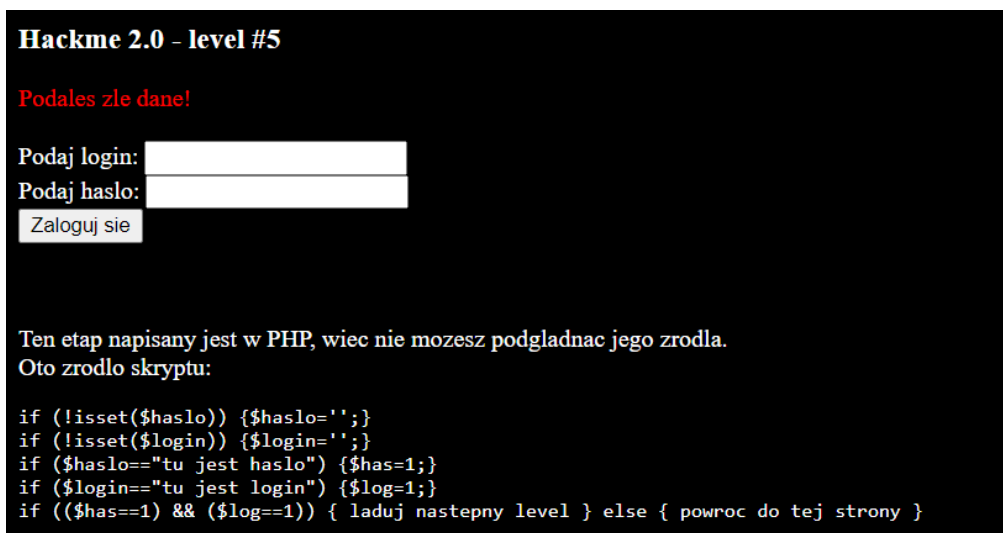


Rysunek 3.5.1. Piąty poziom gry HACKME 2.0



Rysunek 3.5.2 Błędne hasło i login – piąty poziom gry HACKME 2.0

Podpowiedź kryje się w ostatniej linijce udostępnionego kodu. Do pól na login i hasło została przeze mnie wpisana wartość jeden. Po kliknięciu przycisku zaloguj, dostaliśmy komunikat, że zestaw jest niepoprawny. Warto jednak zwrócić uwagę, że zmienił się adres strony na którym się znajdowaliśmy.



Rysunek 3.5.3 Zmiana linku – rozwiązanie piątego poziomu gry HACKME 2.0



Rysunek 3.5.4 Ukończenie piątego poziomu gry HACKME 2.0.

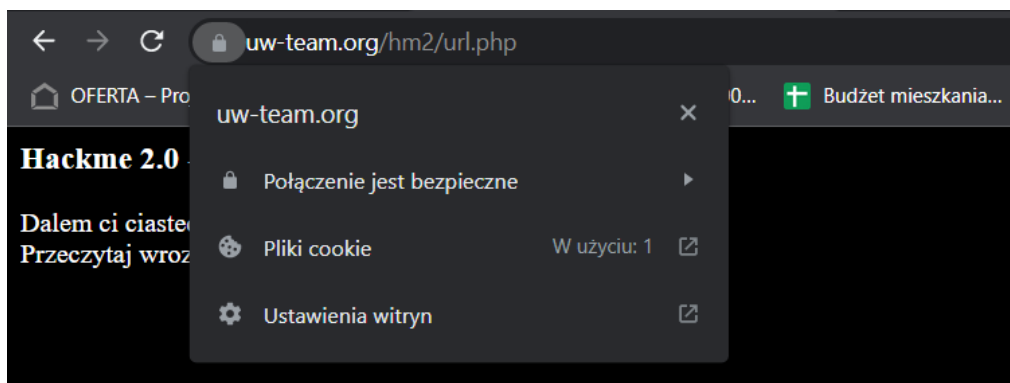
W tym momencie możemy skorzystać z podpowiedzi w ostatniej linijce i zmienić adres z linku: <http://www.uw-team.org/hm2/102.php?login=1&haslo=1> na link: <http://www.uw-team.org/hm2/102.php?log=1&has=1>. Co pozwoliło ukończyć piąty poziom.

3.6. Rozwiązanie szóstego poziomu

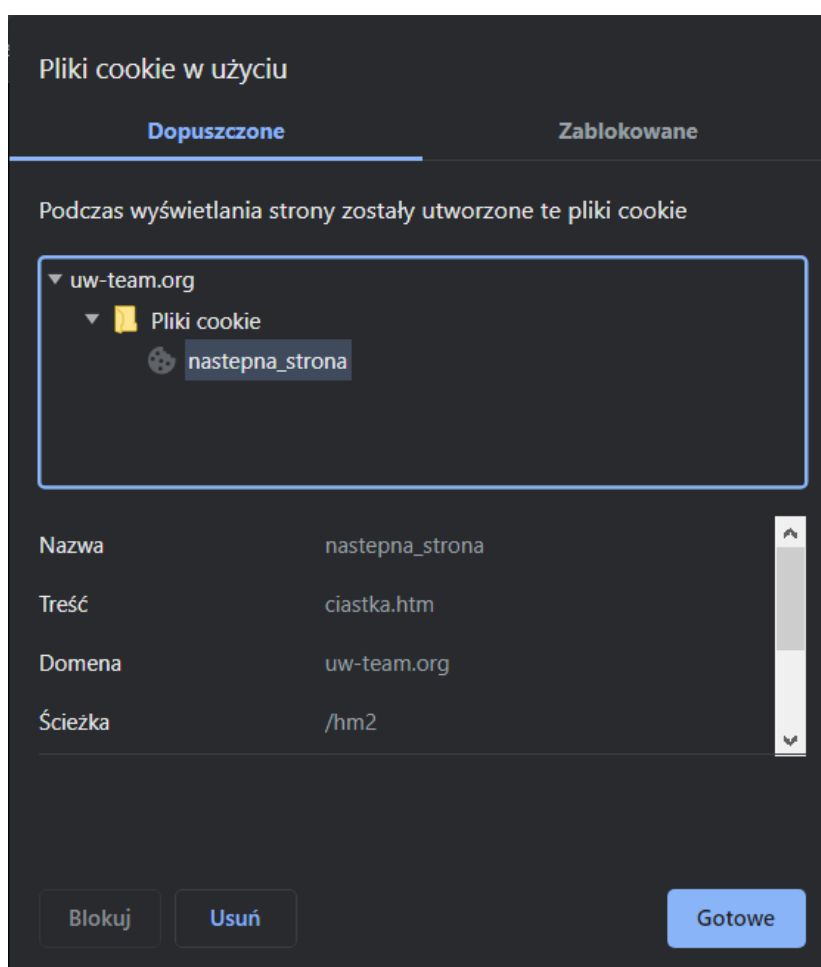
Po wyświetleniu jego zawartości już dostajemy podpowiedź. Ponieważ powiedzenie „Dałem ci ciasteczko z wróżbą” odnosi się właśnie do ciasteczek czyli *plików cookie*. Po zjrzeniu co się tam znajduje, widzimy plik o nazwie: *nastepna_strona* o treści „*ciastka.htm*”. To właśnie jest naszym rozwiązaniem. Należy zmienić adres strony na <http://www.uw-team.org/hm2/ciastka.htm>



Rysunek 3.6.1 Szósty poziom gry HACKME 2.0



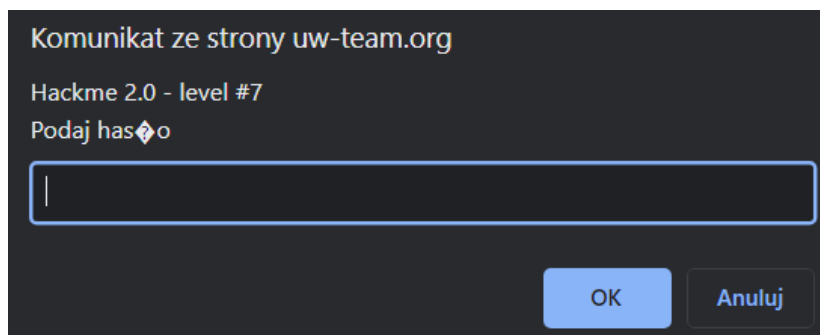
Rysunek 3.6.2 Ciasteczka dołączone do szóstego poziomu gry HACKME 2.0 cz. 1



Rysunek 3.6.3 Ciasteczka dołączone do szóstego poziomu gry HACKME 2.0 cz. 2

3.7. Rozwiązanie siódmego poziomu

W celu znalezienia hasła ponownie korzystamy ze skrótu klawiszowego **CTRL + U**.



Rysunek 3.7.1. Poziom siódmy gry HACKME 2.0

W kodzie źródłowym w piątej linijce kodu mamy wyraźne wskazanie hasła. Już po wejściu do `/include` ukazuje nam się plik **cosik.js**, co oznacza że jako hasło należy wpisać: **cosik**.



```

Zawijaj tekst ☐
1 <html><head><title>Hackme 2.0 - by Unknow</title></head><body text="white" bgcolor="black"
2 <script>
3 strona='zle.htm';
4 haslo=prompt("Hackme 2.0 - level #7 \nPodaj hasło", "")
5 document.write('<script type="text/javascript" src="include/'+haslo+'.js"></script>');
6 onload=function(){
7 if(haslo==null) { self.location='http://www.uw-team.org/' } else location.href=strona; }
8 </script>
9 </body></html>

```

Rysunek 3.7.2. Kod siódmego poziomu gry HACKME 2.0

Index of /hm2/include

Name	Last modified	Size	Description
 Parent Directory		-	
 cosik.js	2008-11-19 16:39	21	

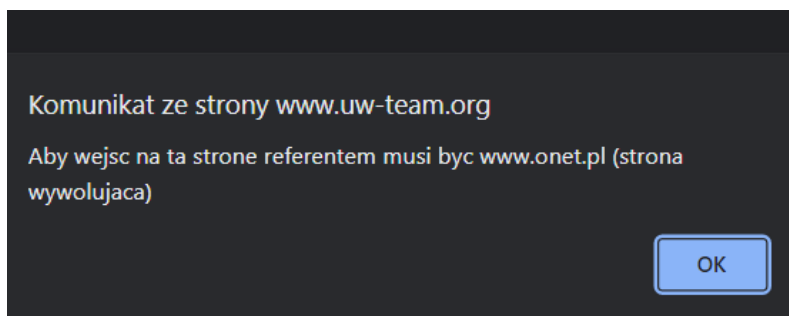
Rysunek 3.7.3 Zawartość katalogu `/include`

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

Rysunek 3.7.4 Zawartość pliku `cosik.js`

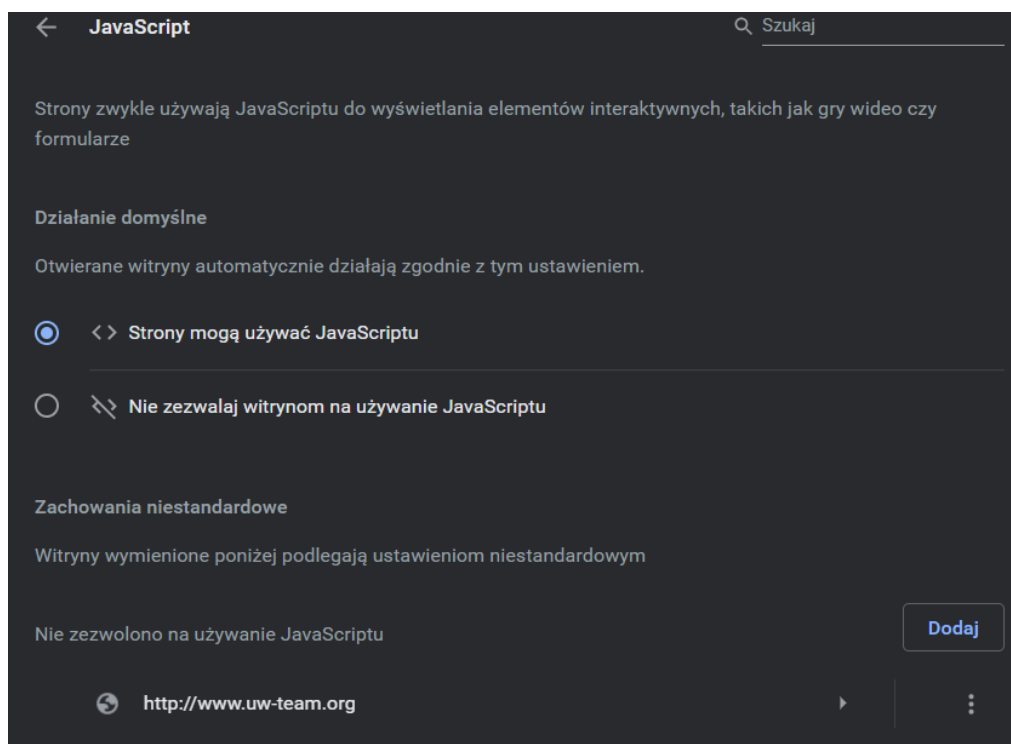
3.8. Rozwiązanie ósmego poziomu

Ten poziom podobnie jak poprzedni jest w postaci wyświetlonego komunikatu. Niestety nie mamy nigdzie okienka do podania hasła. Tak samo jak poprzednio możemy użyć skrótu klawiszowego do wyświetlenia kodu źródłowego.

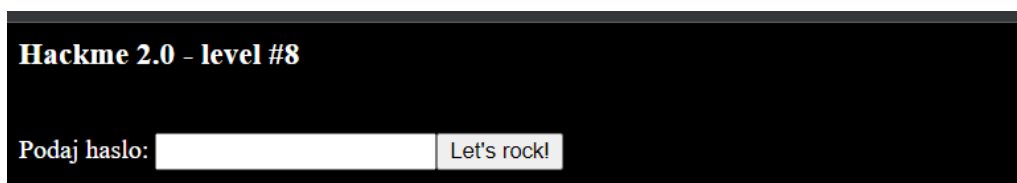


Rysunek 3.8.1. Poziom ósmy gry HACKME 2.0

Jednak żeby mieć gdzie wpisać hasło skorzystać można z drugiej opcji czyli wyłączyć działanie Java Script.



Rysunek 3.8.2. Ustawienia – brak pozwolenia stronie z grą na używanie JavaScriptu



Rysunek 3.8.3 Wyświetlenie poziomego ósmego gry HACKME 2.0

Po wyświetleniu kodu źródłowego, ukazuje nam się blok kodu z formatowaniem tekstu. Odczytując tekst z góry na dół, dostajemy komunikat: „*haslem do tego etapu jest słowo kxnxgxnx*”.

```
<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> == $0
  <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>
```

Rysunek 3.8.4 Kod poziomu ósmego gry HACKME 2.0

Zatem poprawnym hasłem jakie należało wpisać, aby przejść na kolejny poziom było: *kxnxgxnx*.

Następny etapik ukryty jest w pliku pokaz.php

Hackme 2.0 - level #8

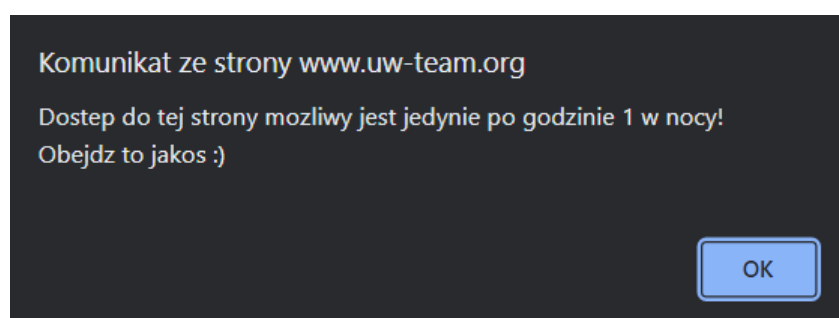
Podaj hasło:

3.9. Rozwiązanie dziewiątego poziomu

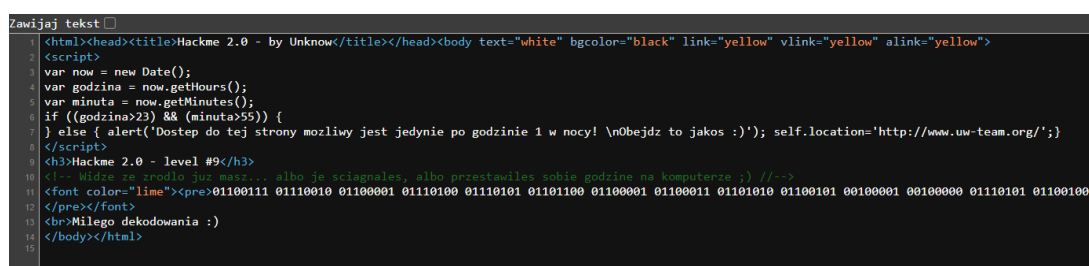
Po przejściu na wskazany wcześniej adres strony dostajemy komunikat, że dostęp do strony jest możliwy tylko po pierwszej w nocy. Są dwa możliwe scenariusze aby rozwiązać ten scenariusz.

Pierwszym jest wyświetlenie kodu źródłowego. Po wyświetleniu kodu i przeanalizowaniu działania skryptu, możemy po prostu **zmienić godzinę w komputerze** i wtedy wejść na stronę.

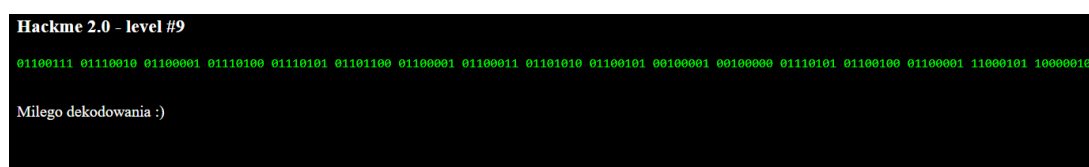
Drugą opcją było **wyłączenie możliwości uruchomienia JavaScriptu**. Powoduje to, że przechodząc na tą stronę mamy bezpośrednio wyświetloną jej zawartość.



Rysunek 3.9.1 Poziom dziewiąty gry HACKME 2.0



Rysunek 3.9.2 Kod źródłowy strony dla dziewiątego poziomu HACKME 2.0



Rysunek 3.9.3 Zawartość dziewiątego etapu gry HACKME 2.0

Po zobaczeniu treści kolejnego zadania, najprostszy sposobem będzie skorzystanie z programu online, który w szybki sposób pomoże nam odszyfrować wiadomość.

Binary to Text Translator

Enter binary numbers with any prefix / postfix / delimiter and press the *Convert* button
(E.g: 01000101 01111000 01100001 01101101 01110000 01101100 01100101):

Open File

Open Binary File

Paste binary numbers or drop file:

```
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
```

Character encoding (optional)

ASCII/UTF-8

Convert

Reset

Swap

gratulacje! udało Ci się ukończyć tę wersję Hackme.

Rysunek 3.9.4. Odszyfrowanie ciągu binarnego

Spis ilustracji

Rysunek 2.1. Strona główna pierwszej gry HACKME.....	3
Rysunek 2.1.1. Kod źródłowy pierwszego poziomu gry HACKME 1.0.....	4
Rysunek 2.1.2. Rozwiązanie pierwszego poziomu gry HACKME 1.0.....	4
Rysunek 2.2.1. Kod źródłowy drugiego poziomu gry HACKME 1.0.....	5
Rysunek 2.2.2. Dodatkowa zawartość dodana do drugiego poziomu gry HACKME 1.0.....	5
Rysunek 2.2.3. Rozwiązanie drugiego poziomu gry HACKME 1.0.....	5
Rysunek 2.3.1. Kod źródłowy trzeciego poziomu gry HACKME 1.0.....	6
Rysunek 2.3.2. Rozwiązanie trzeciego poziomu gry HACKME 1.0.....	9
Rysunek 2.4.1 Kod źródłowy czwartego poziomu gry HACKME 1.0.....	9
Rysunek 2.4.2. Rozwiązanie czwartego poziomu gry HACKME 1.0.....	10
Rysunek 2.5.1 Kod źródłowy piątego poziomu gry HACKME 1.0.....	10
Rysunek 2.5.2. Pomocniczy skrypt napisany w C++.....	11
Rysunek 2.5.3 Rozwiązanie piątego poziomu gry HACKME 1.0.....	11
Rysunek 2.6.1 Kod źródłowy szóstego poziomu gry HACKME 1.....	13
Rysunek 2.6.2 Rozwiązanie szóstego poziomu gry HACKME 1.0.....	13
Rysunek 2.7.1 Kod źródłowy siódmego poziomu gry HACKME 1.0.....	14
Rysunek 2.7.2. Rozwiązanie siódmego poziomu gry HACKME 1.0.....	15
Rysunek 2.8.1 Kod źródłowy ósmego poziomu gry HACKME 1.0.....	16
Rysunek 2.8.2. Dodatkowa zawartość dołączona do ósmego poziomu gry HACKME 1.0.....	16
Rysunek 2.8.3. Kod wyświetlony po zbadaniu elementu na stronie.....	17
Rysunek 2.8.4. Ukryta zawartość dodana do ósmego poziomu gry HACKME 1.0.....	17
Rysunek 2.8.5. Rozwiązanie ósmego poziomu gry HACKME 1.0.....	18
Rysunek 2.8.6. Komunikat powiadamiający o ukończeniu gry HACKME 1.0.....	18
Rysunek 3.1. Strona główna drugiej gry HACKME.....	19
Rysunek 3.1.1. Kod źródłowy pierwszego poziomu gry HACKME 2.0.....	19
Rysunek 3.1.2. Rozwiązanie pierwszego poziomu gry HACKME 2.0.....	19
Rysunek 3.2.1. Kod źródłowy pierwszego poziomu gry HACKME 2.0.....	20
Rysunek 3.3.1. Kod źródłowy trzeciego poziomu gry HACKME 2.0.....	21
Rysunek 3.4.1 Czwarty poziom gry HACKME 2.0.....	21
Rysunek 3.4.2. Kod źródłowy czwartego poziomu gry HACKME 2.0 cz.1.....	22
Rysunek 3.4.3. Kod źródłowy czwartego poziomu gry HACKME 2.0 cz.2.....	22
Rysunek 3.5.1. Piąty poziom gry HACKME 2.0.....	23
Rysunek 3.5.2 Błędne hasło i login – piąty poziom gry HACKME 2.0.....	23
Rysunek 3.5.3 Zmiana linku – rozwiązanie piątego poziomu gry HACKME 2.0.....	24
Rysunek 3.5.4 Ukończenie piątego poziomu gry HACKME 2.0.....	24
Rysunek 3.6.1 Szósty poziom gry HACKME 2.0.....	24
Rysunek 3.6.2 Ciasteczka dołączone do szóstego poziomu gry HACKME 2.0 cz. 1.....	25
Rysunek 3.6.3 Ciasteczka dołączone do szóstego poziomu gry HACKME 2.0 cz. 2.....	25
Rysunek 3.7.1. Poziom siódmy gry HACKME 2.0.....	26
Rysunek 3.7.2. Kod siódmego poziomu gry HACKME 2.0.....	26
Rysunek 3.7.3 Zawartość katalogu /include.....	26

<i>Rysunek 3.7.4 Zawartość pliku cosik.js.....</i>	<i>26</i>
<i>Rysunek 3.8.1. Poziom ósmy gry HACKME 2.0.....</i>	<i>27</i>
<i>Rysunek 3.8.2. Ustawienia – brak pozwolenia stronie z grą na używanie JavaScriptu</i>	<i>27</i>
<i>Rysunek 3.8.3 Wyświetlenie poziomu ósmego gry HACKME 2.0.....</i>	<i>27</i>
<i>Rysunek 3.8.4 Kod poziomu ósmego gry HACKME 2.0.....</i>	<i>28</i>
<i>Rysunek 3.8.5 Rozwiązanie poziomu ósmego gry HACKME 2.0.....</i>	<i>29</i>
<i>Rysunek 3.9.1 Poziom dziewiąty gry HACKME 2.0</i>	<i>29</i>
<i>Rysunek 3.9.2 Kod źródłowy strony dla dziewiątego poziomu HACKME 2.0.....</i>	<i>29</i>
<i>Rysunek 3.9.3 Zawartość dziewiątego etapu gry HACKME 2.0.....</i>	<i>29</i>
<i>Rysunek 3.9.4. Odszyfrowanie ciągu binarnego.....</i>	<i>30</i>