Mateusz Dudziak 20.03.2022

Zadanie rozgrzewkowe

HackMe 1

Adres

https://uw-team.org/hackme/

Level 1

Hasło można odnaleźć przeglądając kod strony. Jest ono zlokalizowane w funkcji „sprawdz” w tagu <script> w <body>. Można je rozpoznać po funkcji porównania, która weryfikuje, czy wprowadzone przez użytkownika hasło dostępu jest równoznaczne z ciągiem znaków „a jednak umiem czytac”.

```
1.  <script>
2.  function sprawdz(){
3.  if (document.getElementById('haslo').value=='a jednak umiem czytac')
    {self.location.href='ok_next.htm';} else {alert('Zle haselko :)');}
4.  }
5.  </script>
```

Po wprowadzeniu „a jednak umiem czytac” powinno być możliwe przejście na kolejny poziom.

W ramach tego poziomu w kodzie strony znajduje się również inna funkcja „sprawdz” zlokalizowana w <head>. Została ona umieszczona by zmylić użytkownika i wprowadzenie „i am too lame” nie pozwoli na przejście do kolejnego poziomu.

Level 2

Obserwując funkcję „sprawdz” znajdującą się w tagu <head> tak jak poprzednio można odnaleźć porównanie mające na celu weryfikację hasła, jednak w tym przypadku hasło nie zostało zapisane za pomocą ciągu znaków, tylko w postaci zmiennej „has”. Dalsza weryfikacja musi zatem polegać na odnalezieniu zwartości wspomnianej zmiennej. Można tego dokonać weryfikując zawartość pliku „haselko.js”, na co wskazuje wpis fragment strony „<script src="haselko.js"></script>”.

```
1.  <script src="haselko.js"></script>
2.  <script>
3.  function sprawdz(){
4.  if (document.getElementById('haslo').value==has) {self.location.href=adresik;} else
    {alert('Nie... to nie to haslo...');}
5.  }
6.  </script>
```

Zawartość pliku haselko.js znajdującego się pod adresem: „https://uw-team.org/hackme/haselko.js”:

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

Poza wartością samego hasła w pliku przechowywane jest adres, pod którym znajduje się poziom trzeci: „formaster.htm”. Do kolejnego poziomu można przejść wpisując hasło „to bylo za proste” lub bezpośrednio udając się pod adres wskazany w pliku: „https://uw-team.org/hackme/formaster.htm”.

Level 3

Tym razem hasło zostało ukryte przed użytkownikiem i jego poznanie wymaga zidentyfikowania wyniku poszczególnych funkcji. W tym celu użytkownik musi zlokalizować dwie zmienne: „dod" oraz „literki".

```
1.  var dod='unknow';
2.  var literki='abcdefgh';
```

Samo hasło jest przechowywane w zmiennej „ost", zadeklarowanej:

```
1.  var ost='';
```

Jego konstrukcja odbywa się w funkcji „losuj":

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

Pobierany jest fragment ciągu znaków „literki", konkretnie 3 i 4 znak za pomocą „substring" (ID liczone od 0, dlatego w kodzie widzimy przedział 2-4, co odpowiada znakom 2 i 3), dodawany jest ciąg „qwe" oraz fragment ciągu znaków „now", pobierany z „unknow" (4, 5 i 6 znak, licząc od zera – 3, 4 i 5), co finalnie tworzy ciąg znaków: „cdqwenow". Dodatkową trudność stanowi zaszycie deklaracji oraz funkcji pośród większego kodu". Hasło stanowi jednocześnie fragment adresu kolejnego poziomu: "https://uw-team.org/hackme/cdqwenow.htm".

Level 4

Kolejny poziom jest zdecydowanie bardziej matematyczny. I wymaga rozwiązania działania. Ponownie należy w kodzie strony odnaleźć funkcję „sprawdz".

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

Na podstawie analizy kodu można wywnioskować, że jedynie podanie w polu „haslo" wyniku działania „(Math.round(6%2)*(258456/2))+(300/4)*2/3+121" pozwoli przejść do następnego poziomu. Działanie można wykonać w bardzo prosty sposób, wykonując w konsoli przedstawione powyżej polecenie, jednak warto jest również zrozumieć, w jaki sposób zostało ono skonstruowane. Funkcja Math.round() zwraca wartość liczby zaokrągloną do najbliższej liczby całkowitej. Operator % zwraca resztę pozostałą po podzieleniu jednego operandu przez drugi, tak więc działanie „Math.round(6%2)" będzie równe 0, a więc i mnożenie go przez „(258456/2)" nie ma większego sensu, bo wynik tej części działania wyniesie 0. Właściwy wynik jest konstruowany w drugiej części działania: „(300/4)*2/3+121", która jest dodawana do wyniku pierwszej części, czyli zera. W tym przypadku pamiętając o kolejności działań wystarczy podzielić 300 przez 4 oraz pomnożyć wynik przez 2 i ponownie podzielić, tym razem przez 3. Do tak uzyskanej liczby 50 należy dodać 121, co pozwoli uzyskać wyniki 171 – hasło do kolejnego poziomu i jednocześnie lwią część adresu kolejnego poziomu tworzonego przez dołączenie ciągu znaków „go" przed i „.htm" po: „go171.htm". Do następnego poziomu można przejść po kliknięciu w przycisk ze znakiem zapytania.

Level 5

Kolejny poziom bazuje na liczniku czasowym. Oprócz wpisania odpowiedniego hasła wymagane jest wykonanie tej czynności, gdy licznik będzie wskazywał odpowiednią sekundę (możemy się o tym dowiedzieć obserwując kod pobierający aktualny czas i wyświetlający go w „txt”. Ponownie bardzo istotna jest funkcja „sprawdz”. Analizując ją możemy zidentyfikować zmienną przechowującą wartość wymaganą do przejścia do następnego poziomu: „ile”.

```
1.  function sprawdz(){
2.  ile=((seconds*(seconds-1))/2)*(document.getElementById('pomoc').value%2);
3.  if (ile==861) {self.location.href=seconds+'x.htm'} else {alert('Zle haslo!');}
4.  }
```

Zmienna “ile” przyjmuje wynik działania „((seconds*(seconds-1))/2)*(document.getElementById('pomoc').value%2)”, gdzie „pomoc” przechowa wartość wprowadzoną przez użytkownika. Na tej podstawie można utworzyć konkretne równanie: „((x*(x-1))/2)*y%2 = 861”, gdzie x oznacza sekundę, w której uruchomimy skrypt, y wartość, którą wpiszemy w polu „Cyfra pomocnicza”, a 861 to oczekiwany wynik, do którego przyrównywana jest zmienna „ile”. Tak jak w poprzednim wypadku wykorzystano tu operator modulo na wartości wprowadzanej przez użytkownika. Podanie liczby podzielnej przez 2 jednoznacznie sprawi, że cały wynik będzie równy zero, ponieważ reszta będzie wynosiła 0. Na tej podstawie można jasno określić, że wymagane jest wprowadzenie liczby, która modulo 2 da resztę 1, np. cyfrę 1. Teraz wystarczy zidentyfikować sekundę, w której wymagane będzie kliknięcie przycisku „[wejdz]”. Po wstępnej analizie można pominąć zmienną y i uznać, że będzie wynosiła 1 i rozwiązać równanie „(x*(x-1))/2 = 861”, czyli „x^2-x-1722 = 0”, przy założeniu, że x musi być większe, bądź równe 0, ponieważ przyjmuje ono wartość sekundy. Wykorzystując metodykę rozwiązywania równań kwadratowych lub narzędzie „https://www.wolframalpha.com/”, można odnaleźć dwa wyniki równania: „-41” i „42”. Wynik powyżej zera, czyli „42” pozwoli dostać się do kolejnego poziomu. Jego adres jest dodatkowo konstruowany z wyniku, do którego dodawana jest fraza „x.htm”.

Level 6

Poziom 6 opiera się o wprowadzenie hasła konstruowanego w obrębie tagu „<script>”. Do utworzenia hasła zadeklarowano szereg zmiennych: „lit”, przechowująca ciąg znaków, „hsx” przechowująca wynikowe hasło, „znak”, przechowująca „x” lub „_”, w zależności od działania instrukcji warunkowej oraz „licznik” inkrementowany podczas działania pętli „for”. Konstrukcja rozpoczyna się od pętli „for”. Zgodnie z warunkiem, ma ona wykonać dalsze operacje dokładnie trzy razy. Iterator „i” przyjmie kolejno wartości: 1, 3 i 5. Wartość początkową ustawiono na 1, następnie wartość ta jest podnoszona o 2, do momentu, aż będzie nie większa, niż 5. Przy każdym przejściu funkcja podnosi wartość zmiennej „licznik” o 1, co pozwoli jej przyjąć kolejno wartości: 1, 2 i 3. Podzielność zmiennej „licznik” przez 2 jednoznacznie będzie wpływać na wartość zmiennej „znak”, która w wypadku, gdy zmienna „licznik” będzie podzielna przez 2 przyjmie wartość „_”, a w innym wypadku „x”. Ostatecznie zmienna „znak” przy kolejnych przejściach pętli przyjmie wartości: „x”, „_” i „x”. Na dalszym etapie do wartości zmiennej „hsx” dodawany jest fragment ciągu znaków zapisany w zmiennej „lit”, uzależniony od wartości iteratora „i” oraz wcześniej ustalona wartość zmiennej „znak”. Zawsze dodawany jest jeden znak, a jego pozycja to „i”, licząc od zera. Takie działanie jest możliwe ze względu na specyfikę funkcji „substring”, do której w ramach argumentów podawane są: pozycja pierwszego znaku oraz ogranicznik. Tym sposobem zmienna „hsx” po kolejnych przejściach pętli będzie wynosiła: „bx”, „bxd_” i „bxd_ex”. Poniżej wartości poszczególnych zmiennych podczas kolejnych przejść pętli:

i	i+1	licznik	znak	znak pobierany z lit	hsx

1	2	1	x	b	bx
3	4	2	_	d	bxd_
5	6	3	x	e	bxd_ex

Następnie ponownie przy wykorzystaniu „substring” pobierany i dołączany na końcu jest fragment ciągu znaków ze zmiennej hsx. Ograniczenia zdefiniowano poprzez odwołanie się do całkowitej długości zmiennej i pozwoliły one rozbudować hasło o 3 ostatnie znaki tak, że w ostatecznej postaci wygląda ono następująco: „bxd_ex_ex”.

```
<script>
var lit='abcdqepolsrc';
function sprawdz(){
var licznik=0;
var hsx='';
var znak='';
zaq=document.getElementById('haslo').value;
for (i=1; i<=5; i+=2){
licznik++;
if ((licznik%2)==0) {znak='_';} else {znak='x';}
hsx+=lit.substring(i,i+1)+znak;
}
hsx+=hsx.substring(hsx.length-3,hsx.length);
if (zaq==hsx) {self.location.href=hsx+'.htm';} else {alert('Zle haslo!');}
}
</script>
```

Wprowadzenie hasła pozwoli przejść na kolejny poziom dostępny pod adresem „https://uw-team.org/hackme/bxd_ex_ex.htm”.

Level 7

Poziom siódmy oparty został o szyfr podstawieniowy. Tablica podstawień została zdefiniowana za pomocą prostych instrukcji warunkowych:

```
if (lx=='a') {ly='z'}
if (lx=='b') {ly='y'}
if (lx=='c') {ly='x'}
if (lx=='d') {ly='w'}
if (lx=='e') {ly='v'}
if (lx=='f') {ly='u'}
if (lx=='g') {ly='t'}
if (lx=='h') {ly='s'}
if (lx=='i') {ly='r'}
if (lx=='j') {ly='q'}
if (lx=='k') {ly='p'}
if (lx=='l') {ly='o'}
if (lx=='m') {ly='n'}
if (lx=='n') {ly='m'}
if (lx=='o') {ly='l'}
if (lx=='p') {ly='k'}
if (lx=='q') {ly='j'}
if (lx=='r') {ly='i'}
if (lx=='s') {ly='h'}
if (lx=='t') {ly='g'}
if (lx=='u') {ly='f'}
if (lx=='v') {ly='e'}
if (lx=='w') {ly='d'}
if (lx=='x') {ly='c'}
if (lx=='y') {ly='b'}
if (lx=='z') {ly='a'}
if (lx==' ') {ly='_'}
```

Wartość szyfrogramu również jest znana („plxszn_xrv”) i została zapisana w innej instrukcji warunkowej porównującej ją z zaszyfrowanym ciągiem wprowadzonym przez użytkownika.

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

By wprowadzić poprawne hasło użytkownik musi za każdą z liter we wskazanym szyfrogramie podstawić odpowiednik znajdujący się w pierwszej kolumnie. To pozwoli uzyskać hasło: „kocham cie”.

Zadanie jednak można rozwiązać znacznie prościej, odnajdując adres kolejnego poziomu. Analizując konstrukcję tworzenia adresu następnego poziomu na podstawie hasła, można dojść do wniosku, że jest ono wynikiem połączenia dwóch ciągów znaków: szyfrogramu („plxszn_xrv”) i „.htm”.

Level 8

W ramach ostatniego poziomu HackMe w kodzie strony umieszczono bardzo dużo dodatkowej treści, mającej zaciemnić właściwy kod wykorzystywany do konstrukcji hasła. Należy rozpoznać deklarację ciągu znaków „alf”, zmiennej przechowującej ostateczne hasło „wyn” oraz zmiennej wskazujący na znak pobierany z „alf” – „qet”.

```
1.  wyn='';
2.  alf='qwertyuioplkjhgfdsazxcvbnm';
3.  qet=0;
```

W pierwszej kolejności wykonywana jest pętla for. Zapisane w niej instrukcje zostaną wykonane 6 razy, co jest realizowane przez iterator „i” przyjmujący kolejno wartości: 0, 2, 4, 6, 8, 10. Następnie do zmiennej „wyn” dodawany jest znak ze zmiennej „alf” znajdujący się na pozycji o wartości sumy zmiennych „qet” oraz „i” (licząc od 0). Na koniec każdej iteracji wartość zmiennej „qet” jest podnoszona o jeden.

i	qet	qet+i	wyn	qet++
0	0	0	q	1
2	1	3	qr	2
4	2	6	qru	3
6	3	9	qrup	4
8	4	12	qrupj	5
10	5	15	qrupjf	6

W tym przypadku należy szczególnie pamiętać, że wartość zmiennej „qet” jest inkrementowana dopiero po pobraniu znaku ze zmiennej „alf”.

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

Do wartości zmiennej “wyn” dodawany jest następnie wynik działania „eval(ax*bx*cx)”.

```
1.  wyn+=eval(ax*bx*cx);
```

Wartości zmiennych “ax”, “bx” i „cx” są przechowywane w odrębnym pliku, do którego adres zamieszczono w górnej części strony: „%7A%73%65%64%63%78%2E%6A%73”, co prowadzi do pliku „zsedcx.js”

```
1.  ax=eval(2+2*2);
2.  bx=eval(ax/2);
3.  cx=eval(ax+bx);
```

Na podstawie zapisanych wyżej działań można jasno określić wartości wszystkich trzech zmiennych:

```
1.  ax = 6
2.  bx = 3
3.  cx = 9
```

Do zmiennej „wyn” zostanie dołączona zatem wartość 6 * 3 * 9, czyli 162. Ostateczne hasło będzie następujące: „qrupjf162”. To pozwoli zakończyć poziom, a jednocześnie grę.

HackMe 2

Adres

https://uw-team.org/hm2

Level 1

Poziom pierwszy gry HackMe 2 posiada hasło wprowadzone w kodzie strony w tagu „<imput>” – formularz. Można go odkryć usuwając z kodu frazę „type="hidden"”, jednak nie jest to konieczne, ponieważ hasło zostało zapisane w „value”, jako „text”. Wprowadzenie tej wartości w polu do podawania hasła przeniesie użytkownika na drugi poziom.

```
1.  <input value="text" name="formularz" id="formularz" type="hidden">
```

Level 2

Poziom drugi wymaga od użytkownika znalezienia w kodzie strony funkcji „spr” oraz występującego w niej ciągu znaków „%62%61%6E%61%6C%6E%65”.

```
1.  <script>
2.  function spr(){
3.  if (document.getElementById('haslo').value==unescape('%62%61%6E%61%6C%6E%65')) {
    self.location=document.getElementById('haslo').value+'.htm'; } else { alert('Zle
    haslo!'); }
4.  }
5.  </script>
```

Po odkodowaniu ciągu znaków z formatu URL na przykład za pomocą strony „https://www.url-encode-decode.com/” poznać można hasło: „banalne”.

Level 3

Poziom trzeci opiera się o podobny schemat, co poziom drugi, jednak wymaga od użytkownika przeliczenia liczby binarnej „10011010010” znajdującej się w funkcji „spr” na liczbę w formacie dziesiętnym. Wynikiem ostatecznie jest „1234”.

```
1.  <script>
2.  function binary(liczba) {
3.  return liczba.toString(2);
4.  }
5.  function spr(){
6.  if (binary(parseInt(document.getElementById('haslo').value))==10011010010) {
    self.location=document.getElementById('haslo').value+'.htm'; } else { alert('Zle!
    \nPodstawy matematyki sie klaniaja :)');}
```

```
7.  }
8.  </script>
```

Wprowadzenie w ramach hasła ciągu znaków „1234” pozwoli na przejście do kolejnego poziomu.

Level 4

By rozwiązać poziom 4 użytkownik musi przejść w ramach narzędzi deweloperskich (dostępnych pod przyciskiem F12) do zakładki „Network” i wybrać „Preserve log”, by logi ruchu sieciowego pozostały po zmianie strony. Następnie w oknie, które pojawi się w ramach poziomu wpisać „X”, by zatrzymać skrypt. Kolejnym krokiem jest odnalezienie odpowiedzi na zapytanie GET dotyczącego strony „1234.htm”. Właściwy kod rozpoczyna się w linii 125. Kluczem do odnalezienia hasła w ramach tego poziomu jest zdekodowanie wartości zmiennej „cos”. Można to zrobić na przykład za pomocą strony „https://www.urldecoder.org/”. Wartość zmiennej „cos”, która po ostatniej operacji wynosi 258, musi jednak zostać dodatkowo zamieniona na „string” funkcją toString(). Dodatkowo wykorzystany jest parametr „radix”, który pozwala na konwersję liczby do innego systemu liczbowego, w tym przypadku o podstawie 16. Ostateczny wynik będzie zatem wynosił 102, co jednocześnie jest hasłem do następnego poziomu.

```
1.  <script>
2.  haslo='';
3.  cos=parseInt(unescape('%32%35%38'));
4.  while ((haslo!=cos.toString(16)) && (haslo!='X') ){
5.  haslo=prompt('podaj haslo:\nwpisz X aby zatrzymac skrypt','');
6.  }
7.  if (haslo==cos.toString(16)) { self.location=haslo+'.php';} else
    {self.location='http://www.uw-team.org/';}
8.  </script>
```

Level 5

Piąty poziom został oparty o prosty skrypt w języku PHP.

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

Wymogiem przejścia do następnego poziomu jest wprowadzenie poprawnych danych logowania. To pozwoli na ustawienie wartości zmiennych „has” i „log” na wartość 1, co następnie przyczyni się do odblokowania dostępu do następnego poziomu. Można to zrobić poprzez adres URL, wprowadzając ręcznie wartości dla obu zmiennych: „102.php?log=1&has=1”. Dzięki temu możliwe będzie uzyskanie adresu URL następnego etapu: „url.php”.

Level 6

Link do następnego poziomu można uzyskać analizując zakładkę „Application” w ramach narzędzi deweloperskich. Adres URL poziomu 7 znajduje się w ciasteczku „nastepna_strona” i posiada wartość „ciastka.htm”.

Level 7

W ramach poziomu siódmego użytkownik już na wstępie poproszony zostaje o podanie hasła. Analizując zakładkę „Network” w ramach narzędzi deweloperkich w prosty sposób zauważyć można, że wprowadzone hasło zostaje następnie nazwą pliku w formacie JS z folderu „/include/”. Wymogiem jest więc podanie nazwy pliku z tego właśnie folderu. By zweryfikować, jakie pliki faktycznie znajdują

się w folderze „/include/” najlepiej udać się pod adres „https://uw-team.org/hm2/include”. W folderze można znaleźć jedynie plik „cosik.js”. Słowo „cosik” jest więc poszukiwanym hasłem. Zawartość pliku:

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

Zawartość wskazuje na adres kolejnego poziomu – „listing.php”

Level 8

Poziom 8 wymusza na użytkowniku zmianę wartości nagłówka „Referer”, wskazującego na adres strony, która przekierowała użytkownika. Nie jest to jednak jedyne rozwiązanie, uzyskać dostęp do strony można blokując wykonywanie Javascript w jej obrębie. Następnie hasło można odnaleźć w kodzie strony w sposób szczególnie utrudniający odczytanie go:

```
1.  <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>
2.  <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>
3.  <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>
```

Po odczytaniu wiadomości („haslem do tego etapu jest slowo kxnxgxnxa”) jasnym staje się, że w pole “Podaj hasło”, użytkownik może wpisać: „kxnxgxnxa”, co poinformuje go o adresie kolejnego poziomu: „pokaz.php”. Hasło można również wprowadzić jako wartość zmiennej „haslo” w adresie URL: „haslo=kxnxgxnxa”.

Level 9

Dostęp do ostatniego poziomu jest możliwy jedynie po godzinie pierwszej w nocy. Wymaga to od użytkownika wejścia pod wskazany adres URL o konkretnej godzinie, przestawieniu czasu w komputerze lub zablokowaniu obsługi Javascript w obrębie strony. W wyniku użytkownik uzyskuje dostęp do strony zawierającej zakodowaną binarnie wiadomość:

```
1.  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
```

Jej odkodowanie można przeprowadzić na przykład za pomocą strony „https://www.rapidtables.com/convert/number/binary-to-ascii.html” dekodującej zapis binarny do ASCII. Ostateczny komunikat brzmi:

```
1.  gratulacje! udało Ci się ukończyć te wersje Hackme.
```