Bezp. syst. i usług inform. 2
Gra programistyczna Hackme

Adam Zimny 209787

6 listopada 2016

1 Cel projektu

Celem projektu było przejście dwóch gier polegających na rozwiązywaniu łamigłówek wykorzystujących wiedzę programistyczną. Każda z gier złożona była z 8 poziomów. W celu przejścia do kolejnego poziomu należało podać hasło, które ukryte było w kodzie źródłowym strony.

2 Realizacja

Rozwiązanie każdego poziomu należało rozpocząć od analizy źródła strony, oraz w wielu przypadkach analizy kodu skryptów na niej się znajdujących. Przy rozwiązaywaniu zadań przydatna była konsola developerska Google Chrome.

2.1 Hackme

Poziom trudności pierwszej z gier był niski. Rozwiązanie większości poziomów można było znaleźć poprzez ustawienie punktu wstrzymania w linii skryptu odpowiadającej za sprawdzenie poprawności hasła, a następnie sprawdzenie, z jaką wartością jest porównywany tekst wpisany przez użytkownika.

Poziom 1

Poprawne hasło dla tego poziomu to `a jednak umiem czytac`. Hasło było ukryte w kodzie źródłowym strony, ukryte przez liczne znaki nowej linii, tak aby znalazło się na nowej stronie

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

Poziom 2

Poprawne hasło dla tego poziomu to `to bylo za proste`. Skrypt porównuje wpisany przez użytkownika tekst z wartością zmiennej. W kodzie źródłowym strony znaleźć możemy dołączony plik JavaScript zawierający definicje zmiennych użytych przy sprawdzaniu hasła.

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

Po otwarciu tego pliku w przeglądarce otrzymamy wartości poprawnego hasła oraz adresu kolejnego poziomu.

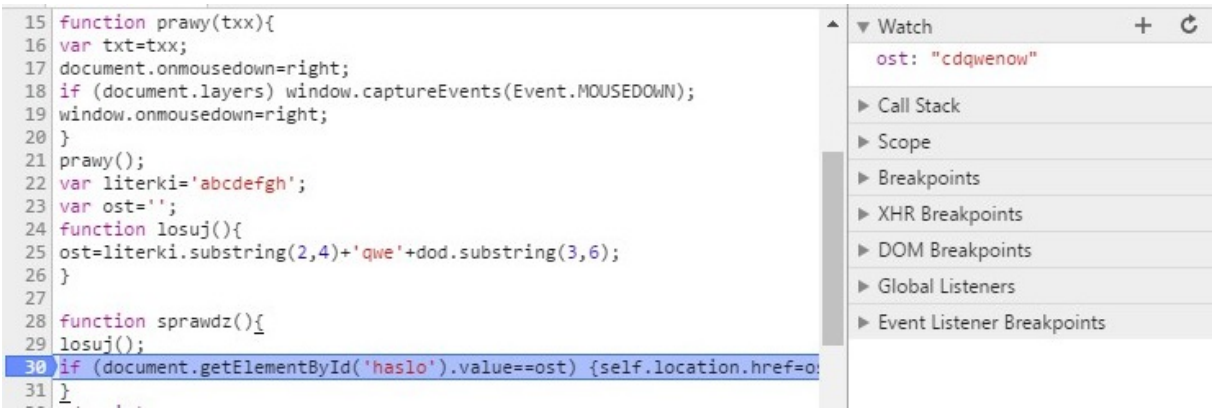
```
var has='to bylo za proste';
var adresik='formaster.htm';
```

Poziom 3

Poprawne hasło dla tego poziomu to `cdqwenow` . Skrypt tworzy hasło korzystając z wartości zmiennych `var dod='unknow';` oraz `var literki='abcdefgh';` Do utworzenia hasła wykorzystywana jest funkcja `substring()` pobierająca wskazany fragment łańcucha znaków.

```
ost=literki.substring(2,4)   // "cd"
+'qwe'
+dod.substring(3,6);     // "now"
```

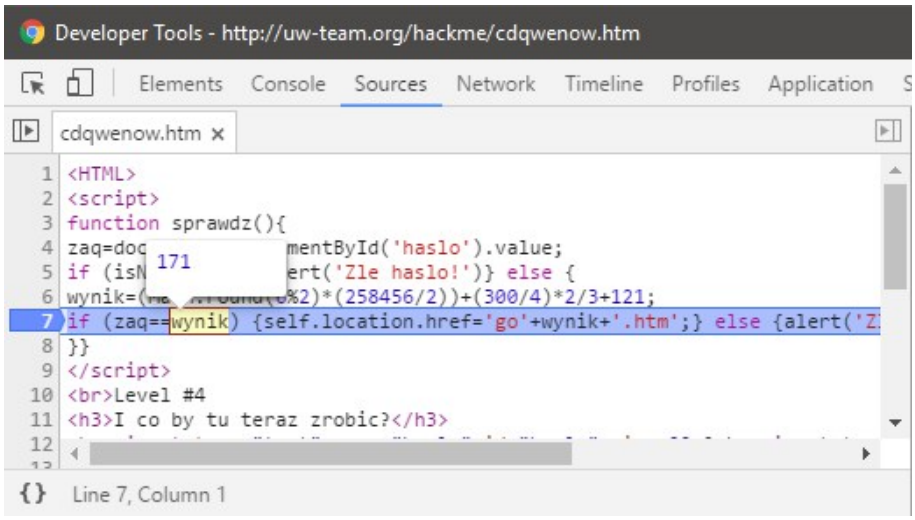
Innym sposobem rozwiązania tego poziomu jest wykorzystanie konsoli do sprawdzenia oczekiwanej wartości. Po otwarciu pliku źródłowego i ustawieniu punktu przerwania należy dodać zmienną `ost` do okna Watch i uruchomić skrypt. Po zatrzymaniu w ustawionym punkcie można odczytać hasło.



Poziom 4

Poprawne hasło dla tego poziomu to `171` . W celu jego uzyskania należy obliczyć wartość wyrażenia `wynik=(Math.round(6%2)*(258456/2))+(300/4)*2/3+121` . Wyrażenie `Math.round()` zaokrągla liczbę do najbliższej liczby całkowitej, a znak `%` oznacza resztę z dzielenia, która wynosi w tym przypadku 0. . Wyrażenie może zostać uproszczone do `0 + 50 + 121` co daje wynik 171.

Oczywiście, podobnie jak w przypadku poprzedniego poziomu, wartość może być odczytana z kosoli developerskiej:



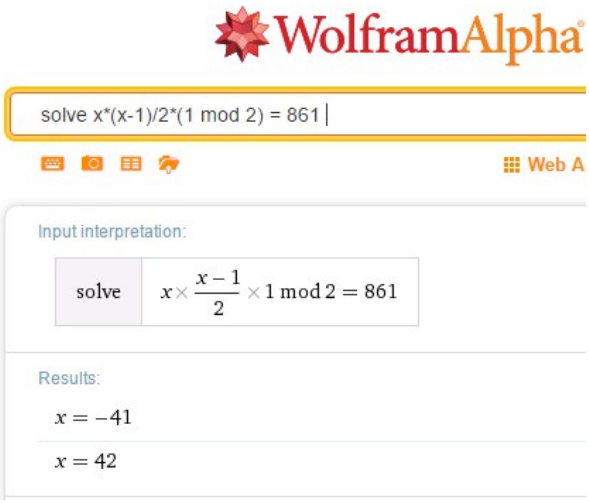
Poziom 5

Poprawne hasło dla tego poziomu to dowolna liczba nieparzysta. Hasło należy wprowadzić gdy na zegarze pojawi się wartość 42s. Funckja sprawdzająca poprawność hasła oblicza wartość na podstawie wprowadzonej przez użytkownika wartości oraz wartości zegara.

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

W celu uzyskania poprawnej wartości należy zatem rozwiązać równianie z niewiadomą:

`x*(x-1)/2*(i mod 2)= 861` gdzie `x` to wartość zegara, a `i` wartość wprowadzana przez użytkownika. Z równania widać, że wartość zmiennej nie ma znaczenia, istatna jest tylko jej parzystość. Po wprowadzeniu równania do serwisu WolframAlpha uzyskuje się następujące rozwiązania:



Poziom 6

Poprawne hasło dla tego poziomu to `bxd_ex_ex` . Wartość generowana jest przez pętlę:

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

Pętla wykonuje 3 iteracje dla `i = 1,3,5` . Wartości po zakończeniu iteracji pokazuje tabela:

iteracja	i	licznik	znak	hsx
1	1	1	x	bx
2	3	2	_	bxd_
3	5	3	x	bxd_ex_

Wartość zmiennej `hsx` może także zostać odczytana z konsoli w sposób opisany w poprzednich poziomach.

Poziom 7

Poprawne hasło dla tego poziomu to `kocham cie` . Litery w haśle należy zastąpić lustrzanie względem środka alfabetu, tzn. `a` na `z` , `b` na `y` itd. tak tak aby uzyskać ciąg znaków `plxszn_xrv` . Rozwiązanie można odczytać w ten sam sposób, zmieniając znaki w tekście zarzyfrowanym według tej samej zasady.

Poziom 8

Poprawne hasło dla tego poziomu to `qrupjf162` . Rozwiązanie tego poziomu uzyskuje się w podobny sposób do poziomu 6, analizując przebieg pętli generującej ciąg znaków.

```
alf='qwertyuioplkjhgfdsazxcvbnm';
qet=0;
for (i=0; i<=10; i+=2){
        get+=10;
        wyn+=alf.charAt(qet+i);
        qet++;
}
wyn+=eval(ax*bx*cx);
```

Wartości zmiennych `ax, bx, cx` zdefiniowane są w osobnym pliku JavaScript. Aby uzyskać do niego dostęp trzeba odkodować nazwę pliku z linii

```
document.write('<\s'+'c'+'r'+'i'+'p'+'t src="\%7A
    \%73\%65\%64\%63\%78\%2E\%6A\%73"><\/s'+'c'+'r'+'i'+'p'+'t
    '+'>');
```

. Tekst jest zakodowany jako URL, można go odczytać korzystając z narzędzia `http://coderstoolbox.net/string` . Zawartość odczytanego pliku `zsedcx.js` to:

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

2.2 Hackme 2.0

Poziom 1

Poprawne hasło dla tego poziomu to `text` . Wartość wpisana przez uzytkownika porównywana jest z wartością pola `formularz` dokumentu html. Polu został nadany atrybut `hidden` , aby nie wyswietlało się na stronie.

```
if (document.getElementById('formularz').value==document.
    getElementById('haslo').value)
...
<input value="text" name="formularz" id="formularz" type="hidden">
```

Poziom 2

Poprawne hasło dla tego poziomu to `banalne` . Ponownie tekst jest zakodowany jako URL, należy go odkodować korzystając z odpowiedniego narzędzia. W skrypcie wartość jest odczytywana przy pomocy funkcji `unescape()` .

```
if (document.getElementById('haslo').value==unescape('\%62\%61\%6E
    \%61\%6C\%6E\%65'))
```

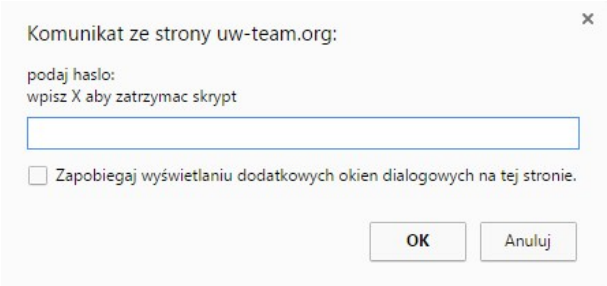
Poziom 3

Poprawne hasło dla tego poziomu to `1234` . Wartość hasła jest zapisana w postaci binarnej, należy podać jej wartość w systemie dziesiętnym. Do konwersji liczby można wykorzystać narzędzie `http://www.binaryhexconverter.com/binary-to-decimal-converter` .

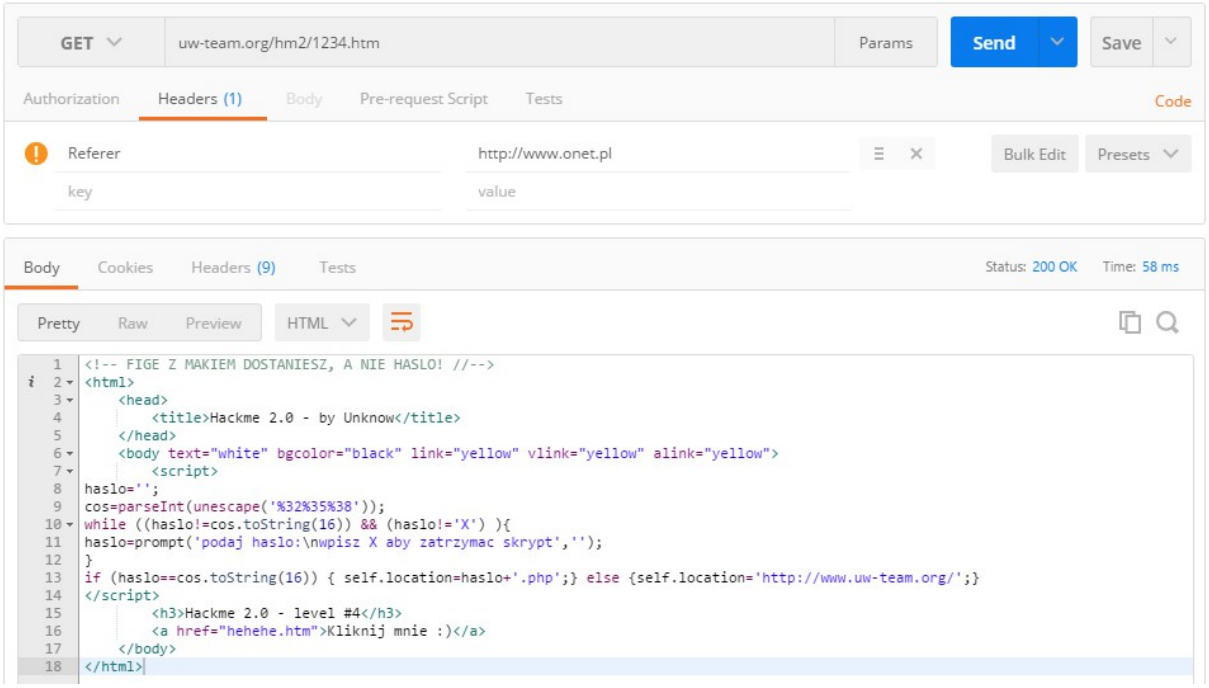
```
if (binary(parseInt(document.getElementById('haslo').value))
    ==10011010010)
```

Poziom 4

Po przejściu poprzedniego poziomu przeglądarka wyświetli okno dialogowe z polem do podania hasła:



Okno to blokuje możliwość wyświetlenia źródła strony poprzez kliknięcie pracym przyciskiem myszy. Aby pobrać źródło strony należy skorzystać z narzędzia Postman pozwalającego na wysyłanie zapytań http. W odpowiedzi na zapytanie zwracany jest kod strony:



W celu rozwiązania zadania należy odczytać wartość liczby `cos` zakodowanej jako URL, a następnie zamienić ją na postać szesnastkową. Za konwersję odpowiedzialna jest funkcja `toString(16)` .

Odczytane w ten sposób hasło to `102` .

Poziom 5

Ten etap gry napisany jest w języku PHP, więc nie możliwe jest podejrzenie jego źródła. Kod został podany w tekscie na stronie:

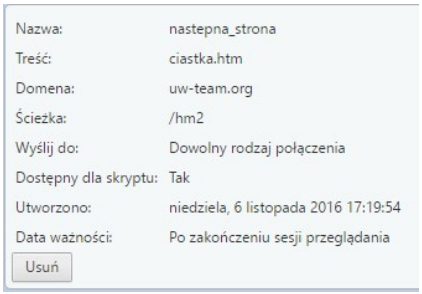
```
if (!isset($haslo)) {$haslo='';}
if (!isset($login)) {$login='';}
if ($haslo=="tu jest haslo") {$has=1;}
if ($login=="tu jest login") {$log=1;}
if (($has==1) && ($log==1)) { laduj nastepny level } else { powroc
    do tej strony }
```

W celu przejścia tego poziomu należy ustawić wartości zmiennych `log` i `has` na 1. W tym celu należy wykonać atak nazywany wstrzykiwaniem kodu. Do adresu strony należy dopisać żądane zmienne jako parametry zapytania GET w następujący sposób:

```
http://uw-team.org/hm2/102.php?has=1&log=1
```

Poziom 6

W tym etapie należy odczytać plik cookie odebrany ze strony po otwarciu. Można to zrobić z poziomu ustawień przeglądarki Google Chrome wybierając opcje `Ustawienia treści... > Wszystkie pliki cookie i dane witryn...` w sekcji **Prywatność**



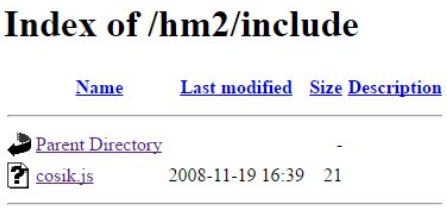
Z pliku odczytać można adres kolejnego poziomu: `ciastka.htm` .

Poziom 7 Po wejściu na tę stronę zostaje wyświetlone okno dialogowe z prośbą o hasło, podobnie jak na poziomie 5. W celu wyświetlenia strony można ponownie skorzystać z narzędzie Postman, lub dopisać odpowiedni przedrostek do adresu strony w przeglądarce Google Chrome: `view-source:http://uw-team.org/hm2/ciastka.htm` .

W otrzymanym kodzie można odnaleźć linię dołączającą kolejny plik javascript. Nazwą pliku jest szukane hasło:

```
document.write('
            <script type="text/javascript" src="include/'+haslo+'.
                js"><\/script>');
```

Należy zauważyć, że plik dołączany jest z folderu `include` . Po otwarciu w przeglądarce tego folderu można odczytać nazwę szukanego pliku:



Zawartość pliku `cosik.js` określa adres strony kolejnego poziomu: `strona='listing.php';`

Poziom 8

Po wejściu na stronę tego poziomu wyświetlone zostaje następujące okno dialogowe:

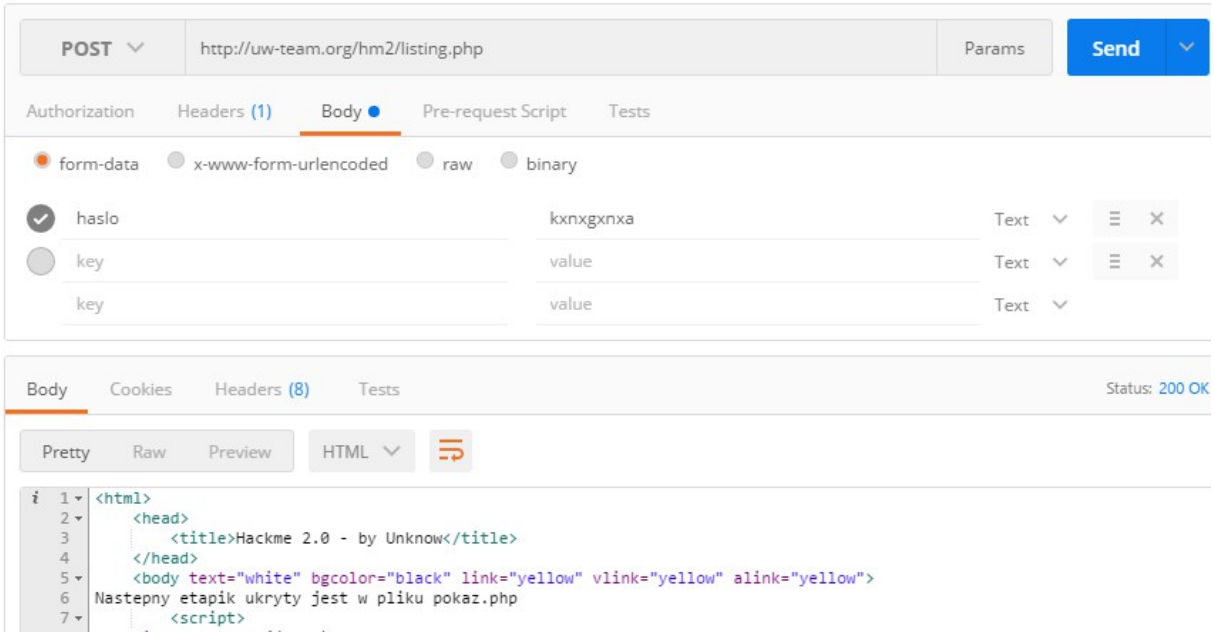
Oznacza to, że aby uzyskać dostęp do tej strony, przekierowanie na nią musi nastąpić z serwisu Onet. Informacja o przekierowaniu zawarta jest w nagłówku Referer zapytania http. W celu wejścia na stronę można skorzystać z rozszerzenia do przeglądarki pozwalającej definiować własne nagłówki.

Można także pobrać źródło strony na jeden ze sposobów opisanych przy poprzednich etapach gry. Z pobranego kodu odczytać można hasło:

```
<form action="listing.php">
            <br>Podaj haslo:
            <input type="password" name="haslo">
            <input type="submit" value="Let's rock!">

        </form>
        <br>
        <br>
        <div id="ukryte" style="display:none">
            ...
            <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>
```

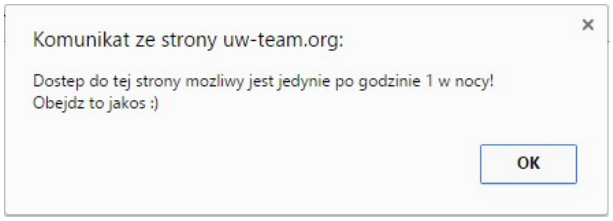
Linia `<form action="listing.php">` mówi o tym, że hasło następnie należy przesłać jako parametr zapytania na adres `listing.php` . Ponownie korzystając z Postmana nalezy stworzyć odpowienie zapytanie. Typ zapytania nie został sprecyzowany, zatem można skorzystać z zapytania POST:



Po wykonaniu zapytania ze zwróconego kodu można odczytać adres strony kolejnego poziomu gry.

Poziom 9

Po wejściu na stronę tego poziomu wyświetlone zostaje następujące okno dialogowe:



Po pobraniu kodu strony wyświetlony zostaje tekst zakodowany binarnie. Do jego odkodowania można skorzystać z narzędzia

`http://www.roubaixinteractive.com/PlayGround/Binary_Conversion/Binary_To_Text.asp`

Po odkodowaniu można odczytać tekst wiadomości:

Gratuluje :) Udalo ci sie rozkodowac ten etapik :] Nie bylo to specjalnie trude... Wystarczylo zrobic sobie program konwertujacy, lub wejsc na www.google.pl i wpisac text to binary. To byl juz ostatni etap tej gry. Aby byc wpisanym na liste zwyciezcow przeslij haslo bezkvu6r na adres unkn0w@wp.pl