Assignment: Reversing an MD5 hash (password cracking)

In this assignment we build code to reverse an MD5 hash using a brute force technique where we simply 'forward hash' all possible combinations of characters in strings. This would be similar to a situation where an e-commerce site stored hashed passwords in its database and we somehow have gotten our hands on the database contents and we want to take the hashed password and determine the actual plaintext passwords.

This following is a list of people, and their hashed PIN values. PIN value.

```
email                pin   hash_pin
-----                ---   --------
csev@umich.edu       ????  0bd65e799153554726820ca639514029
nabgilby@umich.edu   ????  aa36c88c27650af3b9868b723ae15dfc
pconway@umich.edu    ????  1ca906c1ad59db8f11643829560bab55
font@umich.edu       ????  1d8d70dddf147d2d92a634817f01b239
collemc@umich.edu    ????  acf06cdd9c744f969958e1f085554c8b
...
```

You should be able to easily crack all but one of these these PINs using your application.

The simplest brute force approach generally is done by writing a series of nested loops that go through all possible combinations of characters. This is one of the reasons that password policies specify that you include uppper case, lower case, numbers, and punctuation in passwords is to make brute force cracking more difficult. Significantly increasing the length of the password to something like 20-30 characters is a very good to make brute force cracking more difficult.

Sample solution

You can explore a sample solution for this problem at

http://www.wa4e.com/solutions/crack/

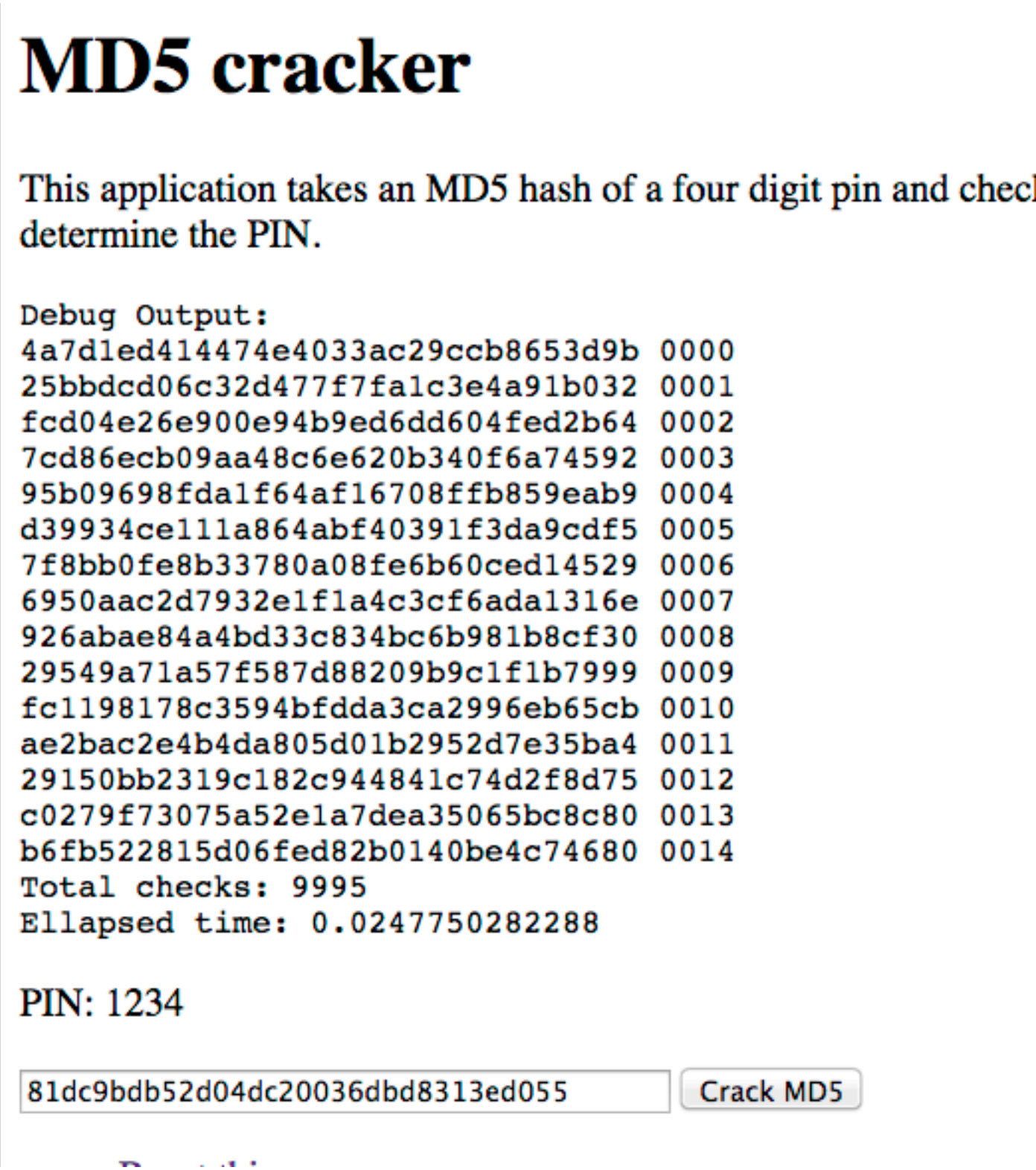
Resources

There are several sources of information so you can do the assignment:

- Chapters 14, 23-28, 31 and 32 from the free textbook The Missing Link: An Introduction to Web Development and Programming written by Michael Menendez and published by Open SUNY Textbooks.
- Lectures and materials on Expressions, Control Flow, Arrays, Functions, and Forms www.wa4e.com

- Partially working sample code. You can play with this application at http://www.wa4e.com/code/crack/ and download a ZIP of the code athttp://www.wa4e.com/code/crack.zip.

Specifications



Your application will take an MD5 value like "81dc9bdb52d04dc20036dbd8313ed055" (the MD5 for the string "1234") and check all combinations of four-digit "PIN" numbers to see if any of those PINs produce the given hash.

You will present the user with a form where they can enter an MD5 string and request that you reverse-hash the string. If you can reverse hash the string, print out the PIN:

```
PIN: 1234
```

If the string does not reverse hash to a four digit number simply put out a message like:

```
PIN: Not found
```

You must check all four-digit combinations. You must has the value as a **string** not as an integer. For this shows the right and wrong way to check the has for "1234":

```
$check = hash('md5', '1234');  // Correct - hashing a string
$check = hash('md5', 1234);    // Incorrect - hashing an integer
```

You should also print out the first 15 attempts to reverse-hash including both the MD5 value and PIN that you were testing. You should also print out the ellapsed time for your computation as shown in the sample application.

Consistency Details

In order to make the assignments more consistent, please follow these technical guidelines:

- Put all of your code to do the cracking in your "index.php" so you can hand in one file. You can have other files (like in the sample solution) that you do not have to hand in.
- Name the form field where you pass the MD5 into your application "md5"
- `<input type="text" name="md5" size="40">`
- Use the GET method on your form (i.e. not POST)

What To Hand In

For this assignment you will hand in:

1. A screen shot of one of the MD5's from the list above/in the spec that you can successfully crack. Include the URL of your page in the screen shot so we can see your GET parameter.

2. One of the MD5's list above/in the spec does not crack. Using your code, figure out which MD5 in the list does *NOT* crack and show your application not finding the PIN for the MD5. Include the URL of your page in the screen shot so we can see your GET parameter.
3. Source code of your index.php

Grading

Please review carefully. The actual points are less important and useful comments about what might be wrong and need fixing. You cannot re-submit your assignment unless the instructor allows you to resubmit.

The total number of points for this assignment is 10. You will get up to 5 points from your instructor. You will get up to 3 points from your peers. You will get 1 for each peer assignment you assess. You need to grade a minimum of 2 peer assignments. You can grade up to 5 peer assignments if you like.

Optional Challenges

This section is entirely *optional* and is here in case you want to explore a bit more deeply and test your code skillz.

Here are some possible improvements:

- For fun, crack all of the pins at the top of this document and figure out why each person chose their PIN.
- You can crack some but not all more complex hashed values using a site like: CrackStation.net. For fun, use this site to crack all the above hash values.
- Make your application test a more complex character set like, upper case letters, lower case letters, numbers, and common punctuation.
- Change the code so when it finds a match, it breaks out of all four of the nexted loops. So if the PIN turned out to be 1234 it would only run that many times. Hint: Make a logical variable that you set to true when you get a match and then as soon as that becomes true, break out of the outer loops.
- Make your program handle longer strings - say six characters. At some point when you increase the number of characters and alphabet, it will take longer to reverse crack the string.
- Change the debug output to print an attempt every 0.1 second instead of only the first 15 attempts.
- Super Advanced: Make your program handle variable length strings - perhaps looking for a string from 3-7 characters long. At some point

just making more nested loops produces too much code and you should switch to a more complex but compact approach that uses a few arrays and a while loop. But this can be tricky to construct and prone to infinite loops if you are not careful. This is probably best not attempted unless you have some background in Algorithms and Data Structures.

As your program increases its character length, or tests longer passwords, it will start to slow down. Make sure to run these on your laptop (i.e. not on a server). Many hosted PHP systems prohibit these kinds of CPU-intensive tasks on their systems.

At some point you might run into a time out where PHP decides that your code is running too long and blows up your application. You can check the variable**max_execution_time** in your PHPInfo screen to see how many seconds PHP will let your code run before aborting it.

Provided by: www.wa4e.com