ashish@home:~$

Writeup -- 2013-CRYPTO-400 Backdoor --

23 Feb 2016

This post comes at a very disturbing time - night before electromagnetism midterm exam. I am not sure why I am doing this but I feel a very strong compulsion. This is probably because I have drowned myself in books for the last 3 days without much rest. Still, preparation is far from complete. This post marks the start of CTF writeups by me. Future writeups will probably be under a new category, but then there is no category system in this website. I'll have to give some thought to how that should be implemented (tags? link in sidebar?).

Challenge statement

> *Now, this is an open challenge. h4x0r has created his own encryption algorithm and has decided to challenge all the hackers in the world. He has made the code public here.*
>
> *He challenges you to find the text he has encrypted and promises to reward you handsomely if you manage to do so. To make things simpler he has also given you the hint that the text he has encrypted is only alphanumeric.*
>
> *The text encrypted using the above algorithm is:*
>
> *168 232 100 162 135 179 112 100 173 206 106 123 106 195 179 157 123 173*
>
> *The flag is the SHA-256 of the decrypted text.*

Here is the code used to encrypt the message.

```
1  <?php
2  // h4x0r's ultimate encryption algorithm
3  // I put the string to be encrypted here
4  $str = 'samplestring';
5
6  // I convert the string to ASCII
```

```
 7
 8  for ($i = 0; $i<strlen($str); $i++)
 9     $dec_array[] = ord($str{$i});
10  $ar = $dec_array;
11  $max = max($ar);
12
13  // I generate a random key in between 10 and the maximum value of the ASCII
14  // So the key is different everytime B)
15
16  $key = rand(10,$max);
17
18  // Multiply the key by 101 to increase complexity
19
20  $key = 101*$key;
21
22  // Using this key I encrypt my string using the cool algorithm below
23
24  for($i=0;$i<strlen($str);$i++)
25  {
26      $x = $ar[$i];
27      $am = ($key+$x)/2;
28      $gm = sqrt($key*$x);
29      $enc = $am + $gm;
30
31      $encrypt = floor($enc)%255; // This is the final encrypted number
32
33      // the numbers are printed
34      echo $encrypt.' ';
35  }
36  ?>
```

Every time the program is executed, a new key is generated. The challenge boils down to finding the key used to encrypt the message. From line 16 it is obvious that the number of keys is finite. Also from line 31 we know that there can be more than one key for each letter. If we calculate sets of all the possible keys for all characters in the message and take their intersection, we should arrive at the key that is common to all the characters and that will be the key that was used to generate the message. Then we can use the same key to decrypt it.

From line 16, the key is a random number between 10 and the max ASCII char code in decimal among all chars in the message and from the challenge statement we know that the key is only alphanumeric. So `$max` can only be from 48 to 57 or 65 to 90 or 97 to 122. A modifier is applied to the key in line 20 by multiplying it with 101.

Generating the set of all possible keys by:

```
1  keys = [101*z for z in range(48, 122)]
```

Finding the possible keys for the first few chars of the encrypted message.

```
import math

allchars = range(48, 57) + range(65, 90) + range(97, 122) # 0-9 A-Z a-z

def getkeys(enchar):
    possiblekeys = []
    for letter in allchars:
        for key in keys:
            am = (letter + key)/2
            gm = math.sqrt(key*letter)
            enc = am + gm
            num = math.floor(enc)%255
            if num == enchar:
                possiblekeys.append(key)
    return possiblekeys

p1 = set(getkeys(168)) #first encrypted char
p2 = set(getkeys(232)) #second encrypted char
p3 = set(getkeys(100)) #third encrypted char
p4 = set(getkeys(162)) #fourth encrypted char
p5 = set(getkeys(135)) #fifth encrypted char
```

Then we find the intersection of these sets to get the key.

```
key = int(list(set.intersection(p1, p2, p3, p4, p5))[0]) # aha! the key!
```

Now with key in hand, decrypt the message by the same algorithm:

```
def decrypt(enchar):
    dchar = None
    for letter in allchars:
        am = (letter + key)/2
        gm = math.sqrt(key*letter)
        enc = am + gm
        num = math.floor(enc)%255
        if num == enchar:
            dchar = chr(letter)
    return dchar

encr = '168 232 100 162 135 179 112 100 173 206 106 123 106 195 179 157 123 173'.split()
encr = [int(x) for x in encr]
dcphr = ''
for i in encr:
    if decrypt(i) is not None:
        dcphr += decrypt(i)
    else:
        dcphr += '{' + str(i) + '}'

print 'Flag is: ' + dcphr
```

But wait! 2 chars {195}, {206} were not decrypted. This means that either they have a different key, or this was done deliberately to confuse people. Nonetheless, these two chars can be easily guessed by looking at the whole message string.

Flag is `my*********n`
