balidani

Thursday, October 24, 2013

Hack.lu CTF - Crypto 200 (Geier's Lambda)

Hack.lu CTF was great! !SpamAndHex finished at #15, so there is room for improvement, but at the end of the first day we were at #3, which was pretty nice. Geier's Lambda was a crypto challenge, which we managed to solve third out of all the teams, even landing a bonus point. I did not work on this alone, thanks to the 2 other guys that worked on this!

The task

We were given a Haskell source for a cryptographic cipher. The task was to find a collision with a given key, which was "Le1sRl6l". First we wanted to identify the cipher to see if it's an already existing one. By googling the hex version of the integer constants (2654435769, 3337565984) we found that this is most probably the xTea cipher.

First we were trying to bruteforce the collision, but we were looking at it the wrong way. However, we found something important when playing with the haskell code -- the cipher only uses the first 4 characters from the key.

After inspecting the source in detail we found that the "hash" function of the code is not used anywhere, and this gave us a clue. We translated the haskell version to python:

```
def hash(passwd):
    acc = (1, 0)
    for x in passwd:
        (a, b) = acc
        acc = (a + ord(x), a + b + ord(x))
    (a, b) = acc
    return a | (b << 16)
```

All this function does is collecting the sum of ASCII values into one part of the tuple, and collecting another aggregate value into the other. It is quite easy to find a collision for this. This is the bruteforce approach:

```
def brute():
    arr = string.ascii_letters + string.digits
    for x in arr:
        for y in arr:
            for z in arr:
                for w in arr:
                    s = "%c%c%c%c" % (x,y,z,w)
                    if hash(s) == hash("Le1s"):
                        print s
```

This gave us a list of around 1000 possible keys. Then we wrote a function that evaluates the resulting key by executing the Haskell binary (that we compiled from the source) and checking the number of ASCII characters in the deciphered result:

```
for pwd in passwords:
    process = Popen(['./pwd_check', pwd], stdout=PIPE)
    stdout, stderr = process.communicate()
    dat = stdout
    hex_str = "%x" % (int(dat))
    if len(hex_str) % 2 != 0:
        hex_str = "0" + hex_str
    res =  unhexlify(hex_str)
    score = 0
    for ch in string.ascii_letters + string.digits + " _":
        if ch in res:
            score += 1
    if score > 5:
        print res
```

And after running this, we got the key straight away: `T3aP4rTy`

Blog Archive

► 2014 (13)

▼ 2013 (12)

► December (1)

▼ October (2)

NotSoSecure CTF writeup

Hack.lu CTF - Crypto 200 (Geier's Lambda)

► September (2)

► August (1)

► June (1)

► May (5)

About Me

balidani

View my complete profile

Thanks again to the FluxFingers team for the nice challenges.

Posted by balidani at 5:33 AM

5 comments:

Dor1s October 24, 2013 at 1:10 PM

May be flag is T3aP4rTy?
Decryption key - c3Po?

Reply

balidani October 24, 2013 at 1:19 PM

Oh, I accidentally copy-pasted the wrong password. Thanks, I fixed it now! Nice easter egg, I didn't see the decryption hey was c3Po :)

Reply

Dor1s October 24, 2013 at 1:39 PM

All right!
We didn't solve it. We had yTr4Pa3T and few other variants, lol.

Reply

Андрей Трифонов October 24, 2013 at 1:51 PM

Why did you take so strange culling function?

```
for ch in string.ascii_letters + string.digits + " _":
if ch in res:
score += 1
if score > 5:
print res
```

Reply

balidani October 24, 2013 at 2:01 PM

I had no idea what the result would be after deciphering, but I expected it to be some text. I was in a hurry (this is why my "CTF" code is always a mess), so I didn't think it through completely, but the first result was the correct key :)
I could have tuned the parameters in case we didn't get it on the first try.

Reply

Enter your comment...

Comment as: Select profile...

Publish Preview

Newer Post Home Older Post