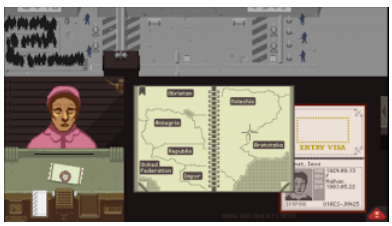
Monday, September 23, 2013

CSAW CTF Quals 2013 - slurp (crypto 500)

Another CSAW write up - crypto 500! This task was solved by adam_i, fel1x and redford on behalf of Dragon Sector.

The challenge was:

"We've found the source to the Arstotzka spies rendevous server, we must find out their new vault key." You are also provided with a slurp.py python script and the ip:port.
Trivia: Arstotzka is the place of a indie game called "Papers, Please". The game has a very unusual gameplay. In the game you play a border control guard who is checking the the passports of persons wanting to enter Arstotzka. The game plays in a fictional Eastern Block state but the setting could also portrait modern day USA.



slurp.py is the server listening on 128.238.66.222:7788. The goal is to pass it's authentication scheme to get the flag.
The authentication protocol is as follows:

Phase 1, sha1 challenge

Server -> Client:
  base64_encoded_24bit_urandom
Client -> Server:
  The client has to find a value so that the hash

$$sha1(base64\_encoded\_24bit\_urandom \; + \; client\_choosen\_chars)$$

  ends with "\xFF\xFF\xFF". If the client is able to generate such a hash, the server continues to phase 2.
We solved this phase with a simple brute force of the sha1 hash. After a few seconds you can find a sha1 hash which suffices the server-supplied challenge.

Phase 2, authentication

Remark 1:
The values sent by the client are transfer in a way that would potentially allow negative values. The formatting uses a unsigned short as packed length, concatenated with a string of the value in hexadecimal notation. However we couldn't find any use for this during the exploitation.
Remark 2:
The random number generation used by the server is not perfect. It generates a 2048 bit number from 320 bit output of urandom, however it's hashed in such a way that a portion of the resulting number will be contain zero bit. We couldn't find a way to use this during the exploitation.

```
def cryptrand(self,n=2048):
 p1=self.hashToInt(os.urandom(40))<<1600
 p1+=self.hashToInt(p1)<<1000
 p1+=self.hashToInt(p1)<<500
 p1+=self.hashToInt(p1)
 bitmask=((2<<(n+1))-1)
 p1=(p1&bitmask)
 return  (p1% self.N)
```

Server -> Client:
  "Welcome to Arstotzka's check in server, please provide the agent number"
Client -> Server:
  The client chooses the values of index and cEphemeral and send them to the server.
  Index has to be at least 2, but not greater than N/2 (N is constant, known prime used as modulus in all operations).
  The only restriction on cEphemeral is that cEphemeral % N != 0.
Server -> Client:
  The server sends the values of sEphermeral, salt.
  The client already knows salt because it's sha512(index).
Client -> Server:
  The client calculates the gennedKey and sends it to the server.
Server -> Client:
  The server checks whether the server-calculated gennedKey is the same as the one provided by the client. Then it tells the client whether authentication was successful or not. If the authentication was successful, the server sends:
  "Well done comrade, the key to the vault is $flag_value".
If you're able to crack the authentication, you get the flag and solve the challenge.
So how does the server test authentication?

Links

Dragon Sector Web Site
[dragonsector.pl]

Contact

contact@dragonsector.pl

Archives

► 2017 (6)
► 2016 (4)
► 2015 (4)
► 2014 (25)
▼ 2013 (22)
► December (3)
▼ September (11)
No cON Name Facebook CTF Quals 2013 - all three le...
CSAW Quals 2013 - Diary (Exploitation 300)
CSAW CTF Quals 2013 - slurp (crypto 500)
CSAW CTF Quals 2013 - RECON (all)
CSAW CTF Quals 2013 - SCP-hack (exploitation 500)
CSAW CTF Quals 2013 - GameMan (exploitation 400)
CSAW CTF Quals 2013 - CryptoMatv2 (web 400_2)
ASIS CTF Finals 2013 - Inaccessible (forensics 312...
ASIS CTF Finals 2013 - memdump
ASIS CTF Finals 2013 - ~Windows (stegano 106)
ASIS CTF Finals 2013 - Chessboard (stegano 175)
► August (4)
► July (4)

Popular Posts

CONFidence DS Teaser CTF registration is open!

Without further ado: http://ctf.dragonsector.pl/ . The teaser will start on the 26 th of April, 9:00 A.M. CEST (GMT+2) - for other tim...

0CTF 2017 - EasiestPrintf (PWN 150)
The task, as the name implies, was a rather basic (at first glance - there was a plot twist) format string bug in a short 32-bit Debian appl...

The client-supplied values are: index, cEphemeral
The server settings are: password, 2048 bit modulus N
The password is actually empty in the provided sources which let us to a dead end; more on this later.
Then the server calculates:

$$salt = sha512(index)$$

$$storedKey = index^{sha512(salt,password)} mod\ N$$

$$sEphemeralPriv = cryptrand()\ \#a\ 2048\ bit\ value,\ random\ in\ every\ connection$$

$$sEphemeral = index^{sEphemeralPriv} + 3 * storedKey\ mod\ N$$

$$sEphemeral = index^{sEphemeralPriv} + 3 * index^{sha512(salt+password)} mod\ N$$

$$\Longleftrightarrow$$

$$index^{sEphemeralPriv} = sEphemeral - 3 * index^{sha512(salt+password)} mod\ N$$

Note that sEphemeral and salt are now sent to the client

$$slush = sha512(cEphemeral, sEphemeral)$$

$$agreedKey = sha512((cEphemeral * storedKey^{slush})^{sEphemeralPriv} mod\ N)$$

$$gennedKey = sha512(sha512(N) \oplus sha512(index), sha512(index), salt, cEphemeral, sEphemeral, agreedKey)$$

The only unknown value is agreedKey.
Then, the compare on the client-supplied gennedKey is done. So, the client needs to find a known agreedKey.

First attempt

Our first attempt was to calculate the agreedKey, which is actually possible without knowing the randomly generated sEphemeralPriv.

$$agreedKey = sha512((cEphemeral * storedKey^{slush})^{sEphemeralPriv} mod\ N)$$

$$agreedKey = sha512(agreedKey\_withouthash)$$

$$agreedKey\_withouthash = (cEphemeral * storedKey^{slush})^{sEphemeralPriv} mod\ N$$

$$agreedKey\_withouthash = (cEphemeral * index^{sha512(salt,password)*slush})^{sEphemeralPriv} mod\ N$$

If we choose

$$cEphemeral = index^{xxx}$$

we get

$$agreedKey\_withouthash = (index^{xxx} * index^{sha512(salt,password)*slush})^{sEphemeralPriv} mod\ N$$

$$agreedKey\_withouthash = (index^{xxx+sha512(salt,password)*slush})^{sEphemeralPriv} mod\ N$$

$$agreedKey\_withouthash = (index^{sEphemeralPriv*xxx+sha512(salt,password)*slush}) mod\ N$$

With the following value from above:

$$index^{sEphemeralPriv} = sEphemeral - 3 * index^{sha512(salt+password)} mod\ N$$

We get the following:

$$agreedKey\_withouthash = (sEphemeral - 3 * index^{sha512(salt+password)})^{xxx+sha512(salt,password)*slush} mod\ N$$

So, we had a formula for calculating agreedKey and knew all variables of the formula. By calculating agreedKey we were able to calculate gennedKey and send it to the server. This all worked well for our test environment. However it did not work on the live flag server. Our guess was that the *password* is different on the live flag server, thus we were not able to calculate the correct agreedKey.

Second attempt

Our second attempt which was successful in the end, was to carefully choose the index. Let's look again at this equation:

$$agreedKey\_withouthash = (cEphemeral * index^{sha512(salt,password)*slush})^{sEphemeralPriv} mod\ N$$

We can set cEphemeral to 1 (the value is from the client), which simplifies the formula to:

$$agreedKey\_withouthash = index^{sha512(salt,password)*slush*sEphemeralPriv} mod\ N$$

Because the exponential is changing for each connection on random, we can assume that it's divisible by some small number, e.g. 3 or 4 (if not, we can retry until it is). So, let's now find the index, such that:

$$index^3 = 1\ (mod\ N)$$

(index is cubic root of 1 modulo prime N)
If we manage to find such index and (sha512(salt, password) * slush * sEphemeralPriv) will be divisible by 3, agreedKey_withouthash will equals to 1. How to find it? One line in mathematica:

```
Reduce[x^3 == 1, x, Modulus->5924486056224…(the value of N)]
```

Result: http://wklej.org/hash/3ca3fcbd545/txt/
There are 3 solutions, but only the second one satisfies constraints on the index value. Now we have to set the index to this number, calculate agreedKey = SHA512(1) and send the data, until it succeeds ;)
Remarks:
Instead of 3 you could use 4 and then find the root using formula:

$$a^{\frac{N-1}{4}}$$

(*a* can be, for example, 2).
You couldn't use 2 instead of 3, because the only quadratic roots of 1 are 1 and -1 (refused by server's constraints).

Posted by Gynvael Coldwind at 00:33

No comments:

Post a Comment

EKOPARTY CTF 2016 - Malware sample (RE 400)
In short, the reversing category 400 pts challenge was a journey starting with negligible x86-64 boilerplate code, leading through a somewha...

EKOPARTY CTF 2016 - FBI 100
The FBI category is something new that I personally have not seen on a CTF (though in all honesty I did have a rather long break). An FBI ta...

0CTF 2017 - complicated xss (web 177)
Complicated xss was a client-side web security task revolving around, well, XSSes. At the very start you were handed a way to XSS the adm...

0CTF 2017 - char (shellcoding 132)
The code in the "char" task was rather simple - you get to send in 2400 bytes of input (using scanf 's "%2400s", so ...


ASIS CTF Finals 2013 - Chessboard (stegano 175)
This is a task from the ASIS CTF Finals 2013, "Stego" (steganography) category, and it was solved by 2 teams including ours. This ...

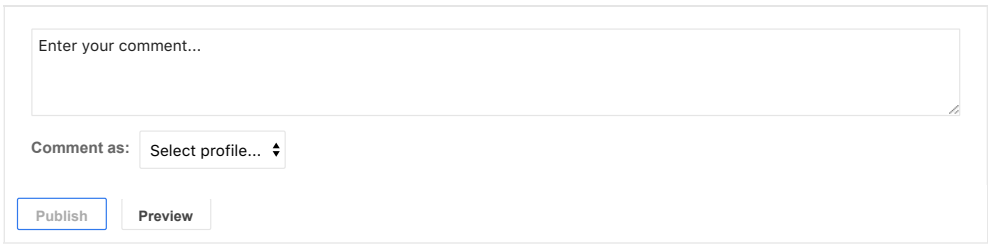
0CTF 2017 - UploadCenter (PWN 523)
Welcome to another Menu Chall right~ Here you can use any function as you wish No more words , Let't begin 1 :) Fill your information...


UFO CTF 2013 - Broken brokoli (forensics 100)
This is a task from UFO CTF 2013, which was a sweet mixture of file format stegano, forensics and decoding weird alphabets (though that'...


Nuit du Hack CTF Quals 2014 - Nibble
The recent Nuit Du Hack CTF Quals CTF was mostly web, crypto and forensics-oriented, with no tasks explicitly categorized as "Exploita...

Contributors

- Adam Iwaniuk
- Gynvael Coldwind
- Jagger
- Krzysztof Katowicz-Kowalewski
- Lympho Cytus
- Marcin Kalinowski
- Mateusz P
- Michał Kowalczyk
- Tomasz Bukowski
- Tomasz Dubrownik
- j00ru
- q3k
- valis



Newer Post Home Older Post

Mathjax