Challenge 04

Flag: FLAG{d1ffi3Prim3}

I noticed in function sendToCNC($msg), the $url = "192.168.223.133/cnc/server.php". This means that it is not possible to test out the php file as it communicates with a private server. Most of the functions, for example bchexdec($hex), does number system conversion. The more important information shown below:

```
// Generate p, g, a;
$p = bin2hex(openssl_random_pseudo_bytes(16));
$pDec = bchexdec($p);
$g = bin2hex(openssl_random_pseudo_bytes(16));
$gDec = bchexdec($g);
$a = bin2hex(openssl_random_pseudo_bytes(16));
$aDec = bchexdec($a);

// Calculate A ( A = g^a mod p)
$ADec = bcpowmod($gDec,$aDec,$pDec);
$A = bcdechex($ADec);

// Send p,g,A (DO NOT SEND a) (tts our private key)
$send = array();
$send['p'] = $p;
$send['g'] = $g;
$send['A'] = $A;
$send = json_encode($send)."\n";
$resp = sendToCNC($send);

// Retrieve B
$resp = json_decode(trim($resp), true);
$B = bchexdec($resp['B']);
$id = $resp['id'];

// Calculate shared secret s
$s = bcdechex(bcpowmod($B, $aDec, $pDec));
$secret = bchexbin($s);

// send encrypted message
$msg = 'some very secret message here'; // decrypt the actual sent msg to get the flag
$enc = xorString($secret, $msg);
$send = array();
$send['msg'] = base64_encode($enc);
$send['id'] = $id;
$send = json_encode($send)."\n";
$resp = sendToCNC($send);
```

There is a key exchange algorithm in place before the message is sent. I look into the pcap to analyse information for filling in possible variables.

I open the pcap in wireshark and followed the two TCP streams. I obtained values for variables $send['p'], $send['g'], $send['A'], $resp['B'], $resp['id'] and $send['msg'].

Wireshark · Follow TCP Stream (tcp.stream eq 0) · captured_msg_2019.pcap

```
POST /cnc/server.php HTTP/1.1
Host: 192.168.223.133
Accept: */*
Content-Type: application/json
Content-Length: 119

{"p":"247510a0783cb074db1692c92d0202c2","g":"9dfe7be3bfe2137afcc40adc3fa22479","A":"207037dcd2f4af78b1a324e4f6e96f11"}
HTTP/1.1 200 OK
Date: Thu, 23 May 2019 03:43:08 GMT
Server: Apache
X-Powered-By: PHP/5.5.12
Content-Length: 61
Content-Type: text/html

{"B":"1c11ffd3845a67ec0977407e3eced3f","id":"2178578919192"}
```

Wireshark · Follow TCP Stream (tcp.stream eq 1) · captured_msg_2019.pcap

```
POST /cnc/server.php HTTP/1.1
Host: 192.168.223.133
Accept: */*
Content-Type: application/json
Content-Length: 109

{"msg":"c5CbTPScpbXvu27qzybUqFSZmFL4i+2j7vQ8+9Njkb1MmZ0e+IulgMaZW\/TfN9e9ScuqTPiVtrs=","id":"2178578919192"}
HTTP/1.1 200 OK
Date: Thu, 23 May 2019 03:43:08 GMT
Server: Apache
X-Powered-By: PHP/5.5.12
Content-Length: 14
Content-Type: text/html

Done Finished
```

Basically a secret key, $secret, is generated from the key exchange algorithm. It is then XOR-ed with the $msg(flag) to encrypt the message. The $enc(encrypted message) is then base64 encoded before sending out as send['msg']. We know that XOR is reversible. Thus, we need to base64 decode send['msg'], then XOR the result with the $secret to get the flag. So now we just have to figure out how to obtain $secret.

```
$decrypt = base64_decode("c5CbTPScpbXvu27qzybUqFSZmFL4i+2j7vQ8+9NjkblMmZ0e+Iu
$actualMsg = xorString($secret,$decrypt); #XOR to get back original message
echo $actualMsg;
```

To obtain $secret, we need to get $a, the private key, which means solving bcpowmod. I tried solving it through bruteforcing (which is clearly not the way) and gave up after a few million iterations. Went to check out more about the key exchange algorithm, it's a Diffie-Hellman key exchange algorithm. Did a research and came across a research on "Attacking Diffie-Hellman Protocol implementation in the Angler Exploit Kit" link: https://securelist.com/attacking-diffie-hellman-protocol-implementation-in-the-angler-exploit-kit/72097/

It suggested that using a modified Pohlig-Hellman algorithm to solve the discrete logarithm. Going through the article, Appendix A gave the attack implementation in Java. All I have to do is to find the prime factorisation of p, the expansion of the Euler function for p and its prime factor, modify the values in the attack algorithm (Main.java). After executing the script for a period, the shared secret ($secret) is obtained.

```
static BigInteger p = new BigInteger("247510a0783cb074db1692c92d0202c2", 16); //2 × 3 × 17 × 22367366261 × 2124069786361309
static BigInteger psi = new BigInteger("1520315099946987476199401822758524800O"); //Euler function for p = 1 x 2 x 16 x 223
static BigInteger g = new BigInteger("9dfe7be3bfe2137afcc40adc3fa22479", 16).mod(p);
static BigInteger A = new BigInteger("207037dcd2f4af78b1a324e4f6e96f11", 16);
static BigInteger B = new BigInteger("01c11ffd3845a67ec0977407e3eced3f", 16);
static long[] q = new long[]{256L, 125L, 7L, 13L, 17L, 131L, 149L, 241L, 263L, 4252351L, 58375935947731L};
//factor of psi(Euler function prime factors) = 2^8 * 5^3 * 7 * 13 * 17 * 131 * 149 * 241 * 263 * 4252351 * 58375935947731
```

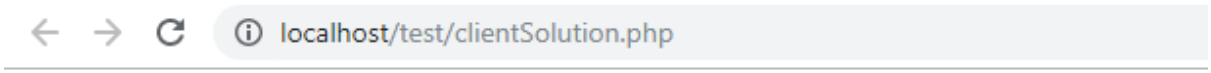
```
[255]
[72]
[2]
[7]
[16]
[44]
[52]
[43]
[210]
[2262349]
[i] Processing 1
[i] Processing 1000001
[i] Processing 2000001
[i] Processing 3000001
[i] Processing 4000001
[i] Processing 5000001
[i] Processing 6000001
[i] Processing 7000001
[57780195143280]
20f8fa3e91f885c68ad81c8fbb06b1db


...Program finished with exit code 0
Press ENTER to exit console.
```

Including the $secret value, I run the code below (in clientSolution.php):

```
$decrypt = base64_decode("c5CbTPScpbXvu27qzybUqFSZmFL4i+2j7vQ8+9NjkblMmZ0e+Iu
$actualMsg = xorString($secret,$decrypt); #XOR to get back original message
echo $actualMsg;
```

The flag is obtained:



Shared secret established, the flag is FLAG{d1ffi3Prim3}