Pico 2017 CTF Write-Up

1. Master Challenge: Lazy Dev
I really need to login to this website, but the developer hasn't implemented login yet. Can you help?

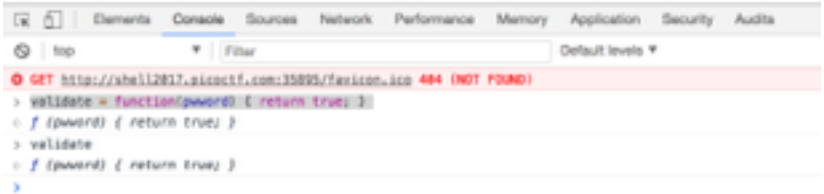
Solution: When I saw the source code, I noticed we need to override the current "validate" function in order to bypass the authorization. I opened Chrome's developer tools and simply wrote another "validate" function that always returned true. I entered a character in the input box and submitted it. I got the flag back.

index.html

```
<!DOCTYPE html>
<html lang="en">
<body>
   <h1>Enter the password</
h1>
   <input id="password">
   <button type="button"
onclick="process_password()"
>Submit</button>
   <p id="res"></p>

</body>
<script type="text/javascript"
src="/static/client.js"></
script>
</html>
```

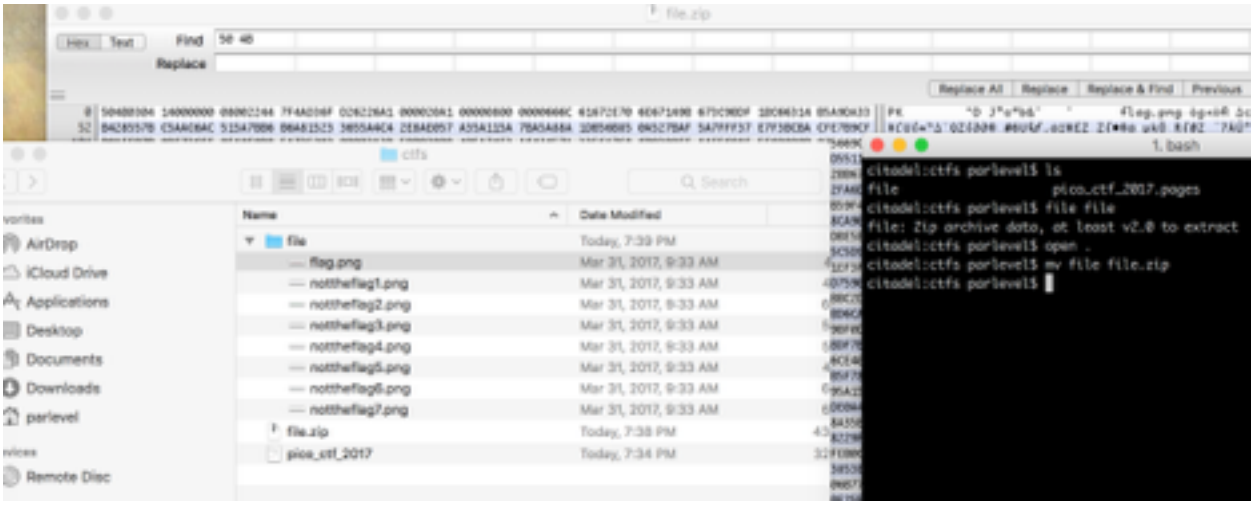
static/client.js

```
//Validate the password. TBD!
function validate(pword){
  //TODO: Implement me
  return false;
}

//Make an ajax request to the server
function make_ajax_req(input){
  var text_response;
  var http_req = new XMLHttpRequest();
  var params = "pword_valid=" + input.toString();
  http_req.open("POST", "login", true);
  http_req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  http_req.onreadystatechange = function() {//Call a function when the state changes.
          if(http_req.readyState == 4 && http_req.status == 200) {
     document.getElementById("res").innerHTML = http_req.responseText;
    }
  }
  http_req.send(params);
}

//Called when the user submits the password
function process_password(){
  var pword = document.getElementById("password").value;
  var res = validate(pword);
  var server_res = make_ajax_req(res);
}
```



2. Master Challenge: Missing Identity
Turns out, some of the files back from Master Challenge 1 were corrupted. Restore this one file and find the flag. The flag starts with the character z. Filename is “file”, inside the current folder.

Solution: For this challenge, the file seemed to be corrupted. Inspecting the file with a hex editor, we can see names of images such as “flag.png” when we see read the hex ascii characters. I assumed this file might be a zip file with a few images compressed in the folder. I looked for the header of a zip file which is 50 4b 03 04 14 00 00 00 08. I noticed a few weird “XXXX” characters at the beginning of the file, so I changed those characters to the zip file header. Then I added the extension .zip to the file. I uncompressed the file and I found the flag.



3. Master Challenge: War
Win a simple Card Game. Source. Connect on shell2017.picoctf.com:4415. File is “war” inside the current folder.

Solution: For this problem, I first tried entering simple strings to see if I could overflow some memory when the binary asked for the a username. I noticed that there was validation when entering numbers to bet, so the only option to exploit was the input name. I tried strings such as “1234567890abcdefghilkmnopkzxywv” and then kept betting one coin each turn. I figured there was an array from which the winnings cards/suits where chosen. If we could overflow the memory after the input name variable, maybe our cards/suits would be selected from the winning array. I finally was successful with ‘1234567890abcdefghilkmnopkzxywvqaaaaa’.

```
How much would you like to bet?
33
you bet 33.
The opponent has a 0 of suit 0.
You have a 10 of suit 3.
You won? Hmmm something must be wrong...
You actually won! Nice job

You have 66 coins.
How much would you like to bet?
66
you bet 66.
The opponent has a 0 of suit 0.
You have a 13 of suit 3.
You won? Hmmm something must be wrong...
You actually won! Nice job

You have 132 coins.
How much would you like to bet?
132
you bet 132.
The opponent has a 0 of suit 0.
You have a 11 of suit 1.
You won? Hmmm something must be wrong...
You actually won! Nice job

You have 264 coins.
How much would you like to bet?
264
you bet 264.
The opponent has a 0 of suit 0.
You have a 12 of suit 1.
You won? Hmmm something must be wrong...
You actually won! Nice job
You won the game! That's real impressive, seeing as the deck was rigged...
$
```

4. Master Challenge: weirderRSA
Another message encrypted with RSA. It looks like some parameters are missing. Can you still decrypt it? Message

Solution: We use Fermat's Theorem to solve the problem. If we can get a multiple of p we can calculate q, then phi and finally the private key d. Then we just plug in to the decryption equation.

Fermat's Theorem: If p is prime and 0 < a < p, a^(p-1) = 1 mod p.

d*e = 1 mod (p-1)
((d mod (p-1))*(e mod (p-1))) mod (p-1) = 1 mod (p-1)
dp*e = r(p-1)+1

```
>>> r = 123456
>>> p = gmpy2.gcd(n, pow(r, (e*dp), n) - r) // gcd(n, (r^edp mod n) - r)
>>> q = gmpy2.div(n, p) // since n = p*q
>>> f = (p-1) * (q-1)
>>> d = gmpy2.invert(e, f) // since d = 1 / ( e mod ϕ(n))
>>> m = int(pow(c,d,n)) // m = c^d mod n
```

We convert the result of m into hex and then to ascii.