This writeup assumes admin-panel-hard has been solved. If not, please do before attempting this challenge.

Now this IS impossible…… OR IS IT?!

The main idea is that this time, instead of xoring with 1 random byte, we xor with 32 random bytes. Although, what struck me as odd is the use of 2 different random generators, namely **os.random()** and **random.randbytes()**.

The first one is cryptographically secure, but the second one…. CAN BE PREDICTED! That is, if the right conditions are met.

Python's random library is based on Mersenne Twister, a PRNG. Long story short, we can use a tool like https://github.com/tna0y/Python-random-module-cracker, a.k.a **randcrack**, to predict numbers. What we need to achieve this is, as a general rule, 624 32-bit integers, or 19968 bits.

Now, in our program we use randbytes(32). Randcrack does not support randbytes, so let's look into the source code to see what randbytes is.

```
## -------------------- bytes methods ---------------------

def randbytes(self, n):
    """Generate n random bytes."""
    return self.getrandbits(n * 8).to_bytes(n, 'little')
```

This is taken from random.py found in the github repo of Python.

This means randbytes essentially generates n*8 bits, or in our case 256 bits.

However, there is a random issue. Is generating 256 bits the same as generating 32 bits 8 times?

```
import random

random.seed(69)
rand_bytes = random.randbytes(32)
rand_32bit_nums = [int.from_bytes(rand_bytes[i:i+4], 'little') for i in range(0,
len(rand_bytes), 4)]
print(rand_32bit_nums)
random.seed(69)
rand_getrandbits = [random.getrandbits(32) for _ in range(8)]
print(rand_getrandbits)
```

Running this little program proves how we can convert the randbytes(32) to 8 32 bit integers.

624/8=78, so we need 78 randbytes(32) strings to predict the next ones.

Now, for the next step, let's make a POC to simulate what we need to do with the strings from the server:

```
import random
from randcrack import RandCrack
rc = RandCrack()
rand_32bit_nums = []
for _ in range(78):
    rand_bytes = random.randbytes(32)
    rand_32bit_nums.extend([int.from_bytes(rand_bytes[i:i+4], 'little') for i in
range(0, len(rand_bytes), 4)])
for num in rand_32bit_nums:
    rc.submit(num)
predicted_values = [rc.predict_randrange(0, 4294967295) for _ in range(8)]
print("Predicted next 8 random values:", predicted_values)
predicted_bytes = b''.join([val.to_bytes(4, 'little') for val in
predicted_values])
print("Predicted byte string:", predicted_bytes)
actual_bytes = random.randbytes(32)
print("Actual generated 32 random bytes:", actual_bytes)

if predicted_bytes == actual_bytes:
    print("Prediction matched the actual generated bytes!")
else:
    print("Prediction did not match the actual generated bytes.")
```

And it works! So, we need to apply this algorithm to the random bytes from the server, predict the next byte, send it, the bruteforce the password like before. We simply need to combine the POC with the code we used in admin-panel-hard, with slight adjustments and logic to extract the hex string.

```
from pwn import *
import re
from randcrack import RandCrack
rc = RandCrack()
context.log_level = 'error'
host = "34.159.151.77"
port = 31044
connection = remote(host, port)
connection.recvuntil("> ")
rand_32bit_nums = []
for i in range(78):
    connection.sendline("2")
    connection.recvuntil("> ")
    connection.sendline("A" * 64)
    output=connection.recvuntil("> ")
    match = re.search(r"XORing with: (\w+)", output.decode('utf-8'))
```

```
    hex_string = match.group(1)
    rand_bytes = bytes.fromhex(hex_string)
    rand_32bit_nums.extend([int.from_bytes(rand_bytes[i:i+4], 'little') for i in
range(0, len(rand_bytes), 4)])

for num in rand_32bit_nums:
        rc.submit(num)
predicted_values = [rc.predict_randrange(0, 4294967295) for _ in range(8)]
predicted_bytes = b''.join([val.to_bytes(4, 'little') for val in
predicted_values])
connection.sendline("2")
connection.recvuntil("> ")
connection.sendline(predicted_bytes.hex())
output=connection.recvuntil("> ")

for i in range(256):
    byte = bytes([i])
    connection.sendline("1")
    connection.recvuntil("> ")
    hex_string = (byte * 16).hex()
    connection.sendline(hex_string)
    try:
        connection.recvuntil("> ", timeout=2)
    except EOFError:
        output = connection.recv()
        print(output.decode())
        break

connection.close()
```

And a small wait we get the flag! Surpisingly easy for an insane chall, I've seen medium challs harder than this one. And easy challs harder than admin-panel-hard for that matter.

Made with love by: AndreiCat