WORMCON 0x01 CTF

wormcon Gallery write up

by arrester (Demon Team)

작성자: 김주원(arrester)

arresterloyal@demonteam.org

https://github.com/Demon-KR

https://blog.naver.com/lstarrlodyl

목 차

WORMCON 0x01 CTF

Ⅰ 문제 개요 및 목표

Ⅰ-1. 문제 개요

App Analysis

Ⅰ-2. 문제 목표

App Analysis

WORMCON 0x01 CTF

Ⅱ 문제 풀이

Ⅱ-1. 정보 수집

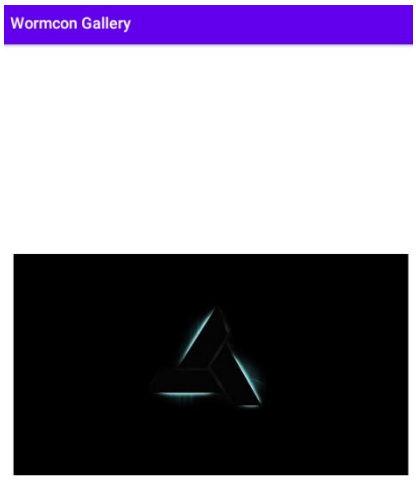

그림 1 wormcon Gallery

앱을 실행하면 이미지가 출력된다.

그림 2 wormcon Gallery next image

next image를 누르면 그림 2와 같은 이미지를 확인할 수 있다.
next image 버튼에 맞춰서 다음 이미지들이 나타나는데 계속 하다 보면 보던 이미지들이 반복되는 것을 확인할 수 있다.

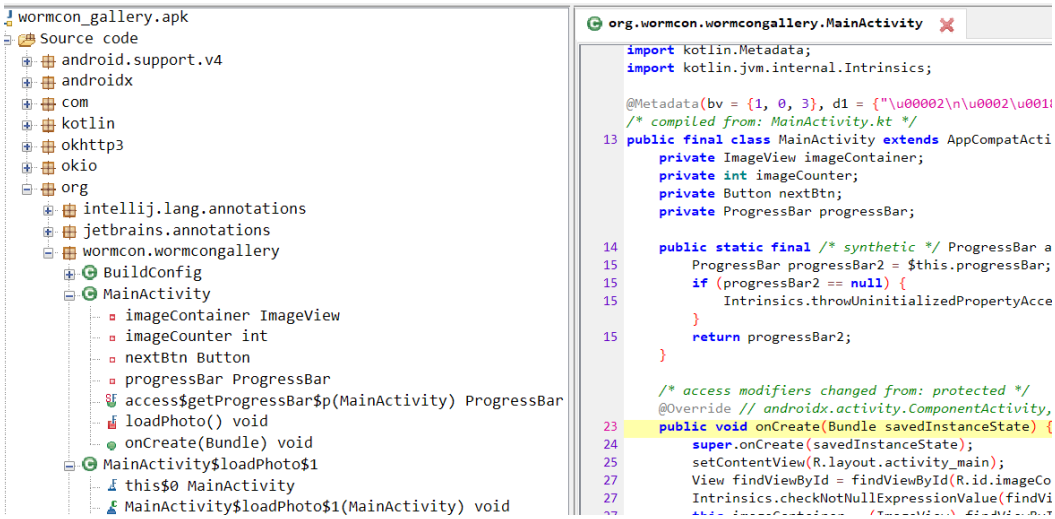

그림 3 jadx check

jadx로 코드를 보면 해당 애플리케이션은 kotlin으로 작성된 것을 확인할 수 있다.

Ⅱ-2. 핵심

MainActivity에 문제의 핵심 코드가 포함되어 있다.

```
public final void loadPhoto() {
    ProgressBar progressBar2 = this.progressBar;
    if (progressBar2 == null) {
        Intrinsics.throwUninitializedPropertyAccessException("progressBar");
    }
    progressBar2.setVisibility(0);
    StringBuilder sb = new StringBuilder();
    sb.append("assets/");
    int i = this.imageCounter + 1;
    this.imageCounter = i;
    sb.append(i);
    String encodeUrl = URLEncoder.encode(sb.toString());
    Picasso picasso = Picasso.get();
    RequestCreator load = picasso.load("https://firebasestorage.googleapis.com/v0/b/wormcon.appspot.com/o/" + encodeUrl + ".jpg?alt=media");
    ImageView imageView = this.imageContainer;
    if (imageView == null) {
        Intrinsics.throwUninitializedPropertyAccessException("imageContainer");
    }
    load.into(imageView, new MainActivity$loadPhoto$1(this));
    if (this.imageCounter == 10) {
        this.imageCounter = 0;
    }
}
```

그림 4 storage load code

Glide대신 picasso를 사용하여 이미지를 로드하고 있는 것을 볼 수 있고 이미지는 Firebase Storage에서 가져오는 것을 확인할 수 있다.

그리고 imageCounter 변수에 값이 next image 버튼을 누를 때마다 증가하며 10이 되면 다시 0으로 초기화 된다. 그러므로 1~10.jpg 이미지가 반복되던 것이다.

```
{
  "prefixes": [],
  "items": [
    {
      "name": "assets/1 bLXAGI1LxP3zQubgzjM1sg.jpeg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/1.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/10.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/113358-nature-Japan.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/1651.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/1821355063_preview_2019-08-01.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/2.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/2560x1600.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/3.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/4.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
      "name": "assets/5.jpg",
      "bucket": "wormcon.appspot.com"
    },
    {
```


그림 5 https://firebasestorage.googleapis.com/v0/b/wormcon.appspot.com/o/

해당 URL에 접근하면 json 형태로 assets 폴더내에 있는 이미지 name 값을 확인할 수 있다.
1~10.jpg 이외 파일들을 확인하면 flag를 확인할 수 있을 것이라 생각했다.
이것을 참고하여 앱에서 로드하는 뒤에 값 .jpg?alt=media를 참고하여 접근하려고 했다.
웹에서는 jpg가 아닌 jpeg, png 등 각 확장자가 변환을 거치지 않기 때문에 웹에서는 해당 확장자에 맞춰서 접근해야 한다. 그래서 그렇게 시도했지만 접근이 불가능했다.

어떻게 해결해야 할까? 하던 중 방법을 고민했다.

1. 앱을 클론코딩 하여 firebase storage 부분만 가져와서 만들고 입력되는 파일 명 변수를 후킹하여 변조하며 값을 확인한다.
2. smali code 수정하여 1번과 같은 내용이 되도록 한다.

이렇게 하다가 다른 방법이 있을 것 같아 팀원들에게 도움을 요청했고 상수형과 재승이가 도움을 주었다. 재승이는 Picasso load 부분 후킹에 대해 방법을 제시하였고 상수형은 웹 캐시를 살펴보면 웹에서도 접근이 가능할 것이라고 했다.

위 내용들을 참고하여 조사하던 중 상수형이 웹에서 접근하는 방법을 알아냈다ㄷㄷ

웹에서 접근하는 방식으로 아래 풀이를 진행한다.

II-3. 풀이

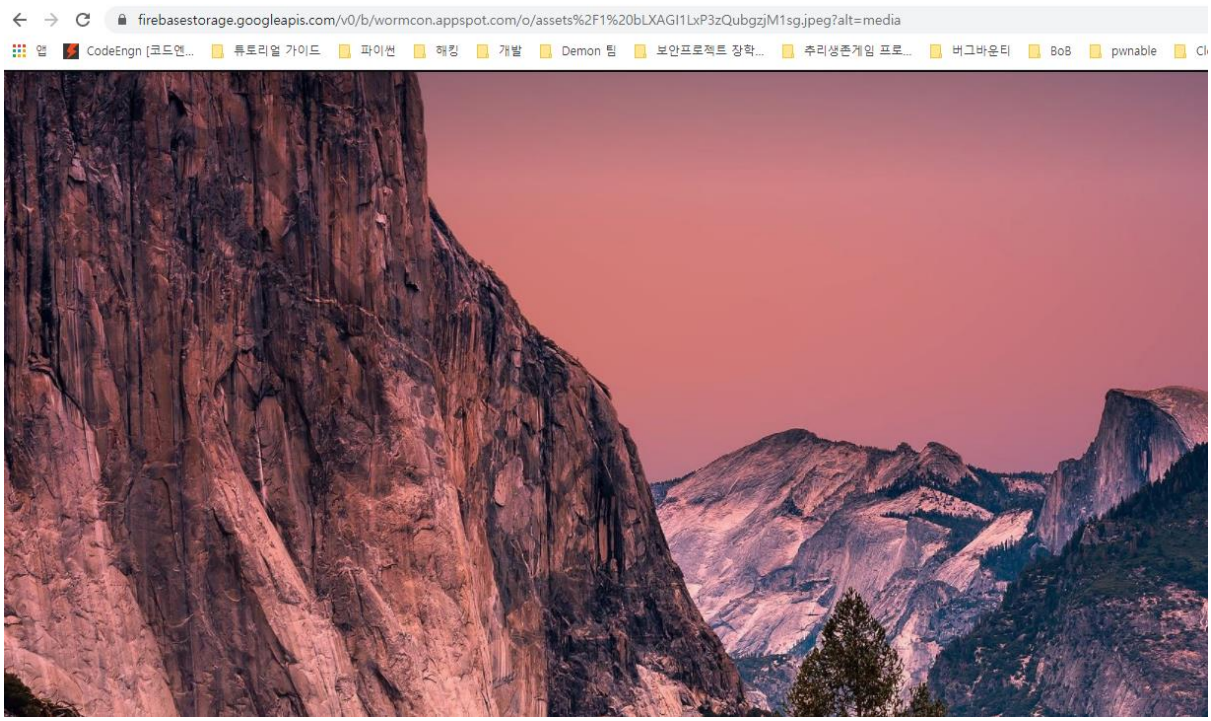

그림 6 갓상수

https://firebasestorage.googleapis.com/v0/b/wormcon.appspot.com/o/assets%2Fname.jpg?alt=media

접근이 불가능한 이유는 assets/ ← 이 부분 때문이다. %2F로 변경하여 접근해야 한다. 위 그림 4의 코드를 보면 URL Encode 부분이 있는데 여기서 변경되어 접근이 되는 것으로 보인다.

storage에 있는 파일은 72개 직접 접근해서 확인 또는 자동화 코드를 작성하는 것도 방법이다.

당시 대회때는 직접 손으로 찾아서 flag를 찾았고 이후 아래와 같은 자동화 코드도 작성했다.

```
import requests

url = "https://firebasestorage.googleapis.com/v0/b/wormcon.appspot.com/o/"

print("start")


with open("./image_value.txt", 'r') as file:
    while True:
        image_value = file.readline()
        r = requests.get(url + image_value.replace("\n", ""))
        if r.status_code == 200:
            print("Success!")
            image_value = image_value.replace("assets%2F", "")
            with open("./img/"+image_value.replace("?alt=media\n", ""), "wb")
as f:
                f.write(r.content)

        else:
            break

        if not image_value: break
```

image_value는 따로 json 내용 값을 정리해서 넣었다.

```
for I in range(1, 72):
    print(image_value['items'][i]['name'].replace("/", "%2F")+"?alt=media)
```

위와 같이 직접 출력 후 따로 txt파일로 생성했다.

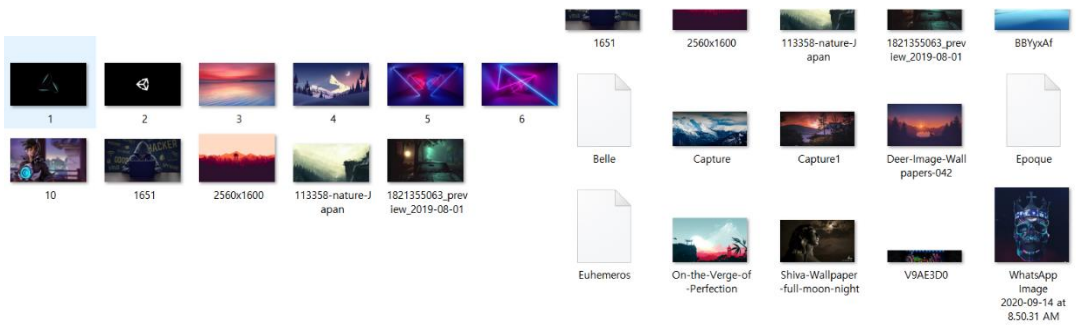

그림 7 image check

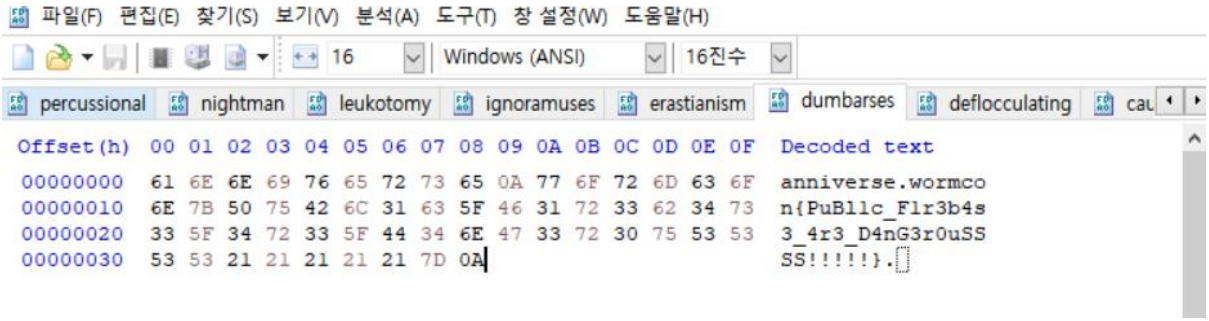

그림 8 flag find

이미지 파일 확장자가 아닌 파일들 내용을 보면 Flag를 찾을 수 있다.