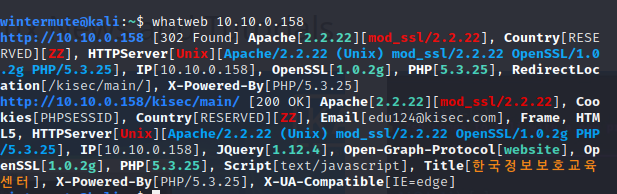
1. **사용한 도구**

|  |  |
| --- | --- |
| **도구 이름** | 도구 정보 |
| **OWASP ZAP** | OWASP TOP 10을 기반으로 웹 취약점을 점검하는 오픈소스 스캐너 |
| **NMAP** | 대상 IP에 열린 포트를 스캔하는 포트 스캐너 |
| **whatweb** | 웹 서버의 정보를 스캔하는 침투 테스트 용 툴 |
| **Burp Suite** | 웹 서비스의 취약점을 프록시를 이용하여 점검할 수 있는 툴 |
| **hashcat** | 해시 함수 크랙용 툴 |
| **Nodejs** | 쿠키 탈취 웹 서버 구축용 프레임워크로 사용 |
| **Python** | Blind SQL Injection 자동화 스크립트 작성용으로 사용 |

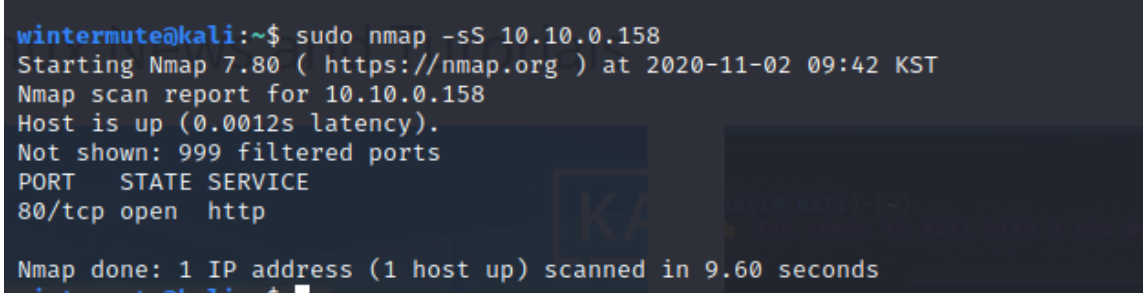
1. **자료수집**

(사전 제공 자료: 타겟 홈페이지 주소: http://10.10.0.158/)

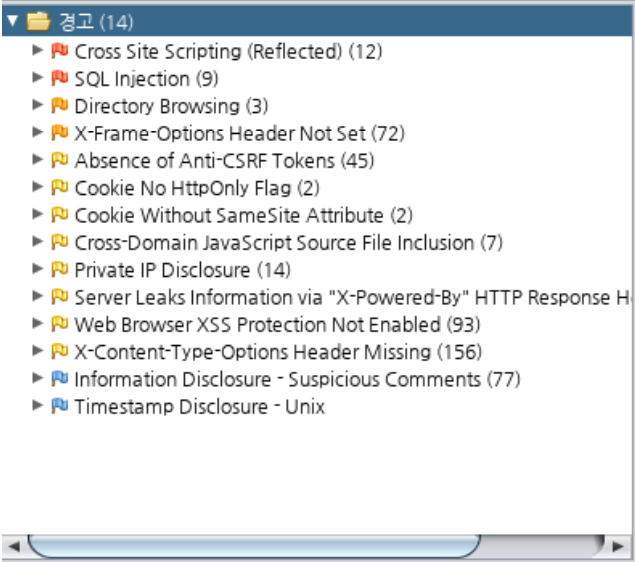
먼저, Kali Linux에서 스캐닝 툴인 OWASP ZAP, NMAP과 whatweb을 이용하여 타겟 페이지의 자료를 수집하였다.



<그림 1.1> whatweb 스캐닝 결과



<그림 1.2> NMAP 포트 스캐닝 결과



<그림 1.3> OWASP ZAP 웹 스캐닝 결과

해당 툴들로 자료 수집을 한 결과, whatweb을 통해 Apache server 2.2.2를 사용하는 서버임을 알았으며( <그림 1.2> ) nmap을 통해 80번 포트가 열려 있어 웹 서비스가 작동 중인 것을 확인하였고 OWASP ZAP을 통해 타겟 웹페이지 몇몇 페이지에서 Blind SQL Injection(이하 Blind SQLi) 또는 XSS 취약점이 발견되었으며 http://10.10.0.158/\_core의 경우 File Directory Indexing 취약점이 존재하는 것을 확인하였다.( <그림 1.3> )

1. **Directory Indexing 및 File Download 취약점을 이용한 타겟 페이지 정보 수집**

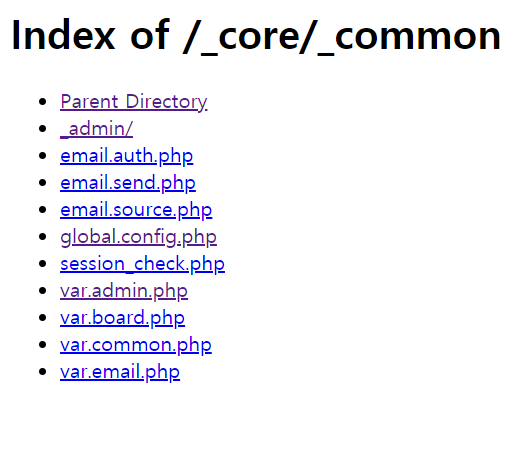
Directory Indexing 취약점이란 웹 서버의 잘 못된 설정에 의해 인덱싱 기능이 활성화되어 있을 경우, 공격자가 브라우저를 통해 서버 내의 파일 및 디렉토리의 정보를 얻을 수 있는 취약점을 말한다.

File Download 취약점이란 파일 다운로드 기능이 있는 웹 페이지에서 파일 경로를 받는 매개변수를 필터링하지 않아 서버 내 중요한 파일 들을 다운로드 할 수 있는 취약점을 말한다.

먼저 File Directory Indexing 취약점을 가진 페이지(<http://10.10.0.158/_core>)로 이동하여 보았다.



<그림 2.1.>Directory Indexing 취약점이 있는 페이지



<그림 2.2> Directory Indexing을 통하여 하위 폴더 이동 확인

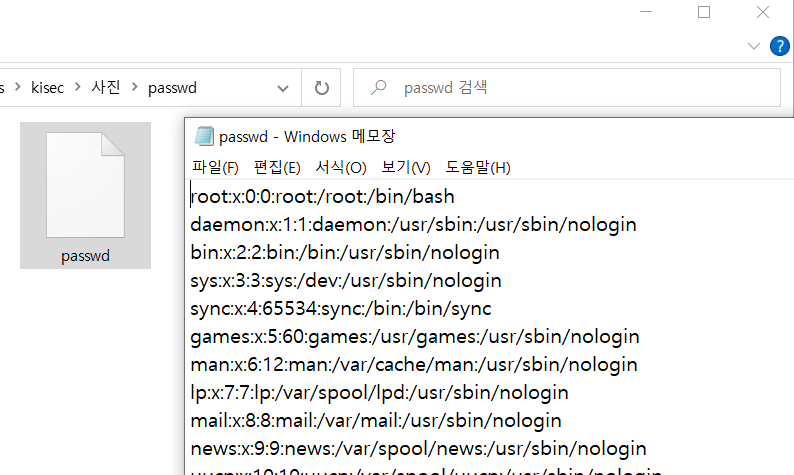
해당 페이지로 이동해본 결과 <그림 2.1>과 같이 Directory Indexing이 되는 취약점이 있었으며 하위 디렉토리로 이동할 수도 있었다. <그림 2.2.>

Directory Indexing을 통해 찾아본 결과 download.php란 파일이 \_core폴더에 존재하였고, 해당 파일을 이용해 파일 다운로드가 가능한 지 찾아보았다. 사용한 URL은 아래 표와 같다.

|  |  |
| --- | --- |
| **다운로드할 파일 이름** | /etc/passwd |
| **파일경로** | ../../../../etc/passwd |
| **다운받은 파일을 저장할 이름** | passwd |
| **URL** | <http://10.10.0.158/_core/_download.php?file_url=../../../../>  etc&file\_name=passwd&real\_name=passwd |



<그림 2.3> File Download URL



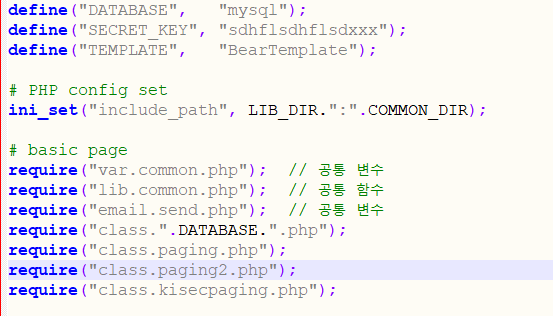
<그림 2.4>다운로드 된 /etc/passwd 파일

그림 <2.4>와 같이 /etc/passwd 파일이 다운로드 되어 시스템 계정 정보가 누출된 것을 확인할 수 있다.

Directory Indexing 취약점을 이용하여 파일들을 찾아보는 중 \_init.php란 파일이 눈에 띄었다.

해당파일을 아래의 표처럼 URL을 구성하여 다운로드 받아보았다.

|  |  |
| --- | --- |
| **다운로드할 파일 이름** | \_init.php |
| **파일경로** | ../../\_core/\_init.php |
| **다운받은 파일을 저장할 이름** | include\_check |
| **URL** | <http://10.10.0.158/_core/_download.php?file_url=../../>\_core&file\_name=\_init.php &real\_name=include\_check |



<그림 2.5>다운로드한 \_init.php파일 확인

File Download 취약점을 이용하여 \_init.php 파일을 다운로드 하여 (<그림 2.5>) 웹페이지의 DB는 MySQL이라는 것도 알 수 있었다. 또한 var.common.php를 require하는 코드 주석에 공통변수라고 써 있었다. 저 파일에 DB connection 관련 정보가 있는 지 확인하기 위해 해당 파일을 다운로드 받아보았다. URL은 아래 표와 같이 구성했으며 아래 <그림 2.6>에 해당 파일의 내용이다.

|  |  |
| --- | --- |
| **다운로드할 파일 이름** | var.common.php |
| **파일경로** | ../../\_core/\_common/var.common.php |
| **다운받은 파일을 저장할 이름** | dbconn |
| **URL** | <http://10.10.0.158/_core/_download.php?file_url=../../_core>  \_common/\_&file\_name=\_var.common.php &real\_name= dbconn |



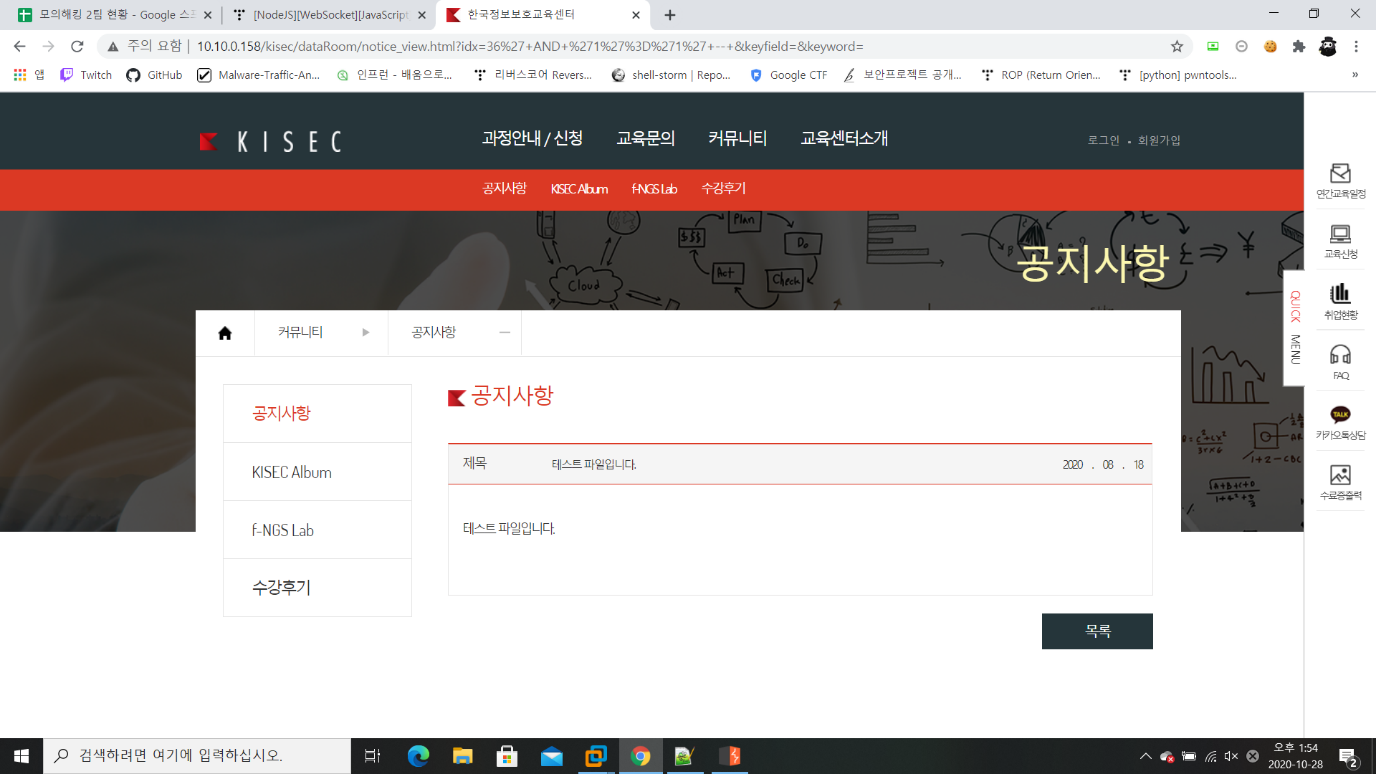
<그림 2.6> var.common.php 파일 다운로드

var.common.php 파일을 다운로드 해보니 DB connection 정보는 아쉽게 없었다. 하지만 DB의 테이블 정보가 그대로 하드코딩 되어있어 DB의 이름을 몰라도 정보수집에서 알아낸 Blind SQLi 취약점을 이용하여 테이블의 정보 및 내부의 데이터를 얻을 수 있다. 위 파일에 나와 있는 admin 테이블(hm\_admin\_tb)을 통해 admin의 아이디와 패스워드를 탈취할 수 있는 지 확인해보겠다.

3.Blind SQLi 취약점을 이용하여 admin 계정정보 탈취하기

SQL Injection(이하 SQLi)란 웹페이지 중 SQL 명령어를 실행하는 부분에서 매개변수를 받을 때 싱글 쿼테이션(‘), 더블 쿼테이션(“), 세미콜론(;) 등의 특수문자를 필터링하지않아 공격자가 악의적인 SQL 쿼리를 실행 할 수 있는 취약점을 말하며 그 중 Blind SQLi는 SQLi를 통한 악의적인 쿼리가 참(True), 거짓(False)의 출력이 달라 공격자가 SQL 쿼리를 통해 DB에 관련 된 정보 또는 DB 내부의 데이터를 유추할 수 있는 공격을 말한다.

먼저 위의 <그림 1.3>에서 수집한 정보를 통해서 몇몇페이지에서 Blind SQLi 취약점이 발생하는 것을 웹 스캐너를 통해 알았으니 정말로 Blind SQLi가 존재하는지 확인하여 보자.



<그림 3.1>Blind SQLi True반응

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 3.2> Blind SQL Injection False반응

위의 <그림 3.1>, <그림 3.2> 는 Blind SQLi가 발생하는 페이지 중 하나로(10.10.0.158/kisec/dataRoom/notice\_view.html)

페이지의 매개변수 중 하나인 idx에서 Blind SQLi이 발생하는 것을 알 수 있다. 그 먼저 근거로 True가 나오는 SQL 구문을 필터링 우회하기위해 URL Encoding과 +문자를 사용하여 아래의 구문이 되도록 삽입하였다. <그림 3.1>

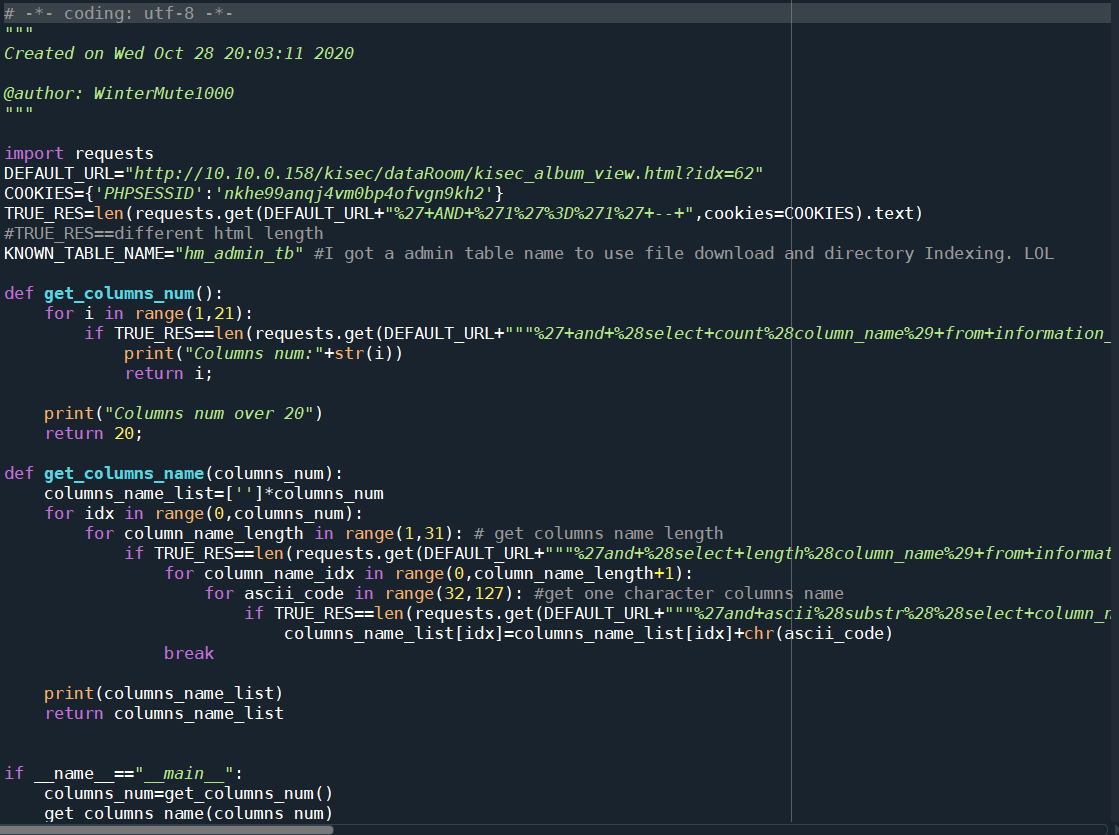
36’ and ‘1’=’1’ –

삽입하여 결과를 보니 공지사항이 잘 출력되었다. 이번엔 False가 나오는 SQL 구문을 같은 방식으로 아래의 구문이 되도록 삽입하였다. (그림 3.2)

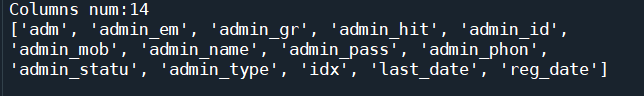
36’ and ‘1’=’2’ –

삽입하여 결과를 보니 이번엔 공지사항이 출력되지 않았다. 즉 True가 나오는 구문과 False가 나오는 구문의 결과가 다르게 나왔으므로 Blind SQLi가 존재하는 것을 알 수 있다.

그 다음으로 Python 스크립트를 작성하여 칼럼의 정보를 알아냈다. 해당 스크립트에서는 위의 페이지가 아닌 KISEC Album 페이지(http://10.10.0.158/kisec/dataRoom/kisec\_album\_view.html)에서 동작하도록 작성하였으나, 해당 페이지에서도 같은 idx 매개변수를 사용하여 위의 공지사항 페이지와 같은 결과가 나왔으므로 Blind SQLi가 있는 것을 알 수 있었으므로 해당 페이지에서 동작해도 무관하다고 생각이 들어 해당 페이지에서 동작하도록 하였다.



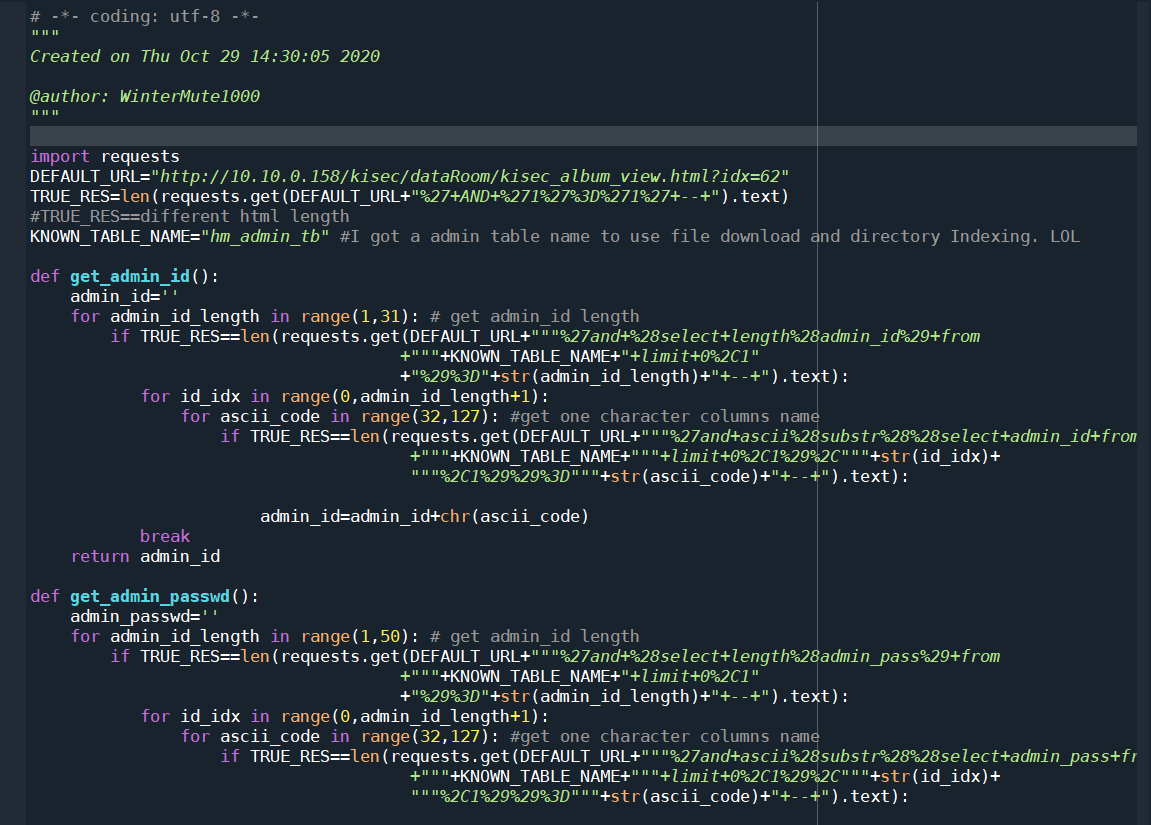
<그림 3.3> Python 스크립트를 통하여 hm\_admin\_tb의 칼럼 정보를 알아내는 스크립트 일부



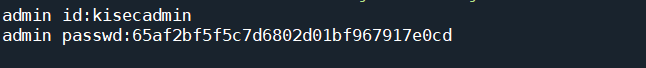
<그림 3.4> <그림 3.3>의 스크립트를 통해 hm\_admin\_tb의 칼럼의 수와 이름을 알아냄

스크립트에서도 역시 URL Encoding과 +문자를 사용하여 필터링을 우회하도록 하였다. 또한 True시 나타내는 response의 경우 html 텍스트 길이가 false나 오류 페이지보다 더 기므로 해당 html의 길이를 이용하여 True인 쿼리를 알아냈다. 사용한 SQL query의 예시는 다음 표와 같다.

|  |  |
| --- | --- |
| **hm\_admin\_tb의 칼럼 수** | ‘and select count(column\_name) from  information\_schema.columns where table\_name=’hm\_admin\_tb’ –- |
| **hm\_admin\_tb에 존재하는 칼럼 이름들의 길이** | ‘and (select length(column\_name) from information\_schema.columns where table\_name=’hm\_admin\_tb’ order by 1 limit 0,1) =1 –- |
| **hm\_admin\_tb에 존재하는 칼럼 이름** | ‘and ascii (substr((select column\_name from information\_schema.columns where table\_name=’hm\_admin\_tb’ order by 1 limit 0,1),0,1))=90 |

위의 예시 SQL문을 통해 Blind SQLi를 실행해 본 결과 <그림 3.5> 와 같이 컬럼의 수와 컬럼들의 이름을 얻을 수 있었다. 해당 결과를 보아 admin\_id가 admin의 아이디, admin\_pass가 admin의 패스워드를 나타내는 칼럼이라고 추측할 수 있다. 추측한 칼럼을 가지고 admin의 아이디, 패스워드를 얻을 수 있는지 확인해보았다. 

<그림 3.5> Python 스크립트를 통하여 hm\_admin\_tb의 데이터 정보를 알아내는 스크립트 일부



<그림 3.6> <그림 3.5>의 스크립트를 통해 admin id와 admin password를 알아냄

Admin의 아이디와 패스워드를 알아내는데 사용한 SQL 쿼리의 예시는 다음 표와 같다.

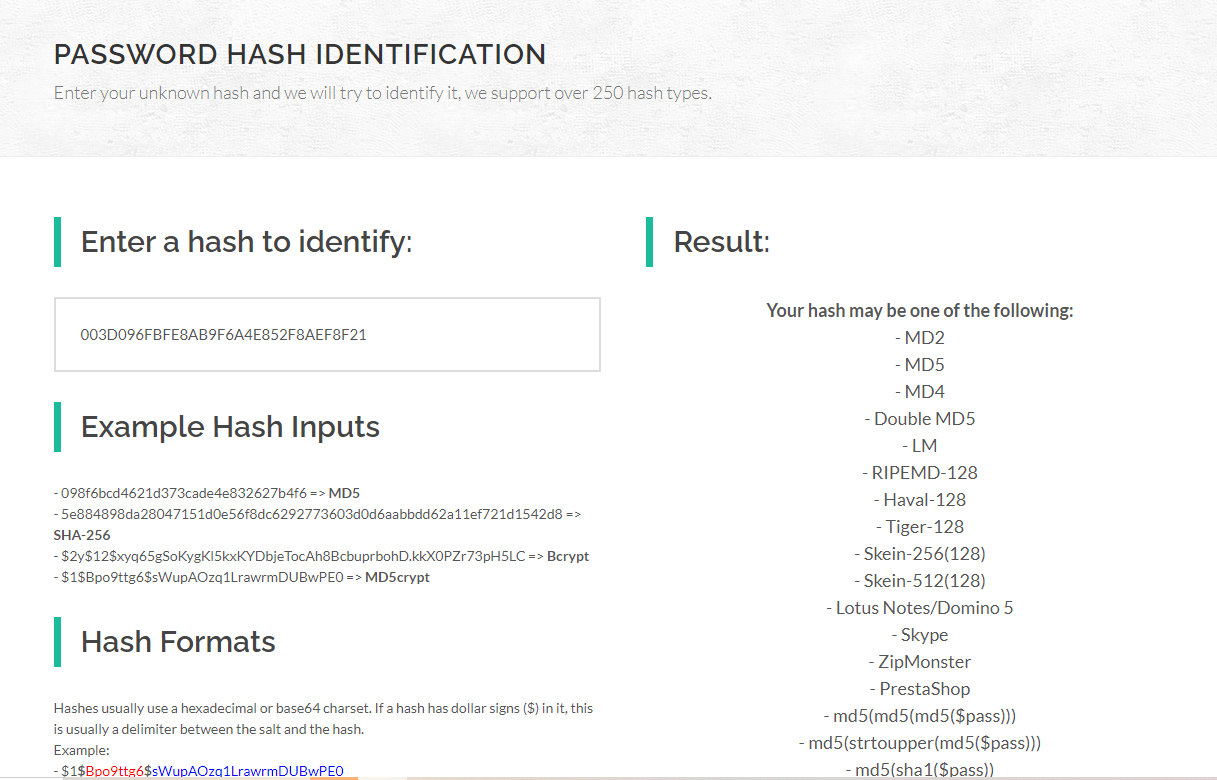
|  |  |
| --- | --- |
| **Admin 아이디의 길이** | ‘and (select length(admin\_id) from hm\_admin\_tb limit 0,1) =4 -- |
| **Admin 아이디** | ‘and ascii (substr((select admin\_id from hm\_table\_tb limit 0,1),0,1))=90 –- |
| **Admin 패스워드의 길이** | ‘and (select length (admin\_pass from hm\_admin\_tb limit 0,1) =4 -- |
| **Admin의 패스워드** | ‘and ascii (substr((select admin\_pass from hm\_table\_tb limit 0,1),0,1))=90 |

<그림 3.6>을 보면 admin의 아이디는 kisecadmin인 것을 알아냈으나, 패스워드의 경우는 암호화 되어있는 것으로 보인다. 하지만 그렇다 하더라도 패스워드가 유추하기 쉽고 취약한 패스워드인 경우 크랙 할 수 있을 것으로 보인다. 다음은 패스워드를 크랙 하여 관리자권한을 얻을 수 있는 지 확인할 것이다.

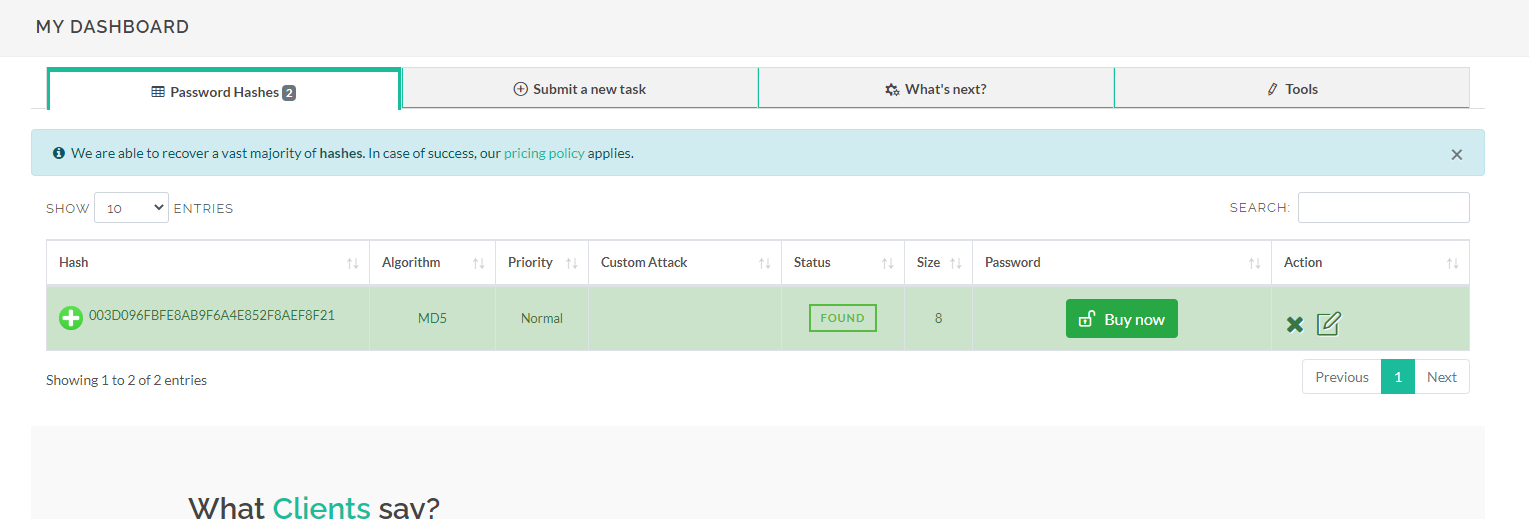
**4. 관리자 페이지 및 관리자 패스워드 유추**

패스워드 크랙 취약점이란 적절하지 않은 패스워드 규칙에 의한 취약한 패스워드를 공격자가 무작위 대입, 사전 공격 등의 기법으로 알아내는 것을 말한다.

관리자 페이지 유출은 관리자 페이지가 공격자가 예상하기 쉬운 이름(admin, \_admin, admin\_, admin\_page 등)으로 지정되어 공격자가 관리자 페이지를 쉽게 예상할 수 있는 취약점을 말한다.

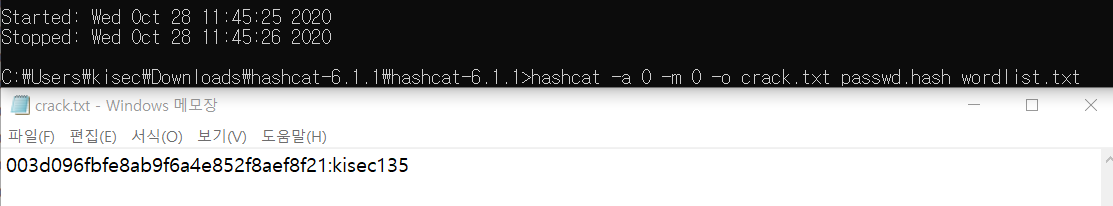


<그림 4.1> 어떤 해시로 암호화 되어있는지 확인 (Online Hash Identifier)

먼저 <그림 4.1>과 같이 해당 해시 암호가 어떤 알고리즘으로 되어있는지 확인해보았다. 여러 해시 알고리즘이 추측되지만 구글검색을 해보니 MySQL에선 해시 암호화시 MD5를 사용한다고 하여 우선 MD5라 추측하였다. 

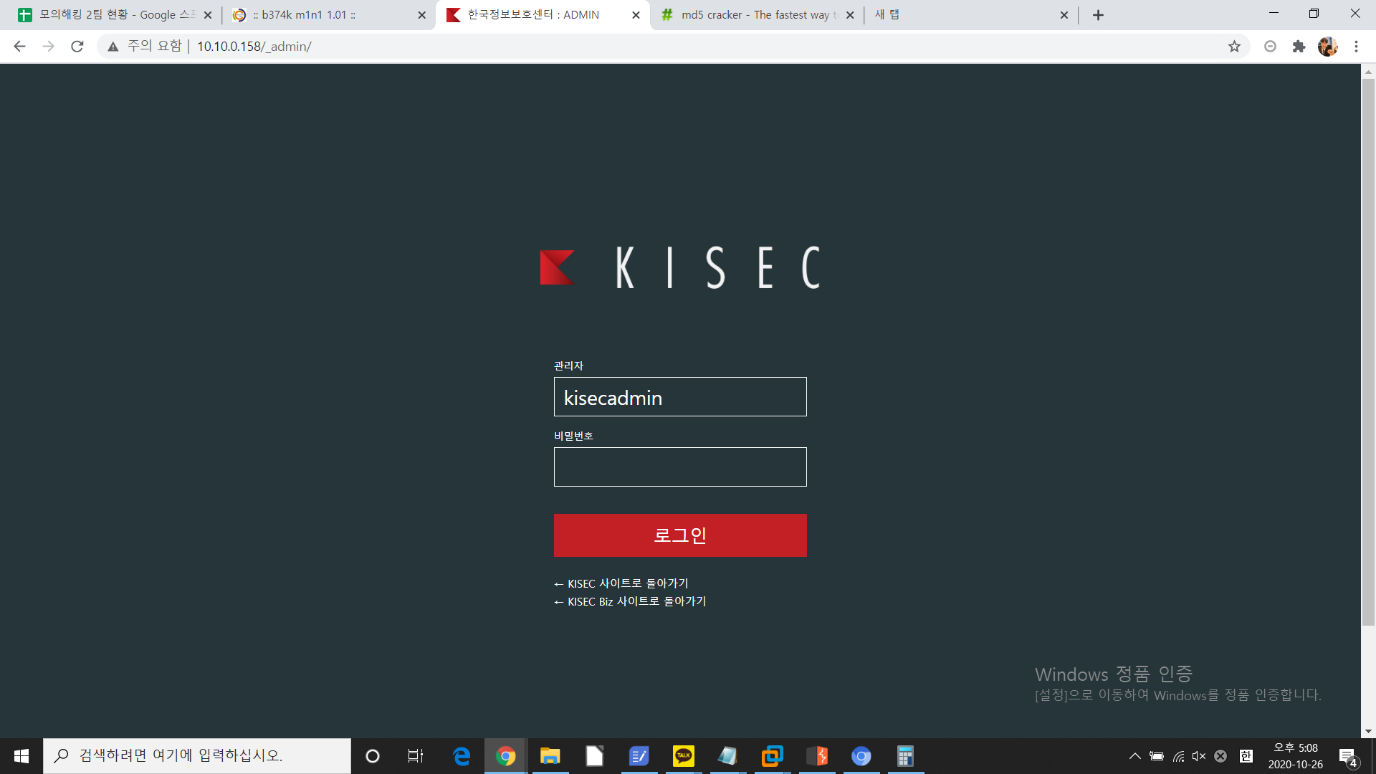
<그림 4.2> Online Password Hash Crack에서 암호화된 해시 값 크랙

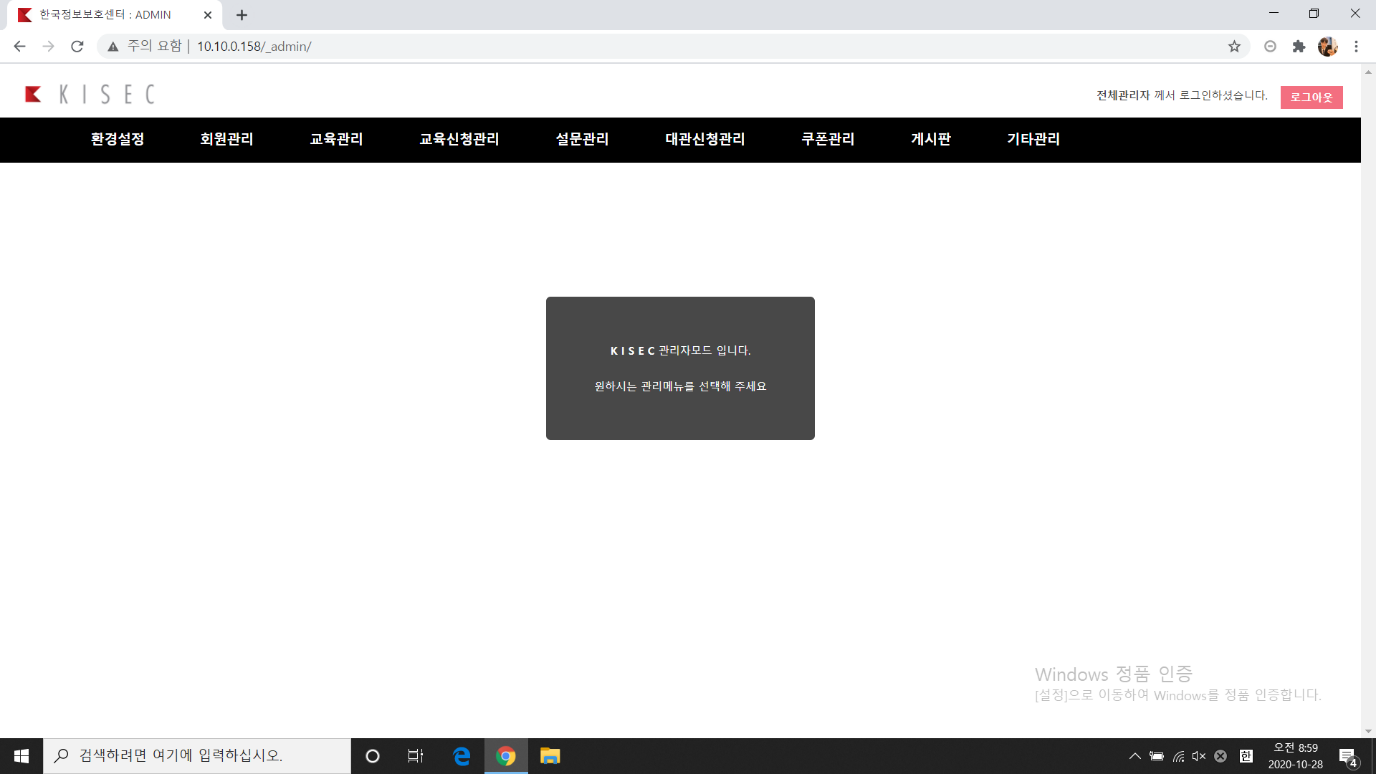
<그림 4.2>의 Online Passwd Hash Crack 사이트에서 크랙을 해본 결과 정확한 패스워드의 값을 알기 위해서는 추가적인 비용이 발생하나 패스워드가 8자리라는 것은 알아냈다. 혹시 유추하기 쉽게 kisec+숫자 세 자리 또는 숫자 세 자리+kisec 일 가능성이 있다고 생각하여 해당 조합으로 사전파일을 만든 후에 hashcat으로 크랙을 시도해보았다.



<그림 4.3> hashcat을 이용하여 admin 패스워드 해시 값 크랙

그림 <4.3>에서 보면 크랙을 시도해보니 비밀번호가 kisec135인 것을 알아냈다. 이제 admin 페이지를 유추해야 한다.



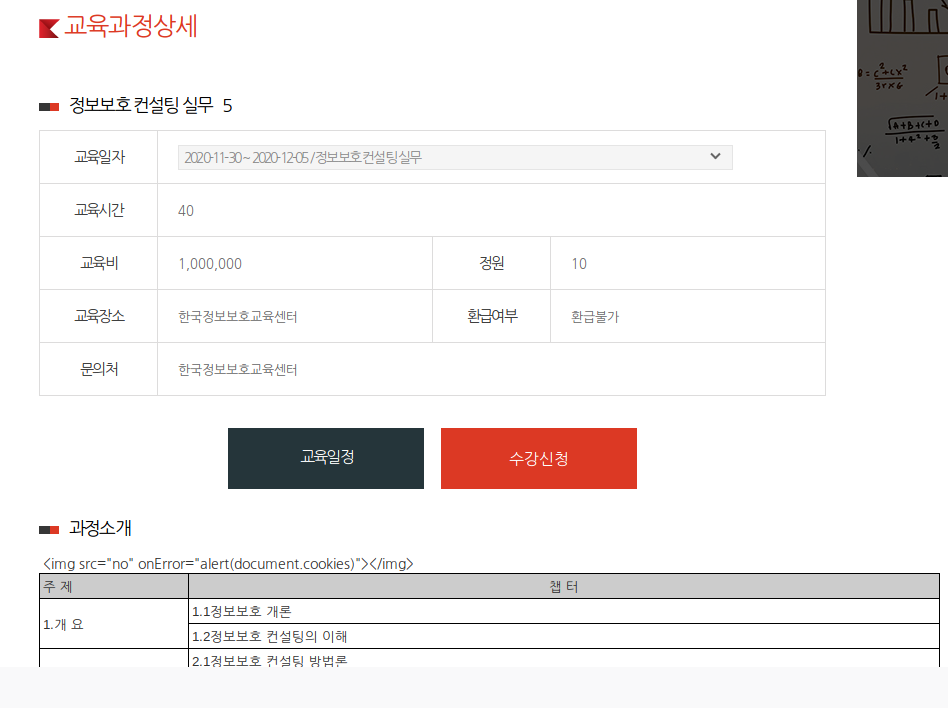
<그림 4.4> 홈페이지 admin 페이지 발견

<그림 4.5> admin 로그인 성공

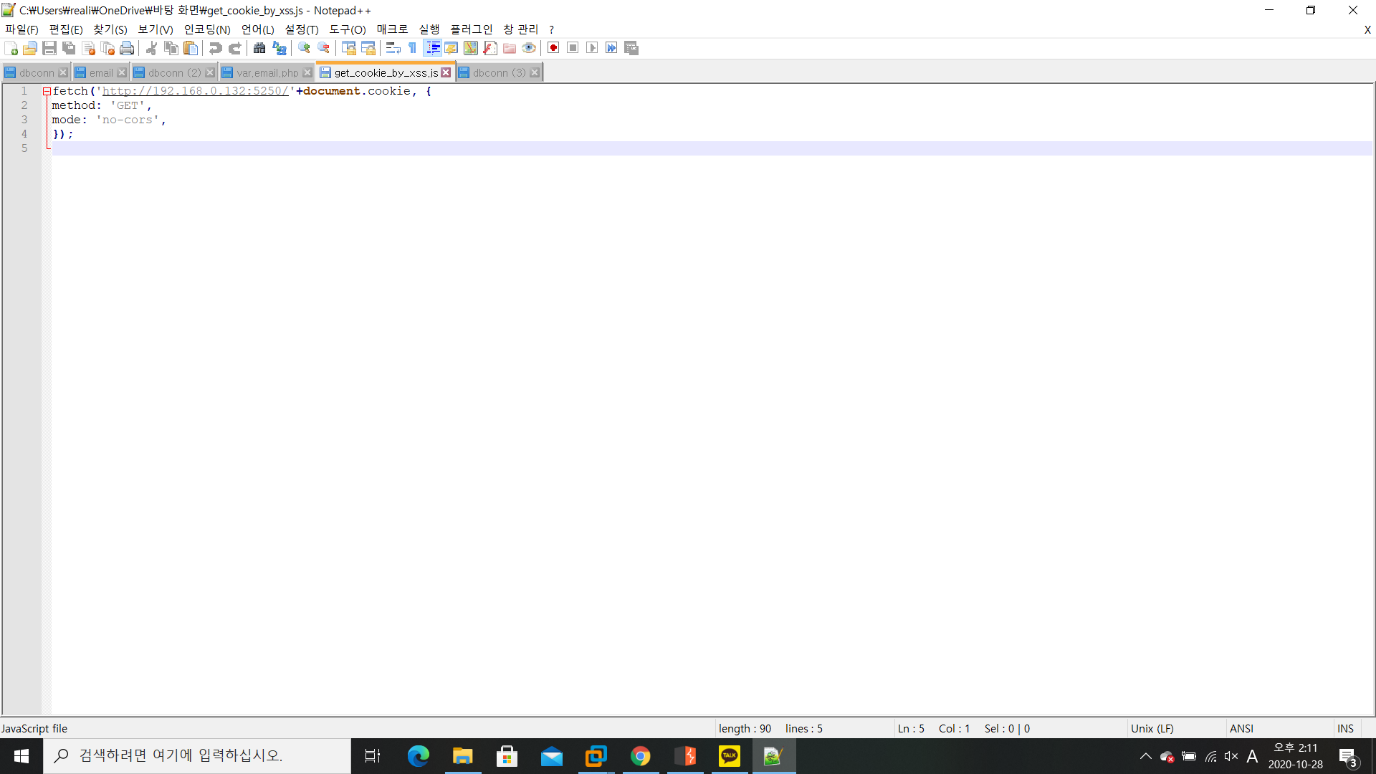
<그림 4.4>에서처럼 admin, admin\_page, \_admin 등으로 주소를 입력해보니 admin 페이지는 <http://10.10.0.158/_admin>이였다. Blind SQLi 및 패스워드 크랙으로 알아낸 아이디 비밀번호로 로그인해보니 admin 페이지 접속이 가능하였다. (<그림 4.5>) 여기서 나는 admin 권한으로 두가지 추가적인 공격 시나리오를 구성하였다.

**5.1 XSS를 이용한 쿠키 탈취로 로그인 세션 취득**

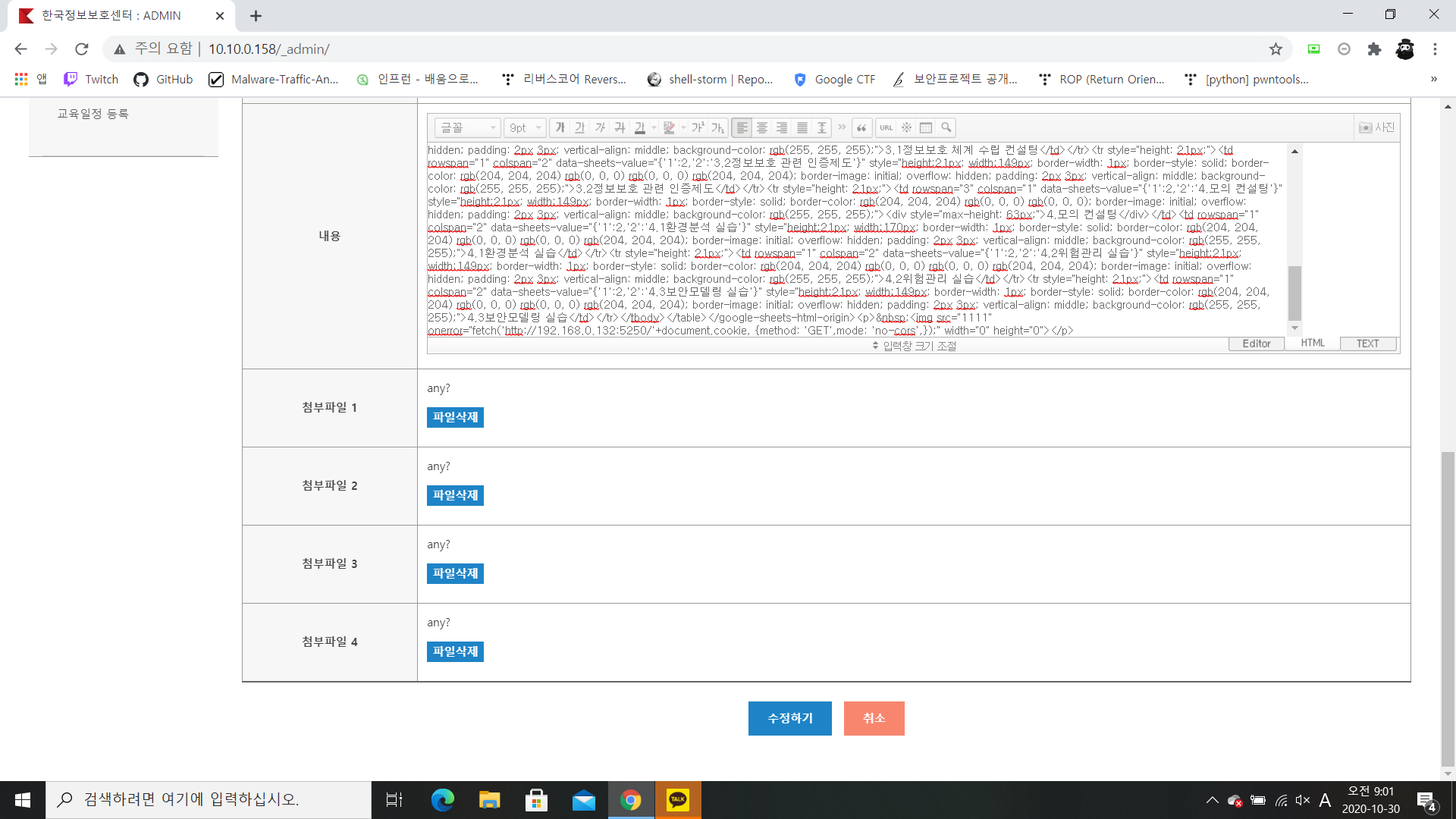
Cross Site Scripting(이하 XSS)취약점이란 사용자가 입력할 수 있는 기능이 있는 페이지에서 특정 HTML 태그를 필터링 하지 않아 공격자가 스크립트를 입력하여 원하는 기능을 수행하게 하는 공격을 말한다.



<그림 5.1.1> XSS Check(그림 재 캡쳐 필요)

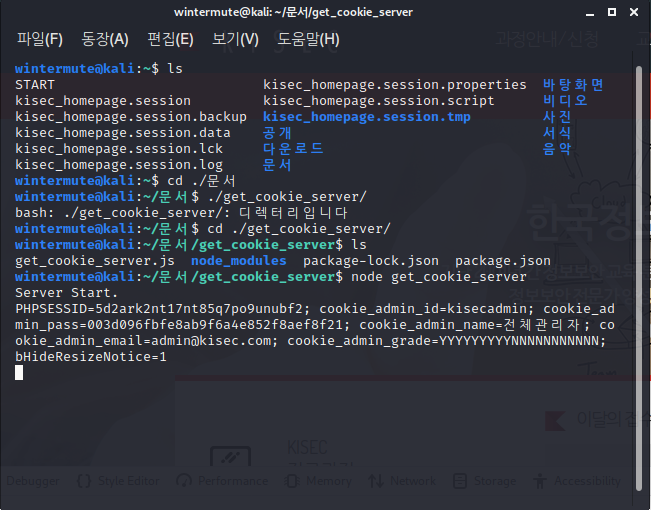
Admin 권한을 통해 교육과정상세 페이지를 수정할 수 있었으며 수정 시 몇몇 XSS 확인 코드를 사용해보니, onerror를 이용한 XSS는 가능한 것이 확인되었다. 그래서 쿠키 값을 서버(Kali Linux)로 보내는 코드를 통해 쿠키 값을 탈취해보았다. 사용된 스크립트는 <그림 5.1.2>를 참조하라.

<그림 5.1.2> 쿠키 탈취에 사용된 스크립트



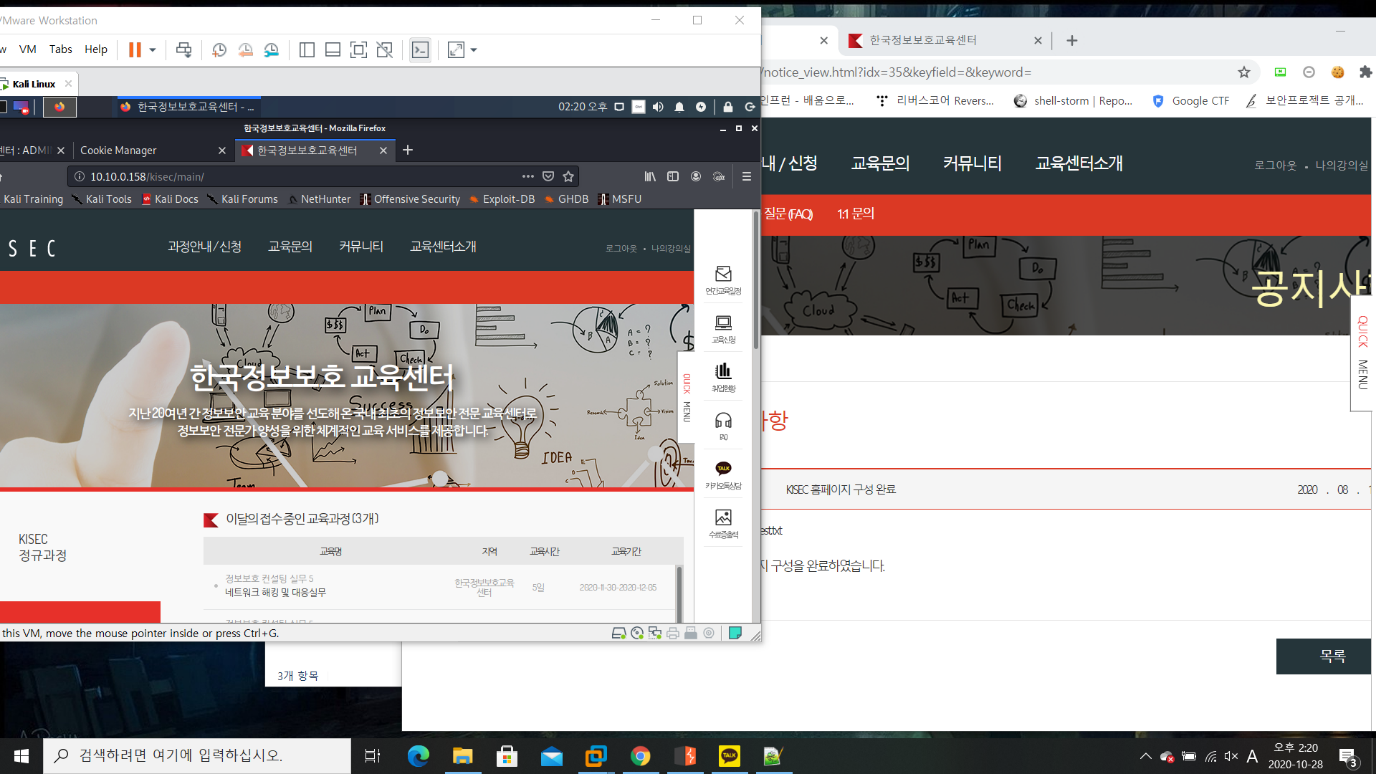
<그림 5.1.3> 교육과정상세 페이지에 스크립트 삽입

그림 <5.1.3>처럼 스크립트를 삽입하였다.



<그림 5.1.4> 서버(Kali Linux)에서 탈취한 쿠키 값

해당 교육과정페이지를 접속한 피해자(Windows 10)의 쿠키가 공격자의 서버(Kail Linux)로 보내진 모습이다. 또한 피해자 PC로 관리자 페이지 접속해서 그런지 admin 관련 쿠키 값도 탈취된 것으로 보인다.

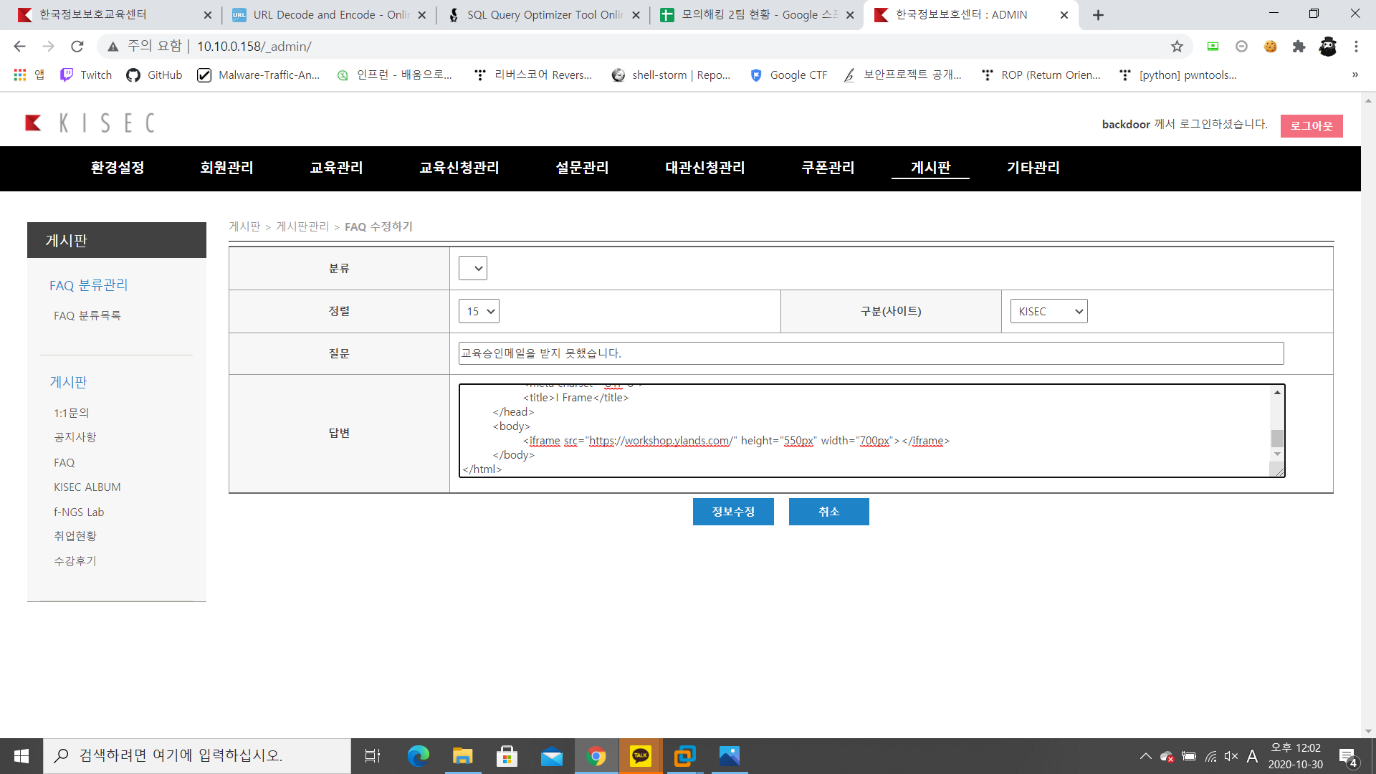


<그림 5.1.5>탈취한 쿠키 값으로 쿠키 변조 후 세션 취득 성공

탈취한 쿠키 값(<그림 5.1.4>의 쿠키 중 PHPSESSID)로 변조해보니 공격자PC(Kali Linux, 좌측 작은 창)가 피해자의 PC(Windows 10, 우측 큰 창)로 로그인 되는 것을 확인하였다.

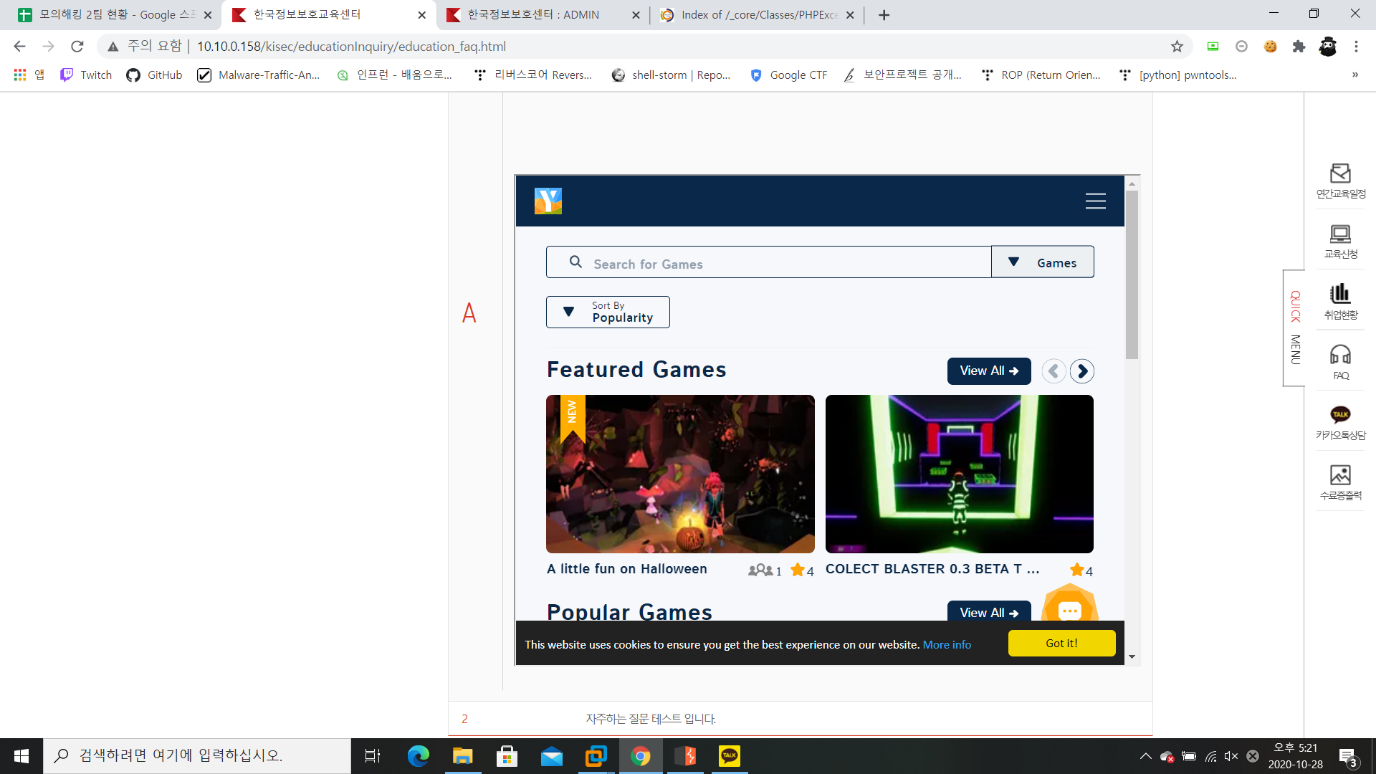
**5.2 iframe 태그 삽입으로 광고 사이트 출력**

악성 콘텐츠 삽입 취약점이란 사용자가 입력할 수 있는 기능이 있는 페이지에서 특정 HTML 태그(특히 iframe)를 필터링하지 않아 공격자가 iframe 태그 등을 이용하여 광고성 사이트 또는 악성코드를 실행할 수 있는 취약점을 말한다.

****

<그림 5.2.1> FAQ페이지에 Iframe 태그 삽입>

<그림 5.2.1> 구글에서 광고성 페이지를 가져오는 iframe이 포함된 HTML코드를 통해 FAQ 페이지를 수정하여 iframe 태그가 실행이 되는지 테스트해보았다.



<그림 5.2.2>FAQ 페이지에서 iframe에서 삽입한 광고성 페이지 출력 확인

<그림 5.2.2> 광고성 페이지가 출력되는 것을 확인하였다. 이는 iframe에 악성코드가 실행될 가능성이 있다는 것이다.

**6. 총평**

전체 몇 개 취약점이 발견되었으며 영향도 상(H): 7개 중(M): 3개 하(L): 0개 총 10개의 취약점이 도출되었다. 주요 취약점은 아래와 같다.

**[SQL Injection]**

타겟이 되는 KISEC 홈페이지에서 로그인 우회 점검, 에러 기반 점검, 문자열 기반 점검 등을 수행하였으며 그 중 공지사항 및 커뮤니티 게시판에서 사용되는 idx 매개변수에서 URL Encoding및 스페이스를 나타내는 ‘+’ 문자를 필터링하지않아 Blind SQL Injection 취약점이 발견되었다. 또한 취약점을 이용하여 admin 계정을 탈취하여 관리자페이지에 접속하는 데 성공하였으며 세션 탈취 및 악성코드 삽입 등의 다른 공격을 수행할 수 있는 Attack Vector로 사용되었으므로 해당 취약점은 [H]에 해당된다.

**[XSS]**

타겟이 되는 KISEC 홈페이지에서 사용자가 입력할 수 있는 게시판 및 관리자 페이지에서 수정할 수 있는

페이지(교육일정상세안내 페이지, FAQ 등)에서 XSS가 발생하였고 이를 통해 교육일정상세안내 페이지에서 공격자 서버에 쿠키를 보내도록 하는 스크립트를 통해 쿠키 값 탈취에 성공하였으므로 해당 취약점은 [M]에 해당된다.

**[Directory Indexing]**

타겟이 되는 KISEC 홈페이지의 경우 Directory Indexing이 되는 지 여러 페이지를 점검하였고, 페이지 중/\_core페이지는 Directory Indexing 취약점이 있어 File Download 취약점을 유추하는데 사용되었으므로 해당 취약점은 [M]에 해당된다.

**[File Download]**

타겟이 되는 KISEC 홈페이지의 경우 /\_core/download.php를 통하여 파일을 다운로드하는데 성공하였고, 이를 통해 DB의 테이블 정보를 누출하는데 성공하였으므로 해당 취약점은 [H]에 해당된다.

**[인증 및 세션 관리]**

타겟이 되는 KISEC 홈페이지의 경우 쿠키로 세션을 유지하고 있으며 쿠키 만료일도 1년 이상이었기 때문에 XSS 공격을 통해 탈취한 쿠키로 로그인 세션을 취득하는데 성공하였으므로 해당 취약점은 [H]에 해당된다.

**[홈페이지 관리자 및 사용자 계정 절취 가능]**

타겟이 되는 KISEC 홈페이지의 경우 위의 XSS 및 SQL Injection을 통해서 관리자의 계정 및 비밀번호를 알아 낼 수 있었으며, 관리자가 취약한 패스워드를 사용하고 있었기 때문에 쉽게 크랙이 가능하였으며 또한 쿠키 값을 탈취하여 사용자 로그인 세션을 절취하였으므로 해당 취약점은 [H]에 해당된다.

**[매개변수 변조(URL/파라미터 변조)]**

타겟이 되는 KISEC 홈페이지의 경우 URL 요청 시 매개변수를 서버 측에서 필터링하지않아 프록시를 통해 쉽게 매개변수 변조가 가능하였고 Directory Indexing이 발생하는 /\_core 폴더의 경우 웹 서비스를 구성하는 중요파일들이 존재하는 폴더임에도 불구하고 사용자가 쉽게 접근할 수 있었기 때문에 해당 취약점은 [M]에 해당된다.

**[웹, 소스코드 등에 중요정보 평문 저장]**

타겟이 되는 KISEC 홈페이지의 경우 File Download 취약점을 이용한 파일들에 사용되는 DB의 테이블의 정보가 그대로 누출되어 있으므로 해당 취약점은 [M]에 해당된다.

**[악성 콘텐츠]**

타겟이 되는 KISEC 홈페이지의 경우 FAQ페이지에서 iframe 태그를 필터링하지 않아 공격자가 광고성 사이트 삽입 및 악성코드를 실행시킬 수 있으므로 해당 취약점은 [H]에 해당된다.

**[관리자페이지 노출]**

타겟이 되는 KISEC 홈페이지의 경우 관리자페이지가 \_admin이므로 쉽게 유추할 수 있었으며, 또한 관리자의 쿠키를 가지고 있거나 관리 아이디와 패스워드를 탈취했을 때 쉽게 관리자 페이지를 접근할 수 있으므로 해당 취약점은 [H]에 해당된다.

**[쿠키 변조]**

타겟이 되는 KISEC 홈페이지의 경우 쿠키로 세션을 유지하고 있기 때문에 공격자가 쿠키 변조를 통해 타 사용자 및 관리자의 로그인 세션을 취득할 수 있으므로 해당 취약점은 [H]에 해당된다.

**7. 대응 방안**

[**SQL Injection**]

1. Information\_schema를 조회할 수 없도록 최소 권한 유저로 DB를 사용한다

2. 사이트 내에 사용하지 않는 SQL 함수 (ascii 등)를 사용하지 않도록 한다.

3.필터링 우회에 사용될 수 있는 문자(URL Encoding, space bar 우회 문자 +)들도 필터링을 해

야 한다.

**[XSS]**

1. 사용자가 입력할 수 있는 값을 제한한다. 예를 들어 idx 매개변수가 존재하는 페이지의 경우 숫자 값만 입력 받도록 구현한다.
2. <script>태그만 필터링 하는게 아니라 XSS를 우회 할 수 있는 URL Encoding, "javascript" 문자열 및 on\* 등의 문자열도 필터링 해야 한다.

[**Directory Indexing**]

1. 디렉터리 인덱싱 관련된 설정을 차단한다. 해당 시스템의 경우 Apache 서버이므

httpd.conf에서 Options Indexes FollowSymLinks MultiViews 옵션을 Options

FollowSymLinks MultiViews로 변경한다.

1. 디렉터리 인덱싱이 필요하다면 인덱싱이 필요한 디렉터리의 이름을 유추할 수 없는 이름을 사용한다.

**[File Download]**

1. 파일 다운로드 받는 기능 중 파일 경로 입력 시 상위 폴더 이동 문자(../)및 우회 문자

(.%2e,%2f, %5c 등)를 필터링 해야 한다.

1. 파일 다운로드 받는 기능에서 파일경로 및 파일이름 노출을 암호화 등으로 최소화하여야 한다.
2. 해당 서버가 PHP 언어로 개발되었으므로 php.ini 내용 중 magic\_guotes\_gpc 항목의 값을 ON으로 하여 ".\나 .../" 값 입력을 치환해야 한다.

**[인증 및 세션 관리 & 홈페이지 관리자 및 사용자 계정 절취 가능]**

1. 회원정보 수정 또는 구매와 같은 기능을 하는 페이지에서 다른 사용자와 구분을 단순한 아

이디나 쿠키 값으로 구분하지 않아야 한다.

1. 아이디나 패스워드 등의 인자 값 변경으로 다른 사용자의 페이지에 접근할 수 있는지 확인하고 접근할 수 있다면 접근 할 수 없도록 수정해야한다.

**[매개변수 변조(URL/파라미터 변조) & 쿠키 변조]**

1. 사용자의 세션마다 추측하기 어려운 임의의 난수 값을 지정하여 통신하고 서버 측에서 난수 값을 확인하여 사용자의 요청을 식별해야 한다.
2. 쿠키를 이용하여 세션을 유지하는 것이 아니라 세션 값을 이용하여 세션을 유지하도록 하여야 한다.

**[웹, 소스코드 등에 중요정보 평문 저장]**

1. HTTP가 아닌 HTTPS를 사용하여야 하며 전자서명 인증서를 사용하여야 한다.

2. 데이터 전송 시 통신 채널 또는 패킷의 암호화를 수행하여야 한다.

3. 소스코드의 경우 난독화 혹은 암호화하여 저장하여야 한다.

**[악성 콘텐츠]**

1. iframe 태그를 필터링하거나 http 헤더에 X-Frame-Options 헤더를 Deny하여 iframe 태그를 작동하지 않도록 하거나 또는 SameOrigin으로 설정하여 사이트 내 소스를 가진 iframe만 동작하게 한다.

**[관리자페이지 노출]**

1. 관리자 페이지가 쉽게 유추 가능하지 않은 경로를 사용하거나 디폴트 관리자 페이지 설정을

변경하여야 한다.

1. GET, POST, OPTIONS Http 메소드 만을 웹페이지에서 사용하여야 한다.
2. 관리자 페이지 접근을 외부 IP가 접근하지 않도록 하여야 한다.
3. 내부에서도 관리자 권한이 없는 사용자는 접근하지 않도록 해야 한다.