웹 해킹 보안의 종류와 대응 방안

지도교수 : 김 윤
연 구 자 : 정 민 서

< 목 차 >

요 약

웹은 사용하지 않는 사람을 찾기 더 힘들 정도로 많은 사람들이 사용하고 있으며 사용량이 많은 만큼 다양한 볼거리와 정보, 큰 도움을 주는 우리 생활에 필수 불가결한 요소가 되었다. 하지만 웹이 발전함에 따라 해킹하고자 하는 공격자들의 시도도 다양하게 늘어나고 있어 이에 대한 대처 방안이 요구되고 있다. 이 논문에서는 다양한 웹 해킹 사례를 통해 이에 따른 적절한 대응 수단을 각 사례별로 상세히 논의하고자 한다.

주요어 : 웹 해킹, SQL Injection, XXS, XXE

1. 서론

1.1 웹이란

웹툰, 웹하드, 웹드라마 등 현대에 들어 자주 쓰이고 있는 접두어 '웹(web)'은 '동영상이나 음성 따위의 각종 멀티미디어를 이용하는 인터넷을 이르는 말'이다.

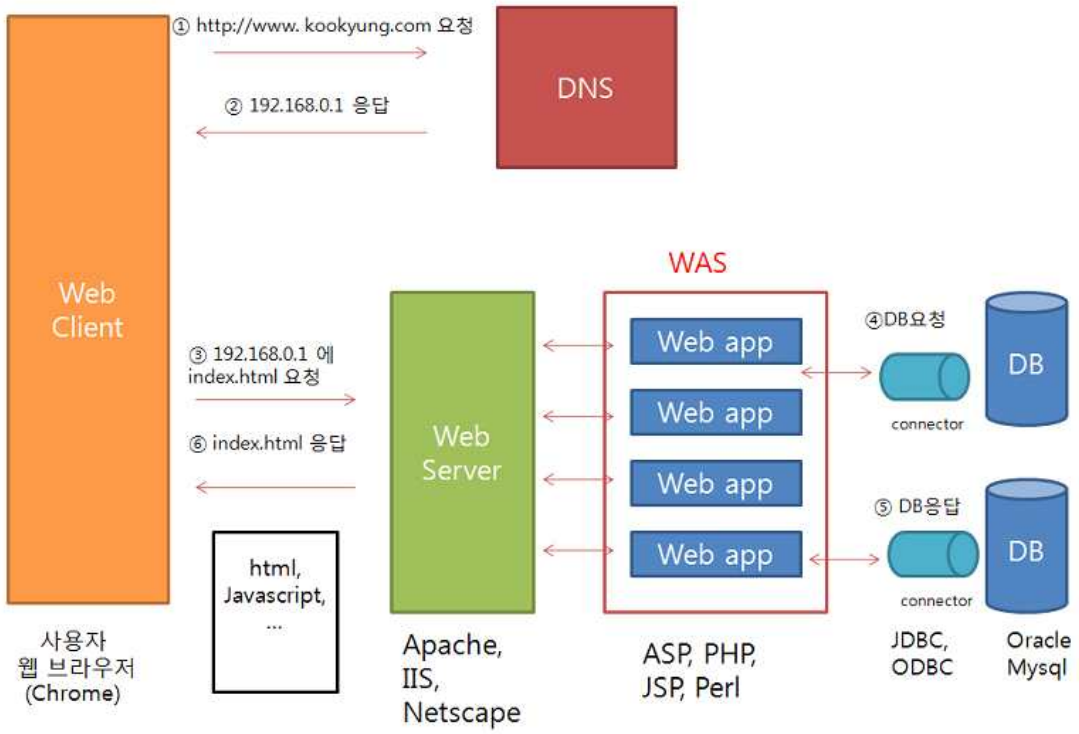
1989년 팀 버너스 리가 웹을 개발한 이후, 그동안 웹은 인터넷의 확산과 모바일 기기의 폭발적 증가로 전세계에 확산되었고 이제는 웹이 없는 삶을 상상할 수 없는 상황까지 발전하게 되었다.

수많은 사람들이 이용하고, 보다 많은 정보가 오가는 공간이니만큼 웹을 노리는 해커들과 해킹 기술들도 몹시 많다. 해킹을 당하지 않으려면 해킹 방법을 알아야 하고, 해킹 방법을 알기 위해서는 웹의 동작 원리를 알아야 할 것이다.

그러므로 해킹 방법을 설명하기 전 웹의 동작 원리와 기능을 설명하고자 한다.

1.2 웹의 구조와 동작 원리

1.2.1 구조



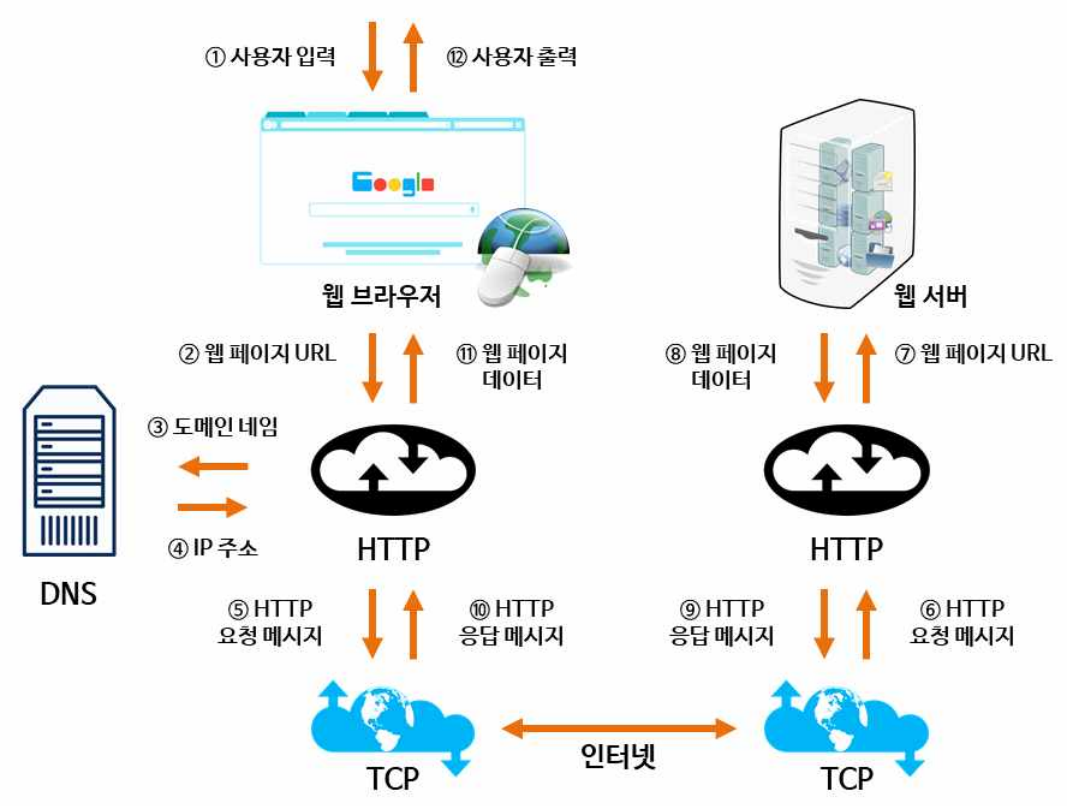
가. [사진 1] 웹의 구조

유저(User)가 1번과 같이 웹 브라우저를 통해 www.naver.com 에 접속을 요청한다고 하자. DNS(Domain Name System) 서버로 도메인 이름이 보내지면 DNS는 이를 IP주소(예: 192.0.2.44)로 변환한다.

그다음, 웹 브라우저는 Web Server에 IP주소로 접속 요청을 보내고, 그 요청은

WAS를 거쳐 DB에서 요청된 파일을 가져온다. 그리고 그 파일은 다시 WAS를 거쳐 웹 서버를 통해 웹브라우저에게 전달된다. 이로써 우리는 웹 브라우저상에 네이버 화면을 볼 수 있다.

1.2.2 동작 원리



나. [사진 2] 웹의 동작 원리

①② 사용자가 웹 브라우저를 통해 찾고 싶은 웹 페이지의 URL 주소를 입력한다.

③ 사용자가 입력한 URL 주소 중에서 도메인 네임(domain name) 부분을 DNS 서버에서 검색한다.

④ DNS 서버에서 해당 도메인 네임에 해당하는 IP 주소를 찾아 사용자가 입력한 URL 정보와 함께 전달한다.

⑤⑥ 웹 페이지 URL 정보와 전달받은 IP 주소는 HTTP 프로토콜을 사용하여 HTTP 요청 메시지를 생성한다. 이렇게 생성된 HTTP 요청 메시지는 TCP 프로토콜을 사용하여 인터넷을 거쳐 해당 IP 주소의 컴퓨터로 전송한다.

⑦ 위의 과정을 거쳐 도착한 HTTP 요청 메시지는 HTTP 프로토콜을 사용하여 웹 페이지 URL 정보로 변환된다.

⑧ 웹 서버는 도착한 웹 페이지 URL 정보에 해당하는 데이터를 검색한다.

⑨⑩ 검색된 웹 페이지 데이터는 또다시 HTTP 프로토콜을 사용하여 HTTP 응답 메시지를 생성한다. 이렇게 생성된 HTTP 응답 메시지는 TCP 프로토콜을 사용하여 인터

넷을 거쳐 원래 컴퓨터로 전송된다.

⑪ 도착한 HTTP 응답 메시지는 HTTP 프로토콜을 사용하여 웹 페이지 데이터로 변환된다.

⑫ 변환된 웹 페이지 데이터는 웹 브라우저에 의해 출력되어 사용자가 볼 수 있게 된다.

이러한 과정을 거쳐 웹이 동작한다고 볼 수 있다.

2. 본론

2.1 웹 해킹이란

웹 애플리케이션 해킹(Web Application Hacking) 이라고도 부르고 웹 서비스상에서 발생할 수 있는 모든 보안 허점(Security Hole)을 이용해 악의적인 행위를 하는 것을 통칭한다.

2.2 웹 해킹의 종류

2.2.1 SQL injection

보안상의 취약점을 이용하여, 임의의 SQL 문을 주입하고 실행되게 하여 데이터베이스가 비정상적인 동작을 하도록 조작하는 행위이다. OWASP[1] Top10 중 첫 번째에 속해 있으며, 공격이 비교적 쉬운 편이고 공격에 성공할 경우 큰 피해를 입힐 수 있는 방법이다. 대표적인 예로 2017년 3월, '여기어때'라는 숙박 어플의 대규모 개인정보 유출 사건이 SQL Injection으로 인한 피해이다.

1) 논리적 에러를 이용한 공격

```
select * from client where name='anjinma' and password='12345'
                                ↓
select * from client where name='anjinma' and password=' or '1'='1 --
```

다. [표 1] SQL injection 기법 중 논리적 에러를 이용한 로그인 예시

OR 1=1 라는 구문을 넣어 1=1은 참, OR은 둘 중 하나라도 참이면 참이므로 이 구문은 참이 되어 로그인에 성공하게 된다.

1) 오픈소스 웹 애플리케이션 보안 프로젝트. 주로 웹에 관한 정보노출, 악성 파일 및 스크립트, 보안 취약점 등을 연구하며 10가지의 **웹 애플리케이션 취약점**인 OWASP TOP 10을 발표하였다.

2) Union 명령어를 이용한 공격

[외부 입력] ID: test' UNION SELECT 1,1 -- PW: anything [실행 쿼리] SELECT * FROM users WHERE ID='test' **UNION** SELECT 1,1 -- and PW='anything'

라. [표 2] SQL injection 기법 중 union 명령어를 이용한 로그인 예시

위 UNION 명령어는 여러 개의 SQL문을 합쳐 하나의 SQL문으로 만들어주는 방법이다. 진하게 표시한 UNION에는 두 가지 종류의 값을 넣을 수 있는데 그것은 다음과 같다.

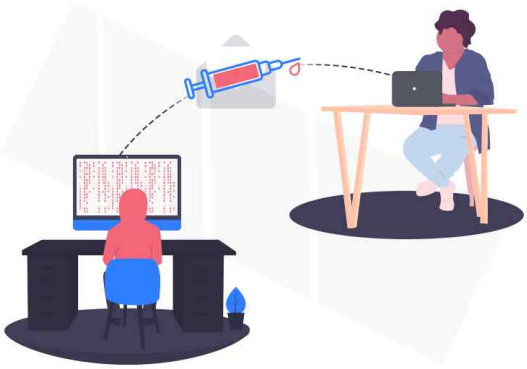
(1) UNION

출력되는 값 중 중복 값 제외

(2) UNION ALL

출력되는 값 중 중복 값 제외하지 않고 전체를 합침

2.2.2 Command injection



마. [사진 3] Command injection

웹 애플리케이션에서 시스템 명령을 사용할 때, 세미콜론 혹은 &, && 를 사용하여 하나의 Command를 Injection 하여 두 개의 명령어가 실행되게 하는 공격이다. 두 개의 명령을 한 줄에 동시에 실행할 수 있도록 하는 연결자인 “세미콜론”을 사용하여 ‘ifconfig’ 명령을 실행시키기 위해 웹 페이지에 요청하고, 웹 서버는 사용자로부터의 요청을 아무런 검증 없이 서버에 전송하게 되면 서버의 민감 정보를 그대로 노출하게 되는 것이다.

```
int main(char* argc, char** argv) {

char cmd[CMD_MAX] = "/usr/bin/cat ";

strcat(cmd, argv[1]);

system(cmd);

}
```

[사진 4] Command injection 예시 코드 1

[사진 4]는 원격 사용자가 파일을 수정하거나 삭제할 수 없는 상태에서 파일의 내용을 볼 수 있도록 하는 코드의 예시다. 프로그램은 루트 권한으로 실행하며, 프로그램은 파일에 대한 읽기 전용 액세스만 가능하므로 이 사용자가 무해하다고 판단하지만 이 코드는 Command injection을 가능하게 한다.

```
";rm -rf /"
```

[사진 5] Command injection 예시 코드 2

공격자가 파일 이름 대신 [사진 5] 같은 문자열을 입력하면 system()의 호출은 실행에 실패하고 운영 체제는 루트 디스크 파티션의 재귀 삭제를 수행한다.

2.2.3 Cross Site Request Forgery (CSRF)

```
<body onload="document.forms[0].submit()">
  <form action="http://netbank.com/transfer.do" method="POST">
    <input type="hidden" name="acct" value="AttackerA"/>
    <input type="hidden" name="amount" value="$100"/>
    <input type="submit" value="View my pictures!"/>
  </form>
</body>
```

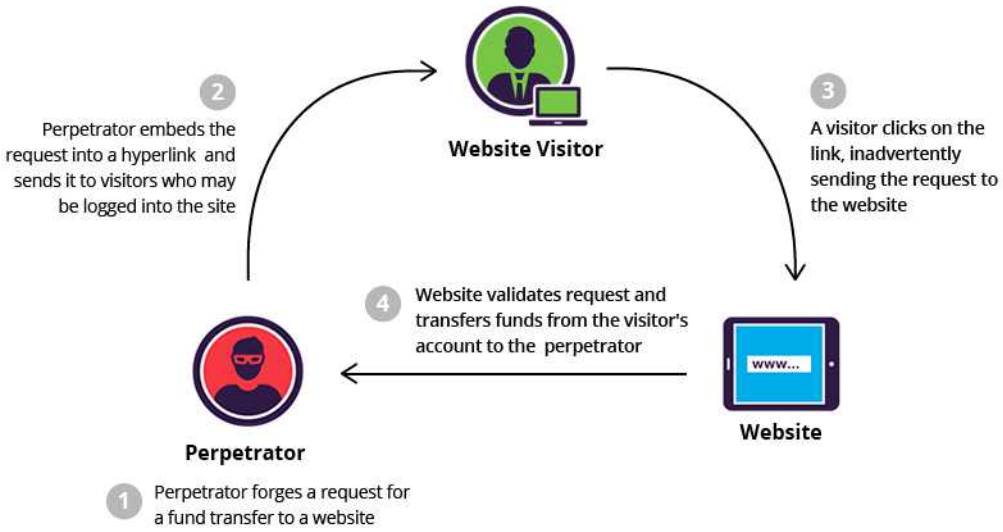
[사진 6] CSRF 기법의 예시 코드

CSRF는 웹 브라우저를 속여 공격자가 게시판에 악의적인 스크립트가 담긴 게시글을 삽입하고 해당 게시글을 클릭할 경우 사용자의 인증 권한을 이용하여 의도하지 않은 요청이 서버에 전송되는 공격기법이다.

일반적으로 이메일이나 링크와 같은 사회 공학적 해킹을 사용하여 피해자를 속여 서버에 위조된 요청을 보내도록 유도하며 사용자는 공격 시 애플리케이션에 의해 인증되기 때문에 서버는 합법적인 요청과 위조된 요청을 구별하는 것이 불가능하다.

이로 인해 계정의 이메일 주소나 비밀번호를 변경할 수 있으며, 계좌 이체도 가능하다. 따라서 공격자는 탈취한 사용자의 계정을 통해 사용자의 권한이 닿는 모든 데이터와 기능을 제어할 수 있다. 이런 CSRF 기법이 성공하려면 세 가지 주요 조건이 충족되어야 한다.

첫 번째. 공격을 유도할만한 행위가 있는지
두 번째. 쿠키 기반의 세션 처리인지
세 번째. 예측할 수 없는 요청 매개변수인지



[사진 7] CSRF 기법 간단 설명

2.2.4 Cross Site Script(XSS)

관리자가 아닌 권한이 없는 사용자가 웹 사이트에 스크립트를 삽입하는 공격 기법이다. XSS공격 역시 OWASP Top10에 포함되어 있다. 대부분 사용자가 글을 쓰고 읽을 수 있는 게시판에 많이 발생하지만, 사용자의 입력값을 웹 페이지에 보여주는 곳에서도 발생한다. 악의적인 사용자가 C&C 서버로 리다이렉션하기 위해 리다이렉션 스크립트를 주입하여 중간 경유지로 활용하기도 하고, 사용자의 쿠키를 탈취하여 세션 하이재킹(Session Hijacking) 공격을 수행하기도 한다.

1) 지속형 or 저장형 XSS 공격

지속적으로 피해를 입히는 XSS 공격인데, 해커는 웹 애플리케이션에서 XSS 취약점이 있는 곳을 파악하고, 악성스크립트를 삽입한다. 삽입된 스크립트는 데이터베이스에

저장이 되고, 저장된 악성스크립트가 있는 게시글 등을 열람한 사용자들은 악성스크립트가 작동하면서 쿠키를 탈취당한다던가, 혹은 다른 사이트로 리다이렉션 되는 공격을 받게 된다. 데이터베이스에 저장이 되어 지속적으로 공격한다고 하여 Persistent XSS 라고 부르며, 데이터베이스에 저장이 되므로 Stored XSS 공격이라고 부르기도 한다. 한 번의 공격으로 악성스크립트를 삽입하여 수많은 피해를 입힐 수 있다는 특징이 있다.

2) 반사형 XSS 공격

Reflected XSS 공격은 사용자에게 입력받은 값을 서버에서 되돌려 주는 곳에서 발생한다. 예를 들면 사용자에게 입력받은 검색어를 그대로 보여주는 곳이나 사용자가 입력한 값을 에러 메세지에 포함하여 보여주는 곳에 악성스크립트가 삽입되면, 서버가 사용자의 입력 값을 포함해 응답해 줄 때 스크립트가 실행된다. 보통 Reflected XSS 는 공격자가 악의적인 스크립트와 함께 URL을 사용자에게 누르도록 유도하고, URL을 누른 사용자는 악의적인 스크립트가 실행되면서 공격을 당하게 된다.

2.2.5 XML External Entity(XXE)

XML은 HTML의 한계를 극복하기 위해 만들어진 마크업 언어로 연결된 시스템끼리 데이터를 쉽게 주고받기 위한 언어다. 개중 XML 문서의 External Entity를 이용하여 공격하는 기법으로 공격자가 의도하는 외부 URL를 실행시키는 공격이다. 이 공격은 OWASP 4위에 등록되었다.

공격의 발생 원인은 여러 이유를 들 수 있으나

1) 애플리케이션이 직접 XML을 입력 받거나 특히 신뢰할 수 없는 곳의 XML를 업로드 하는 경우
2) XML 문서에 신뢰할 수 없는 데이터를 입력할 경우
3) 사용자의 입력값을 검증하지 않아 Document Type Definitions(DTD)가 구문이 실행되는 경우

로 간추릴 수 있다.

2.3 웹 해킹 대응 방안

2.3.1 SQL injection

1) 화이트리스트 기반으로 검증

블랙리스트 기반으로 검증하게 되면 수많은 차단 리스트를 등록해야 하고, 하나라도 빠지면 공격에 성공하게 되기 때문에 화이트리스트를 사용하는 것이 좋다.

2) Prepared Statement 구문 사용

사용자의 입력값이 데이터베이스의 파라미터로 들어가기 전에 DBMS가 미리 컴파일하여 실행하지 않고 대기한다. 그 후 사용자의 입력값을 문자열로 인식하게 하여 공격 쿼리가 들어간다고 하더라도, 사용자의 입력은 이미 의미 없는 단순 문자열이기 때문에 전체 쿼리문도 공격자의 의도대로 작동하지 않는다.

3) Error Message 노출 금지

공격자가 SQL Injection을 수행하기 위해서는 데이터베이스의 정보(테이블 명, 컬럼 명 등)가 필요하다. 데이터베이스 에러 발생 시 따로 처리를 해주지 않았다면, 에러가 발생한 쿼리문과 함께 에러에 관한 내용을 반환해준다. 여기서 테이블 명 및 컬럼 명 그리고 쿼리문이 노출이 될 수 있기 때문에, 데이터베이스에 대한 오류 발생 시 사용자에게 보여줄 수 있는 페이지를 제작 혹은 메시지 박스를 띄우도록 하여야 한다.

4) 웹 방화벽 사용

웹 공격 방어에 특화되어있는 웹 방화벽을 사용하는 것이다.

웹 방화벽은 서버 내에 직접 설치하는 소프트웨어 형, 네트워크상에서 서버 앞 단에 직접 하드웨어 장비로 구성하는 하드웨어 형, DNS 서버 주소를 웹 방화벽으로 바꾸고 서버로 가는 트래픽이 웹 방화벽을 먼저 거치도록 하는 방법인 프록시 형이 있다.

2.3.2 Command injection

1) 직접적으로 시스템 명령어를 호출하지 않게끔 코딩한다.
2) 소스 코드 레벨에서는 직접적으로 명령어 실행 함수를 사용하지 않는 것이 좋다.
3) 프로그래밍 언어 및 라이브러리에서 자체적으로 제공하는 함수를 사용하는 것이 좋다.
4) 허용되지 않는 코드를 거부하기 위해 개발팀이 모든 허용 가능한 값의 화이트리스트를 작성한다.
5) 이미 공격당한 애플리케이션의 연결을 완전히 차단하여 다른 피해를 최소화한다.

2.3.3 Cross Site Script(XSS)의 경우

1) 입출력 값 검증

사용자가 입력한 값에 대한 검증과 사용자가 입력한 값을 그대로 출력할 때 검증이 필요하다. XSS Cheat Sheet 에 대한 필터 목록을 만들어 모든 Cheat Sheet에 대한 대응이 가능하도록 하여야 한다. XSS에 대한 필터링을 적용한 뒤 직접 테스트하여 스크립트가 실행되는지 모의해킹 해보는 것도 좋은 방법이다.

2) XSS 방어 라이브러리, 브라우저 확장 앱 사용

XSS를 막아주는 Anti XSS 라이브러리를 여러 회사에서 제공하는데 이 라이브러리를 사용하면 손쉽게 XSS를 방어할 수 있다. XSS 라이브러리를 사용하는 것은 서버 단에서 개발자가 추가하는 것이고, 사용자들이 각자 본인의 브라우저에서 악의적인 스크립트가 실행되지 않도록 방어하는 것이 중요하다. 방문하는 모든 사이트가 안전하다는 보장이 없기 때문에 브라우저 확장 앱 중 Anti XSS 를 해주는 애플리케이션을 설치하고 방어하여야 한다.

3) 웹 방화벽 사용

웹 방화벽은 웹 공격에 특화되어있기 때문에 XSS 공격을 방어하기 위함만이 아니라 각종 Injection 공격을 효과적으로 방어할 수 있다.

2.3.4 XML External Entity(XXE)의 경우

1) 근본적으로 entity 기능을 비활성화하면 XXE Injection은 불가능하다.
2) 모든 XML 프로세서와 라이브러리, SOAP을 SOAP 1.2나 그 이상으로 업그레이드
3) 서버에서 화이트 리스트를 이용한 입력값 검증, 필터링을 통한 XML 문서, 헤더, 노드에 있는 악성 데이터를 차단한다.
3) XML이나 XSL 파일 업로드 기능이 XSD 검증기 같은 것을 사용해서 XML이 유효한 내용인지 확인하고 검증한다.
4) 코드상 DOCTYPE 태그를 포함하는 입력을 차단하도록 입력 검증을 사용한다.

3. 결론

웹은 현대 사회에 있어서 떼려야 뗄 수 없는 도구이며 사용량과 정보가 늘어날수록 웹 해킹은 줄어들지 않고 더욱 기승을 부릴 것이다. 따라서 시간이 지날수록 결국 보안이 탄탄한 곳이 살아남게 될 것이며 공격은 허점을 노린 것이기 때문에 조금만 신경 쓴다면 생각보다 어렵지 않다. 따라서 이 논문에서 알아본 웹 해킹의 종류들과 각 해킹 별로 피해가 가지 않는 서버에서 실습 후 체득하게 된다면 대응 방안도 쉽게 숙지할 수 있을 것이고 그 지식을 잘 활용한다면 피해를 최소화하는 웹 개발자가 될 수 있을 것이다.

참고문헌

1) SQL Injection 이란? (SQL 삽입 공격), 2019 (https://noirstar.tistory.com/264)
2) 웹 해킹이란 무엇인가?, 2020 (https://peemangit.tistory.com/312)
3) 커맨드 인젝션 공격. 2020 (https://cybersecuritykong.tistory.com/10)
4) SQL INJECTION, 2019 (https://m.mkexdev.net/427)
5) Cross site request forgery (CSRF)
(attackhttps://www.imperva.com/learn/application-security/csrf-cross-site-request-forgery)
6) Web Parameter Tampering
(https://owasp.org/www-community/attacks/Web_Parameter_Tampering
7) https://rsy99.tistory.com/56
8) How to Prevent File Upload Vulnerabilities
(https://www.wordfence.com/learn/how-to-prevent-file-upload-vulnerabilities/)
9) https://www.i2sec.co.kr/ko/home/
10) https://hammieunseo.tistory.com/36
11) https://www.imperva.com/learn/application-security/parameter-tampering/
12) Anshul vyas, IDOR 공격, 2018
https://anshul-vyas380.medium.com/idor-attack-f2dd4aa6910d
13) 한국인터넷진흥원 & 인터넷 침해 대응 센터 - 파라미터 변조 취약점을 이용한 타인 개인정보 유출 사례 분석 및 대응방안, 2014
14) http://tcpschool.com/webbasic/works TCP school - 웹의 동작 원리
15) https://jaeseokim.tistory.com/29 - XXE Injection
16) https://inforsecdreamtree.tistory.com/33 - OWASP TOP 10