

INDEX

과제 1. 시큐어코딩에 대해 설명하시오 1

과제 2. 사례 기반 정보보호 용어 정의 및 설명 2

과제 3. 이러닝 시스템 위협 모델링 4

과제 4. 학생 활동 보고서 7

과제 1. 시큐어코딩에 대해 설명하시오.

시큐어코딩은 여러분들의 컴퓨터, 핸드폰 또는 자동차 등의 소프트웨어에 내포될 수 있는 보안의 취약점을 대비하는 코딩 기법을 말하고 보안을 고려하여 기능을 설계 및 구현하는 등 소프트웨어 개발 과정에서 지켜야 할 일련의 보안 활동을 말합니다.

소프트웨어 개발은 크게 5단계에 걸쳐서 이루어지게 됩니다. 이러한 5단계는 설계단계 → 코딩단계 → 통합단계 → 베타제품 → 제품출시로 이루어져있습니다. 아래의 표는 소프트웨어를 개발하는 과정에서 보안적인 문제가 발생했을 때 시큐어코딩을 적용하기 위해 사용되는 비용을 나타낸 표입니다.

구분	설계단계	구현단계	통합단계	베타제품	제품출시
설계과정	1배	5배	10배	15배	30배
코딩과정		1배	10배	20배	30배
통합과정			1배	10배	20배

표가 보여주다시피 시큐어코딩이 늦어질수록 발생하는 비용은 엄청나게 증가하게 됩니다. 따라서 소프트웨어 개발과정에서 미리 시큐어코딩으로 보안을 확보하는 것은 매우 중요하다고 할 수 있습니다.

지금부터는 어떠한 공격 때문에 시큐어코딩이 필요한지 예시를 들어 설명드리도록 하겠습니다.

여러분들이 특정 사이트에 로그인할 때 일어나는 보안 활동을 예시로 설명드리겠습니다. 여러분들이 특정 사이트에 로그인 시 여러분들의 네트워크상에는 여러분들의 ID, Password에 대한 정보가 기록되게 됩니다. 이때 여러분들의 정보에 대한 보안 활동이 일어나지 않는다면 여러분들의 정보를 탈취하고자 하는 공격자들이 여러분들의 정보를 탈취하는 것이 가능할 것입니다. 여러분들 혹은 주변사람들의 게임 ID를 해킹당하거나 해킹당했다는 얘기를 한 번쯤 들어보신 적이 있으실겁니다. 바로 이러한 일들이 제가 앞서 예시를 들어 설명한 상황과 같이 보안 활동이 일어나지 않았거나 그 보안이 부족해 공격자가 침투하여 여러분들의 정보를 탈취했기에 발생한 사건들이라고 할 수 있습니다.

이와 같이 서비스를 이용할 수 없게 하는 보안사고를 제외한 대부분의 보안사고는 정보를 탈취하려는 사고가 대부분입니다. 정보를 탈취하려는 보안사고에서 정보 탈취를 위해 사용하는 기법에는 여러 종류가 있습니다.

특정 정보를 위장하는 기법인 위장공격(Spoofing), 특정 정보를 변조하는 변조공격(Tampering), 정보의 전송행위를 부인하는 부인공격(Repudiation), 특정 정보를 유출시키는 유출공격(Information Disclosure), 특정 정보의 탈취를 위해 권한을 상승시키는 권한 상승 공격(Elevation of Privilege)가 있고 마지막으로 서비스를 거부시키는 거부공격(Denail of Service)이 있습니다.

이처럼 많은 공격기법을 방어하기 위해 시큐어코딩이 존재한다고 할 수 있습니다.

과제 2. 사례 기반 정보보호 용어 정의 및 설명

(용어 정의)

- 보안 자산(Security Asset)

우리가 개발한 소프트웨어나 운영하고 있는 서비스에서 가치가 있고 보호해야 하는 유형 혹은 무형의 중요한 요소 → 제품의 이해관계자에게 가치가 있는 것

유형 자산: ex) 현금, 장비, 하드웨어

무형 자산: ex) 소프트웨어, 영업권, 지적 재산권

- 취약점(Vulnerability)

시스템을 위협에 노출시키는 약점으로 공격자가 실제로 공격하여 피해를 줄 수 있는 약점

- 위협원(Threat Agent)

해커, 위협을 수행하는 사람이나 조직

- 위협행동(Threat Action)

피해를 입힐 의도로 목표 대상에 취한 모든 조치 → 공격자가 자신의 목표를 달성하기 위한 전략

- 위험(RISK)

보유한 각종 자산에 피해를 입힐 수 있는 잠재적인 가능성,

위험 요소가 일으킬 수 있는 피해 → 관리 가능

- 보안 조치(Security Measure)

보안 위험을 최소화 시키는 작업

위험과 비용의 균형 중요 → 체계적인 접근 필요

(사이버 보안 사고 사례 - 업무 사칭 악성메일에 의한 정보 유출 & 악성코드 유포 사례)

업무 회신하니 정보 유출됐다고?... 사칭 악성메일 주의보

최근 실제 업무 메일에 대한 회신 형태로 악성코드를 유포하는 사례가 발견돼 사용자 주의가 요구된다.

1일 안랩에 따르면 공격자는 다양한 주제의 업무 관련 메일을 미리 수집했다. 이후 해당 메일을 발송한 사용자를 겨냥해 악성 엑셀 파일이 담긴 압축파일을 첨부해 회신 메일을 보냈다.

이번에 발견한 악성 메일은 총 3종류다. 공적조서 송부, 동영상 편집본 확인 요청, 학술 행사안내 등 업무상 메일에 대한 회신 형식을 취한다. 특히 공격자는 정보공개 협박과 업무 요청 내용을 기재해 사용자의 악성 첨부파일 실행을 유도했다.

특정 사용자가 표창 발급을 위해 발송한 공적조서 메일에 대한 회신으로 “당신의 상사에게 이 메일을 보여줄까?”라며 “정보가 공개되기 전에 첨부파일을 확인하라”고 협박했다. 다른 사용자가 보낸 동영상 편집본 확인 요청 메일에 대해서는 “모든 데이터를 직접 확인하면 좋겠다, 파일을 첨부했다”고 회신했다. 특정 단체가 발송한 학술행사 안내 메일에 대한 회신으로는 “일주일

전에 확인하라고 요청한 내용이 있다. 파일을 복사했으니 확인하라”고 유도했다.

세 경우 모두 악성코드 동작 방식은 같다. 사용자가 무심코 해당 회신 메일의 첨부파일을 내려 받아 악성 엑셀파일(.xlsm)을 실행하면 ‘내용을 보기 위해 콘텐츠 사용 버튼을 클릭하라’는 메시지가 나온다. 이에 속아 화면 상단의 ‘콘텐츠 사용’ 버튼을 클릭할 경우 악성코드에 감염된다. 악성코드는 C&C(명령제어)서버에 접속해 랜섬웨어, 정보유출 악성코드 등을 추가로 다운로드할 수 있다.

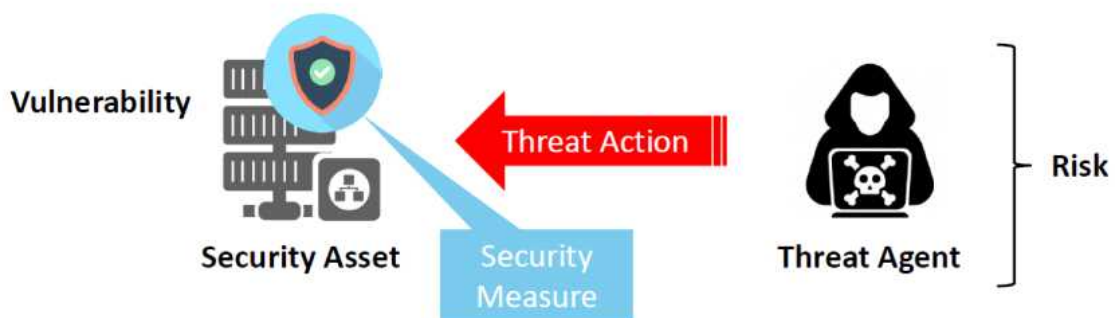
이 같은 악성코드 피해를 줄이기 위해서는 ▲출처가 불분명한 메일의 첨부파일·URL 실행금지 ▲백신 최신버전 유지 및 실시간 감시 기능 실행 ▲파일 실행 전 최신 버전 백신으로 검사 ▲OS(운영체제)와 인터넷 브라우저 및 오피스 프로그램 최신 보안 패치 적용 등 필수 보안 수칙을 지켜야 한다. 현재 V3 제품군은 해당 악성코드를 진단한다.

장서준 안랩 분석팀 주임연구원은 “이번 사례는 공격자가 다양한 방법으로 사용자가 보낸 메일을 수집해 해당 메일의 회신으로 악성코드 유포를 시도한 것이 특징”이라며 “직접 보낸 메일에 대한 회신이기 때문에 사용자들이 의심하지 않고 피해를 당할 수 있어 이를 예방하기 위해서는 출처가 불분명한 메일이나 첨부파일 등은 실행하지 말아야 한다”고 말했다.

[출처: moneys

(<https://moneys.mt.co.kr/news/mwView.php?no=2021060109398087885>)]

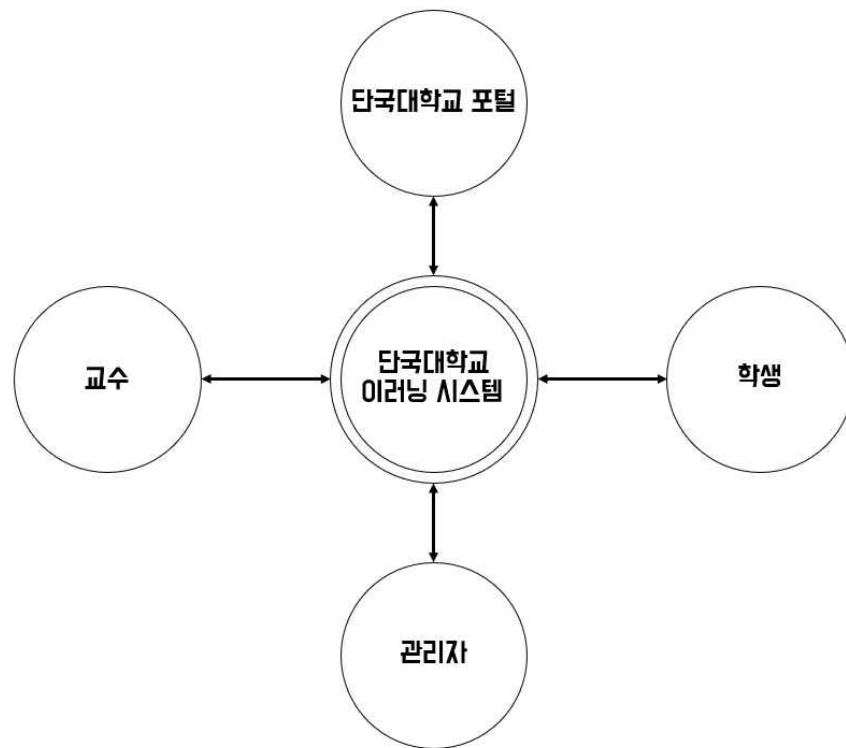
해당 사이버 보안 사고 사례는 공격자가 악성파일이 첨부되어 있는 메일을 피해자에게 전송하고 피해자가 악성파일을 실행시키면 개인정보가 유출되는 사고를 보여준다. 이와 같은 피해를 줄이기 위해서는 사용자의 주의가 제일 중요하다고 할 수 있다. 출처가 불분명한 메일과 사이트의 경우 접근하는 일이 없도록 해야하고 백신을 사용하여 악성파일 혹은 코드를 검사해야한다.



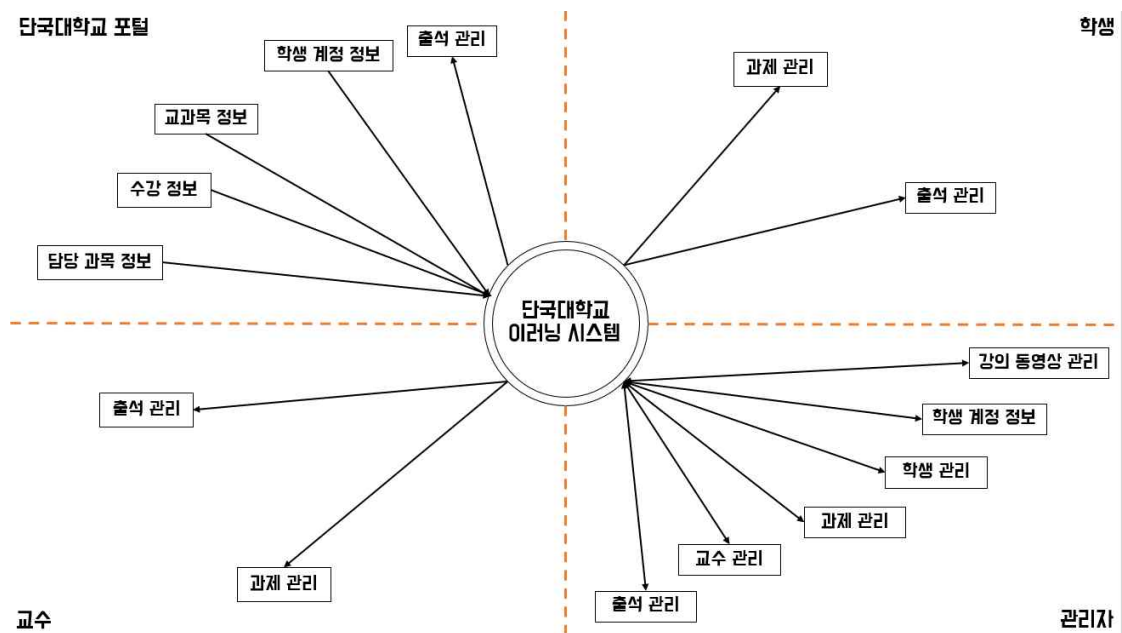
해당 사례에서 **보안 자산**은 유출되는 사용자들의 개인 정보라고 할 수 있고 이러한 개인 정보를 탈취하기 위해 악성파일을 첨부하는 메일을 보내는 행위를 **위협 행동** 이러한 위협 행동을 하는 공격자가 **위협원**이라고 할 수 있다. 이러한 위협은 이러한 사례에 무지한 사용자나 백신이 없는 PC에는 더욱 쉽게 적용(**위협**)될 수 있는데 이를 **취약점**이라고 할 수 있다. 결국 이러한 취약점을 해결하기 위해서는 사용자가 출처가 불분명한 메일과 사이트의 경우 접근하는 일이 없도록 해야하고 백신을 사용하여 악성파일 혹은 코드를 검사하는 것과 같은 **보안 조치**를 취해야 한다.

과제 3. 이러닝 시스템 위협 모델링

(Level 0 DFD)



(Level 1 DFD)



(위협 식별)

No.	위협	데이터 송신	데이터 수신
A1	학생 출석 정보 변경	단국대학교 이러닝 시스템	단국대학교 포털
A2	학생 계정 정보 탈취	단국대학교 포털	단국대학교 이러닝 시스템
A3	교과목 정보 변경	단국대학교 포털	단국대학교 이러닝 시스템
A4	수강 정보 변경	단국대학교 포털	단국대학교 이러닝 시스템
A5	담당 과목 정보 변경	단국대학교 포털	단국대학교 이러닝 시스템
A6	출석 정보 변경	단국대학교 이러닝 시스템	교수
A7	과제 정보 변경	단국대학교 이러닝 시스템	교수
A8	교수 권한 위장		
A9	출석 정보 변경	단국대학교 이러닝 시스템	학생
A10	과제 정보 변경	단국대학교 이러닝 시스템	학생
A11	학생 권한 위장		
A12	강의 동영상 정보 변경	단국대학교 이러닝 시스템 / 관리자	관리자 / 단국대학교 이러닝 시스템
A13	학생 계정 정보 탈취	단국대학교 이러닝 시스템 / 관리자	관리자 / 단국대학교 이러닝 시스템
A14	학생 정보 탈취	단국대학교 이러닝 시스템 / 관리자	관리자 / 단국대학교 이러닝 시스템
A15	과제 정보 변경	단국대학교 이러닝 시스템 / 관리자	관리자 / 단국대학교 이러닝 시스템
A16	교수 정보 탈취	단국대학교 이러닝 시스템 / 관리자	관리자 / 단국대학교 이러닝 시스템
A17	출석 정보 변경	단국대학교 이러닝 시스템 / 관리자	관리자 / 단국대학교 이러닝 시스템
A18	관리자 권한 위장		
A19	서버 마비		
A20	정보 전송 행위의 부인	단국대학교 이러닝 시스템 단국대학교 포털 관리자 교수 학생	단국대학교 이러닝 시스템 단국대학교 포털 관리자 교수 학생

(식별된 위협 STRIDE 매칭)

위협 유형	설명
위장 (Spoofing identity)	false identity를 이용해 시스템 접근 권한을 획득하는 경우
데이터 변조 (Tampering with data)	데이터가 전송될 때 공격자가 데이터를 수정하는 경우
부인 (Repudiation)	공격자 자신이 수행한 특정 행동이나 트랜잭션을 부인하는 경우
정보 유출 (Information disclosure)	개인정보 혹은 잠재적으로 유해한 데이터가 유출되는 경우
서비스 거부 (DoS, Denial of Service)	시스템 또는 어플리케이션의 가용성을 떨어뜨리는 경우
권한 상승 (Elevation of privilege)	권한이 있는 사용자의 권한을 습득하는 경우

No.	위협	STRIDE
A1	학생 출석 정보 변경	T(Tampering)
A2	학생 계정 정보 탈취	I(Information Disclosure)
A3	교과목 정보 변경	T(Tampering)
A4	수강 정보 변경	T(Tampering)
A5	담당 과목 정보 변경	T(Tampering)
A6	출석 정보 변경	T(Tampering)
A7	과제 정보 변경	T(Tampering)
A8	교수 권한 위장	S(Spoofing)
A9	출석 정보 변경	T(Tampering)
A10	과제 정보 변경	T(Tampering)
A11	학생 권한 위장	S(Spoofing)
A12	강의 동영상 정보 변경	T(Tampering)
A13	학생 계정 정보 탈취	I(Information Disclosure)
A14	학생 정보 탈취	I(Information Disclosure)
A15	과제 정보 변경	T(Tampering)
A16	교수 정보 탈취	I(Information Disclosure)
A17	출석 정보 변경	T(Tampering)
A18	관리자 권한 위장	S(Spoofing)
A19	서버 마비	D(Denial of Service)
A20	정보 전송 행위의 부인	R(Repudiation)

(보안 대책)

[S: Spoofing (위장)]

패스워드, 비밀키, 홍채인식, 지문 등의 적절한 인증, 쿠키를 이용한 인증, 전자서명 등으로 위장을 방지할 수 있다.

[T: Tampering (변조)]

Hash, 메시지 인증, 전자서명을 통하여 변조를 방지할 수 있다.

[R: Repudiation (부인)]

전자서명과 감사로그를 사용하여 부인을 방지할 수 있다.

[I: Information Disclosure (정보 유출)]

암호화 프로토콜(대칭키, 공개키)을 사용하여 정보의 유출을 방지할 수 있다.

[D: Denial of Service (서비스 거부)]

서비스에 대한 접근을 필터링하고 지속적인 서비스의 상태를 확인하면 서비스 거부를 방지할 수 있다.

[E: Elevation of Privilege (권한 상승)]

사용자에게 부여하는 권한을 최소화하고 권한의 유효성을 확인함으로써 권한 상승을 방지할 수 있다.

과제 4. 학생 활동 보고서

✓ SQL 삽입(Injection)

(SQL 삽입 공격 실제 사례)

< '여기어때' 해킹 "SQL인젝션 방식 흔적...조사 뒤 손해 배상" >

여기어때.

숙박 온·오프라인연계(O2O) 서비스 '여기어때'가 해킹돼 고객정보가 유출된데 대해 26일 회사 관계자에 의해 “SQL인젝션 방식 흔적이 발견됐다”고 보고 되었다. 회사는 조사 뒤 정확한 피해 규모가 나오면 배상을 실시할 예정이다. 해커가 고객 데이터를 확보한 이상 언제든 추가 피해 가능성이 있다. 전문 보안 컨설팅 업체와 수사기관이 협조해 2차 피해를 막기 위한 장치를 마련 중이다. 고객정보 유출 발표 뒤 이틀 동안 추가 피해 사례는 보고되지 않았다.

여기어때 - 1등 숙박어플 & 5박하면...

여기어때로 숙박예약했는데 오늘 031-8077-2242로 문자가 왔습니다. 제가 언제 어느 모텔에 묵었는지도 알더군요. 제 정보가 알지도 못하는 곳에 유출되었다는 사실에 매우 불쾌합니다. 여기어때 쓰시는 분들 조심하세요 후후로 저 번호 검색해보니 저와 같은 피해를 입으신 분 모두 여기어때로 숙박 예약하신 분들이더군요

번호 유출이라니...지금 보니까 저 말고도 많은 분들이 겪으신 거 같은데. 031 이번호로 문자온거. 고객 정보를 얼마나 허술하게 관리하면 이런 문자가 오는지 모르겠네요

여기 리뷰 보니까 저는 저번에 호텔 찾아보다가 싼게 떠서 결제했더니 다음날 전화와서 잘못 표기난거라고 돈 더내라고 전화옴. 이제 여기어때

여기어때 관계자는 “보안 강화와 추가 피해 막는 게 우선”이라면서 “지금 상황에서 피해 규모를 확정하기 어려워 경찰 수사 종료 뒤 배상할 것”이라고 말했다. SQL인젝션 공격에 당했다면 기존 보안 수준에 문제가 크다는 지적이다. SQL인젝션은 공격자가 주소창이나 아이디·비밀번호 입력창에 명령어를 입력하고 웹사이트에 침투, 서버에서 정보를 탈취하는 기법이다. 흔히 알려진 웹 취약점으로 관련 패치와 업데이트 등이 수차례 배포됐다. 회원수 400만명에 달하는 인기 O2O 서비스가 오래 전 공개된 보안 패치조차 소홀했다는 의미다.

한편 '여기어때' 운영사 위드이노베이션에 따르면 최근 서비스 데이터베이스(DB)에 해커가 침입해 고객정보 일부를 빼갔다. 고객 이름, 전화번호, 이메일, 숙소 정보 등이 유출된 것으로 파악된다. 정확한 피해규모는 조사 중이다. 해커는 문자 대량 발송 서비스 업체도 해킹해 여기어때 고객에게 문자 메시지를 무단 전송했다.

숙박 서비스 이용 관련 성적 수치심이나 불쾌감을 유발하는 내용이다. 문자 메시지 발송 후 여기어때에 비트코인으로 수익원대 금전을 요구하는 협박 메일도 보냈다.

[출처: etnews(<https://m.etnews.com/20170326000040>)]

취약점	해커침입탐지시스템 미준수 개인정보 보호기간 6개월 보관 시 월 1회 정기정검 미이행 개인정보 암호화 미조치 관리자 페이지 접근 변경 권한 미조치 1년간 서비스 미이용자 개인정보 미처리
STRIDE	S(Spoofing identity): 위장
	I(Information Disclosure): 정보 유출

해당 사이버 보안 사고는 여기어때의 개인정보 유출 사건이다. 여기어때의 소홀한 보안 패치가 취약점으로 공격자는 이런 취약점을 공략하여 사용자의 개인정보를 탈취한 것이다. 공격자는 탈취한 개인정보로 사용자에게 문자를 전송하거나 여기어때에 비트코인으로 수억원대의 금전을 요구하는 협박메일도 전송하였다.

(SQL 삽입 공격의 유형)

SQL Injection이란 공격자가 보안상의 취약점을 이용하여, 임의의 SQL 문을 주입하고 실행되게 하여 데이터베이스가 비정상적인 동작을 하도록 조작하는 행위이다. SQL Injection 공격은 OWASP Top10 중 첫 번째에 속해 있으며, 공격이 비교적 쉬운 편이고 공격에 성공할 경우 큰 피해를 입힐 수 있는 공격이다.

대표적인 SQL Injection의 유형에는 Form Based SQL Injection, Union Based SQL Injection과 Blind SQL Injection이 있다.

[Form based SQL Injection]

Form Based SQL Injection은 가장 많이 쓰이고 대중적인 공격 기법이다. Form Based SQL Injection은 웹 페이지 혹은 어플리케이션에서 로그인 폼과 같이 데이터베이스에 쿼리문을 전송하여 데이터를 조회하는 과정에서 취약점이 존재할 때 폼에 특수한 쿼리문을 사용하여 인증과정을 무시 혹은 우회하여 접근하는 공격이다.

다음은 Form based SQL Injection의 예시이다.

(1) 비정상적인 입력 값으로 인증 우회 가능성 확인

안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지

root cause
java.sql.SQLException: Error: executeQueryForObject returned too many results.
com.ibatis.sqlmap.engine.mapping.statement.MappedStatement.executeQueryForObject(MappedStatement.java:124)
com.ibatis.sqlmap.engine.impl.SqlMapExecutorDelegate.queryForObject(SqlMapExecutorDelegate.java:518)
com.ibatis.sqlmap.engine.impl.SqlMapExecutorDelegate.queryForObject(SqlMapExecutorDelegate.java:493)
com.ibatis.sqlmap.engine.impl.SqlMapSessionImpl.queryForObject(SqlMapSessionImpl.java:106)
org.springframework.orm.ibatis.SqlMapClientTemplate.doInSqlMapClient(SqlMapClientTemplate.java:270)
org.springframework.orm.ibatis.SqlMapClientTemplate.execute(SqlMapClientTemplate.java:200)
org.springframework.orm.ibatis.SqlMapClientTemplate.queryForObject(SqlMapClientTemplate.java:200)
kr.co.openeg.lab.login.dao.LoginDaoImpl.selectUserId(LoginDaoImpl.java:19)
kr.co.openeg.lab.login.service.LoginService.checkUserId(LoginService.java:23)
kr.co.openeg.lab.login.controller.LoginController.loginProc(LoginController.java:49)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
java.lang.reflect.Method.invoke(Method.java:606)

사용자명 :
or'a='a
비밀번호 :

로그인 회원가입

ID: 'or 'a'='a
Password: 'or 'a'='a

안전한 소프트웨어를 만들기 위한 노력

Copyright (C) (주)오픈이지(http://openeg.co.kr)

HTTP Status 500 Error가 발생하였고 아래쪽에 'too many results' 라는 구문을 확인 하여 정상적인 상황보다 많은 정보를 요청하여 서버가 표시할 수 없다는 것을 알 수 있다.

진단자가 의도한 실행 쿼리문은 select * from member where id=" or 'a'='a' and password=" or 'a'='a'이고 이에 우리는 SQL 삽입 공격에 취약한 웹 애플리케이션이라는 것을 확인할 수 있다.

(2) 비정상적인 입력 값으로 인증 우회 확인

안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지



ID: admin'#
Password: aaa

로그인 회원가입

안전한 소프트웨어를 만들기 위한 노력

Copyright (C) (주)오픈이지(<http://openeg.co.kr>).

ID에는 공격코드를 삽입한 코드를 입력하고 Password에는 의미 없는 값을 입력해준다.

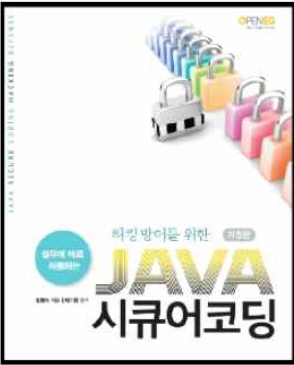
안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지

소프트웨어 보안은 보안소프트웨어가 아닙니다.

소프트웨어 보안을 유지한다는 것은 암호화화 같은 다양한 보안기능의 적용을 위함 하는 것보다 소프트웨어 라이프 사이클 전반에 걸쳐 여러가지 안전한 소프트웨어 개발의 모범사례를 적용하는 것을 의미합니다.

보안문제는 특정 보안기능보다 안전한 시스템을 구성하는 표준의 문제로 인해 발생할 수 있습니다.

그래서 소프트웨어 보안은 전체 개발단계의 라이프 사이클 접근 방식의 일부가 되어야 하는 중요한 이유입니다.



[관리자]님 환영합니다.
로그아웃 정보수정

Copyright (C) (주)오픈이지(<http://openeg.co.kr>).

공격자가 의도한 쿼리문은 `select * from member where id='admin' #' and password='aaa'` 였고 성공적으로 관리자 모드로 로그인 되는 것으로 보아 SQL 삽입 공격에 취약한 웹 어플리케이션임을 확인할 수 있다.

[Union Based SQL Injection]

Union Based SQL Injection는 Union SQL Injection 취약점을 이용하여 단계적으로 보안 자산의 정보를 추출하고 공격자가 원하는 중요 정보 자산을 탈취하는 과정이다.

SQL 에서 Union 키워드는 두 개의 쿼리문에 대한 결과를 통합해서 하나의 테이블로 보여주게 하는 키워드 이다. 정상적인 쿼리문에 Union 키워드를 사용하여 인젝션에 성공하면, 원하는 쿼리문을 실행할 수 있게 된다. Union SQL Injection을 성공하기 위해서는 두 가지의 조건이 있다. 하나는 Union 하는 두 테이블의 컬럼 수가 같아야 하고, 데이터 형이 같아야 한다.

다음은 Union based SQL Injection의 예시이다.



SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 악성 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증우회)

ID: PASSWORD:

(2) MySQL 인젝션

ID:

(3) MS-SQL 인젝션

ID:

실행결과

MySQL 조회결과: IDX: 1 ID: admin PASSWORD: openeg 이름: 관리자

admin에 대한 IDX, ID, PASSWORD, 이름에 대한 정보가 확인되는 것으로 보아 해당 페이지에 SQL 삽입 공격에 대한 취약점이 있다면 UNION을 활용하여 또 다른 정보를 추출할 수 있다는 것을 파악할 수 있다.

(1) 정상적인 요청 처리

SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 작된 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증 우회)

ID:

PASSWORD:

실행

(2) MySQL 인젝션

ID:

실행

(3) MS-SQL 인젝션

ID:

실행

실행결과

MySQL 조회결과: 요청 처리 에러 발생

union 뒤의 select 문을 활용해 데이터베이스에서 사용하고 있는 컬럼의 개수를 확인하는 과정이다.

(2) 공격 가능성 확인

순차적으로 컬럼의 개수를 늘려가며 확인한다.

SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 작된 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증 우회)

ID:

PASSWORD:

실행

(2) MySQL 인젝션

ID:

실행

(3) MS-SQL 인젝션

ID:

실행

실행결과

MySQL 조회결과:

IDX: 1

ID: 2

PASSWORD: 3

이름: 4

ID: admin

PASSWORD: openeg

이름: 관리자

- 12 -

순차적으로 컬럼의 개수를 늘려가며 조회해본 결과 컬럼의 개수가 6일 때 정보가 나타남을 확인할 수 있다. 이에 우리는 데이터베이스의 컬럼의 개수가 6개임을 확인할 수 있고 1~4번에 악의적인 쿼리문을 삽입하면 정보 유출 가능성이 있다는 것을 알 수 있다.

(3) DBMS 버전 확인

SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 작된 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증우회)

ID: PASSWORD: 실행

(2) MySQL 인젝션

ID: admin' union select version(),2,3,4,5,6 from 실행

(3) MS-SQL 인젝션

ID: 실행

실행결과

MySQL 조회결과:	IDX: 1	ID: admin	PASSWORD: openeg	이름: 관리자
	IDX: 5.1.41-community	ID: 2	PASSWORD: 3	이름: 4

union select와 버전 키워드를 사용함으로써 해당 DBMS의 버전 정보를 얻어올 수 있다.

(4) 공격대상 DB목록 확인

SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 작된 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증우회)

ID: PASSWORD: 실행

(2) MySQL 인젝션

ID: admin' union select schema_name,2,3,4,5,6 from information_sc 실행

admin' union select schema_name,2,3,4,5,6 from

(3) MS-SQL 인젝션

information_schema.schemata #

ID: 실행

실행결과

MySQL 조회결과:	IDX: 1	ID: admin	PASSWORD: openeg	이름: 관리자
IDX: information_schema	ID: 2	PASSWORD: 3	이름: 4	
IDX: board	ID: 2	PASSWORD: 3	이름: 4	
IDX: dvwa	ID: 2	PASSWORD: 3	이름: 4	
IDX: hachmebooks	ID: 2	PASSWORD: 3	이름: 4	
IDX: mysql	ID: 2	PASSWORD: 3	이름: 4	
IDX: openeg	ID: 2	PASSWORD: 3	이름: 4	
IDX: owasp10	ID: 2	PASSWORD: 3	이름: 4	
IDX: phpmyadmin	ID: 2	PASSWORD: 3	이름: 4	
IDX: puzzlemalldb	ID: 2	PASSWORD: 3	이름: 4	

해당 공격 구문을 활용하여 우리는 데이터베이스의 목록을 확인할 수 있다.

(5) 특정 DB선택 후, 테이블 목록 확인

SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 조작된 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증 우회)

ID: PASSWORD: 실행

(2) MySQL 인젝션

ID: 실행

(3) MS-SQL 인젝션

admin' union select group_concat(table_name),2,3,4,5,6 from information_schema.tables where table_schema=database() #

ID: 실행

실행결과

MySQL 조회결과: IDX: 1 ID: admin PASSWORD: openeg 이름: 관리자
IDX: board,board_comment,board_member,login_history,openeg_security ID: 2 PASSWORD: 3 이름: 4

해당 공격 구문을 활용하여 우리는 Information_schema 데이터베이스에서 현재 사용하고 있는 데이터베이스 table의 목록을 확인할 수 있다.

(6) 테이블의 컬럼명 확인

SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 조작된 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증 우회)

ID: PASSWORD: 실행

(2) MySQL 인젝션

ID: 실행

(3) MS-SQL 인젝션

admin' union select group_concat(column_name),2,3,4,5,6 from information_schema.columns where table_name='board_member' #

ID: 실행

실행결과

MySQL 조회결과: IDX: 1 ID: admin PASSWORD: openeg 이름: 관리자
IDX: IDX,USERID,USERPW,USERNAME,PINNO,JOINDATE ID: 2 PASSWORD: 3 이름: 4

해당 공격 구문을 활용하여 우리는 board_member table의 컬럼명을 확인할 수 있다.

(7) 컬럼 데이터 추출

SQL 인젝션

외부입력값에 SQL문을 조작할 수 있는 입력값이 안전하게 필터링되지 않고 사용되는 경우 공격자가 의도하는 조작된 쿼리가 수행되는 침해사고가 발생할 수 있습니다.

(1) MySQL 인젝션 (인증우회)

ID: PASSWORD: 실행

(2) MySQL 인젝션

ID: 실행

**admin' union select idx,userid,userpw,username,5,6
from board_member #**

(3) MS-SQL 인젝션

ID: 실행

실행결과

MySQL 조회결과:	IDX: 1	ID: admin	PASSWORD: openeg	이름: 관리자
IDX: 1	ID: admin	PASSWORD: openeg	이름: 관리자	
IDX: 2	ID: test	PASSWORD: test	이름: 테스트	
IDX: 3	ID: abcd	PASSWORD: abcd	이름: abcd	

해당 공격 구문을 활용하여 우리는 board_member table의 컬럼 정보를 추출할 수 있다.

[Blind SQL Injection]

< Boolean Based SQL Injection >

Blind SQL Injection은 데이터베이스로부터 특정한 값이나 데이터를 전달받지 않고, 단순히 참과 거짓의 정보만 알 수 있을 때 사용한다. 로그인 폼에 SQL Injection이 가능하다고 가정했을 때, 서버가 응답하는 로그인 성공과 로그인 실패 메시지를 이용하여, DB의 테이블 정보 등을 추출해 낼 수 있다.

① SELECT * FROM Users WHERE id = 'INPUT1' AND password = 'INPUT2'



②



abc123' and ASCII(SUBSTR(SELECT name FROM information_schema.tables
WHERE table_type='base table' limit 0,1),1,1)) > 100 --
(MySQL 일 경우)

③ SELECT * FROM Users WHERE id = 'abc123' and ASCII(SUBSTR(SELECT name FROM information_schema.tables WHERE table_type='base table' limit 0,1),1,1)) > 100 -- ' AND password = 'INPUT2' 로그인 이 될 때까지 시도

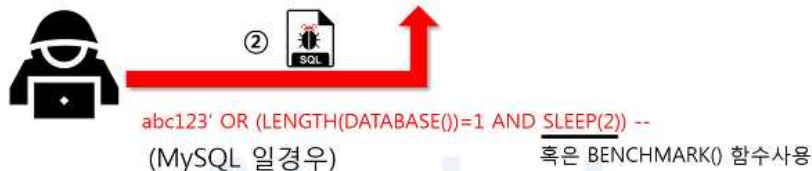
위의 그림은 Blind Injection을 이용하여 데이터베이스의 테이블 명을 알아내는 방법이다. 인젝션이 가능한 로그인 폼을 통하여 악의적인 사용자는 임의로 가입한 abc123 이라는 아이디와 함께 abc123' and ASCII(SUBSTR(SELECT name From information_schema.tables WHERE table_type='base table' limit 0,1),1,1)) > 100 -- 이라는 구문을 주입한다.

해당구문은 MySQL 에서 테이블 명을 조회하는 구문으로 limit 키워드를 통해 하나의 테이블만 조회하고, SUBSTR 함수로 첫 글자만, 그리고 마지막으로 ASCII 를 통해서 ascii 값으로 변환해 준다. 만약에 조회되는 테이블 명이 Users 라면 'U' 자가 ascii 값으로 조회가 될 것이고, 뒤의 100 이라는 숫자 값과 비교를 하게 된다. 거짓이면 로그인 실패가 될 것이고, 참이 될 때까지 뒤의 100이라는 숫자를 변경해 가면서 비교를 하면 된다. 공격자는 이 프로세스를 자동화 스크립트를 통하여 단기간 내에 테이블 명을 알아 낼 수 있다.

< Time Based SQL Injection >

Time Based SQL Injection 도 마찬가지로 서버로부터 특정한 응답 대신에 참 혹은 거짓의 응답을 통해서 데이터베이스의 정보를 유추하는 기법이다. 사용되는 함수는 MySQL 기준으로 SLEEP 과 BENCHMARK이다.

① SELECT * FROM Users WHERE id = 'INPUT1' AND password = 'INPUT2'



③ SELECT * FROM Users WHERE id = 'abc123' OR (LENGTH(DATABASE())=1 AND SLEEP(2)) --
' AND password = 'INPUT2' SLEEP 할 때까지 시도

위의 그림은 Time based SQL Injection을 사용하여 현재 사용하고 있는 데이터베이스의 길이를 알아내는 방법이다. 로그인 폼에 주입이 되었으며 임의로 abc123 이라는 계정을 생성해 두었다. 악의적인 사용자가 abc123' OR (LENGTH(DATABASE())=1 AND SLEEP(2)) - 이라는 구문을 주입하였다. 여기서 LENGTH 함수는 문자열의 길이를 반환하고, DATABASE 함수는 데이터베이스의 이름을 반환한다.

주입된 구문에서, LENGTH(DATABASE()) = 1 가 참이면 SLEEP(2) 가 동작하고, 거짓이면 동작하지 않는다. 이를 통해서 숫자 1 부분을 조작하여 데이터베이스의 길이를 알아 낼 수 있다. 만약에 SLEEP 이라는 단어가 치환처리 되어있다면, 또 다른 방법으로 BENCHMARK 나 WAIT 함수를 사용할 수 있다. BENCHMARK 는 BENCHMARK(1000000,AES_ENCRYPT('hello','goodbye')); 이런 식으로 사용이 가능하다.

(보안 정보 공유 체계(CVE, CWE, NVD))

- CVE

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	894	177	112	172			2	7		25	16	103			2
2000	1020	257	208	206		2	4	20		48	19	139			
2001	1677	403	403	297		7	34	123		83	36	220		2	2
2002	2156	498	553	435	2	41	200	103		127	76	199	2	14	1
2003	1527	381	477	372	2	50	129	60	1	62	69	144		16	5
2004	2451	580	614	408	3	148	291	111	12	145	96	134	5	38	5
2005	4935	838	1627	657	21	604	786	202	15	289	261	221	11	100	14
2006	6610	893	2719	664	91	967	1302	322	8	267	272	184	18	849	30
2007	6520	1101	2601	955	95	706	883	338	14	267	326	242	69	700	45
2008	5632	894	2310	699	128	1101	807	362	7	288	268	188	83	170	76
2009	5736	1035	2185	698	188	963	851	323	9	337	302	223	115	138	738
2010	4653	1102	1714	676	342	520	605	276	8	234	284	238	86	73	1501
2011	4155	1221	1334	735	351	294	470	108	7	197	411	206	58	17	557
2012	5297	1425	1459	833	423	243	759	122	13	344	392	250	166	14	623
2013	5191	1455	1186	856	366	156	650	110	7	352	512	274	123	1	206
2014	7939	1599	1572	841	420	304	1103	204	12	457	2106	239	264	2	403
2015	6504	1793	1830	1084	749	221	784	151	12	577	753	366	248	5	129
2016	6454	2029	1496	1313	717	94	498	99	15	444	870	602	86	7	1
2017	14714	3155	3004	2495	745	508	1518	279	11	629	1659	459	327	18	6
2018	16557	1853	3041	2121	400	517	2048	545	11	708	1239	247	461	31	4
2019	17344	1342	3201	1270	488	549	2390	465	10	710	983	202	535	57	13
2020	18325	1351	3248	1618	409	460	2178	401	14	966	1345	310	402	37	62
2021	7986	800	1663	680	143	233	935	190	1	339	404	123	167	16	
Total	154277	26182	38557	20085	6083	8688	19227	4921	187	7895	12699	5513	3226	2305	4423
% Of All		17.0	25.0	13.0	3.9	5.6	12.5	3.2	0.1	5.1	8.2	3.6	2.1	1.5	

전체 사이버 보안 공격 중 5.6% 차지하고 있는 것을 확인할 수 있고 적지 않은 비중을 차지하고 있다는 것을 알 수 있다.

Security Vulnerabilities (SQL Injection)														
CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9														
Sort Results By: CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending														
Copy Results Download Results														
#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2021-33470			Sql	2021-05-26	2021-05-26	0.0	None	???	???	???	???	???	???
COVID19 Testing Management System 1.0 is vulnerable to SQL Injection via the admin panel.														
2	CVE-2021-33180			Exec Code Sql	2021-06-01	2021-06-01	6.0	None	???	???	???	???	???	???
Improper neutralization of special elements used in an SQL command ("SQL Injection") vulnerability in cgi component in Synology Media Server before 1.8.1-2876 allows remote attackers to execute arbitrary SQL commands via unspecified vectors.														
3	CVE-2021-32615	89		Sql	2021-05-13	2021-05-21	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
Pwigo 11.4.0 allows admin/user_list_backend.php order[0][dir] SQL Injection.														
4	CVE-2021-32104	89		Sql	2021-05-07	2021-05-11	6.5	None	Remote	Low	???	Partial	Partial	Partial
A SQL injection vulnerability exists (with user privileges) in interface/forms/eye_mag/save.php in OpenEMR 5.0.2.1.														
5	CVE-2021-32102	89		Sql	2021-05-07	2021-05-11	6.5	None	Remote	Low	???	Partial	Partial	Partial
A SQL injection vulnerability exists (with user privileges) in library/custom_template/ajax_code.php in OpenEMR 5.0.2.1.														
6	CVE-2021-32099	89		Sql Bypass	2021-05-07	2021-05-11	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
A SQL injection vulnerability in the pandora_console component of Artica Pandora FMS 742 allows an unauthenticated attacker to upgrade his unprivileged session via the /include/chart_generator.php session_id parameter, leading to a login bypass.														
7	CVE-2021-32051	89		Sql	2021-05-14	2021-05-21	5.0	None	Remote	Low	Not required	Partial	None	None
Hexagon Glinius Auskunftsportal before 5.0.0.0 allows SQL injection via the GIPWorkflow/Service/DownloadPublicFile id parameter.														
8	CVE-2021-31856			Exec Code Sql	2021-04-28	2021-04-28	6.0	None	???	???	???	???	???	???
A SQL Injection vulnerability in the REST API in Layers Meshery 0.5.2 allows an attacker to execute arbitrary SQL commands via the /experimental/patternfiles endpoint (order parameter in GetMesheryPatterns in models/meshery_pattern_persist.go).														
9	CVE-2021-31827	89		Sql	2021-05-18	2021-05-25	6.5	None	Remote	Low	???	Partial	Partial	Partial
In Progress MOVEit Transfer before 2021.0 (13.0), a SQL injection vulnerability has been found in the MOVEit Transfer web app that could allow an authenticated attacker to gain unauthorized access to MOVEit Transfer's database engine being used (MySQL, Microsoft SQL Server, or Azure SQL), an attacker may be able to infer information about the structure and contents of the database in addition to executing SQL statements that alter or destroy database elements. This is in MOVEit.DMZ.WebApp in SILHuman.vb.														
10	CVE-2021-31777			Sql	2021-04-28	2021-05-03	6.0	None	???	???	???	???	???	???
The dce (aka Dynamic Content Element) extension 2.0.0 through 2.6.x before 2.6.2, and 2.7.x before 2.7.1, for TYPO3 allows SQL Injection via a backend user account.														
11	CVE-2021-31316	89		Sql	2021-05-18	2021-05-24	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
The unprivileged user portal part of CentOS Web Panel is affected by a SQL Injection via the 'idsession' HTTP POST parameter.														

CVE ID를 살펴보면 'CVE-년도-번호'인 형태로 저장되어 있는 것으로 보아 해당 사건 발생 년도와 해당 사건의 번호로 관리되어 지고 있는 것을 확인할 수 있다. 또한 CWE ID가 적혀있는 것으로 보아 CWE와 연개하여 관리되어지고 있다는 것 또한 확인할 수 있었고 Score를 통해 해당 사건의 위험도를 점수로 표시하고 있는 것을 알 수 있었다.

- CWE

CWE Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

Home > CWE List > CWE- Individual Dictionary Definition (4.4)

Home | About | CWE List | Scoring | Community | News | Guidance | Search

CWE VIEW: Weaknesses in the 2020 CWE Top 25 Most Dangerous Software Weaknesses

View ID: 1350
Type: Graph

▼ Objective
CWE entries in this view are listed in the 2020 CWE Top 25 Most Dangerous Software Weaknesses.

▼ Audience

Stakeholder	Description
Software Developers	By following the CWE Top 25, developers are able to significantly reduce the number of weaknesses that occur in their software.
Product Customers	Customers can use the weaknesses in this view in order to formulate independent evidence of a claim by a product vendor to have eliminated / mitigated the
Educators	Educators can use this view to focus curriculum and teachings on the most dangerous weaknesses.

▼ Relationships
The following graph shows the tree-like relationships between weaknesses that exist at different levels of abstraction. At the highest level, categories and pillars exist to group weak technically weaknesses) are special CWE entries used to group weaknesses that share a common characteristic. Pillars are weaknesses that are described in the most abstract fashion weaknesses are varying levels of abstraction. Classes are still very abstract, typically independent of any specific language or technology. Base level weaknesses are used to present variant is a weakness that is described at a very low level of detail, typically limited to a specific language or technology. A chain is a set of weaknesses that must be reachable or exploitable vulnerability. While a composite is a set of weaknesses that must all be present simultaneously in order to produce an exploitable vulnerability.

Expand All | Collapse All

1350 - Weaknesses in the 2020 CWE Top 25 Most Dangerous Software Weaknesses

- ↳ Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') - (79)
- ↳ Out-of-bounds Write - (787)
- ↳ Improper Input Validation - (20)
- ↳ Out-of-bounds Read - (125)
- ↳ Improper Restriction of Operations within the Bounds of a Memory Buffer - (119)
- ↳ Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') - (89)
- ↳ Exposure of Sensitive Information to an Unauthorized Actor - (200)

CWE를 살펴보면 2020 CWE의 약점 Top 25 가장 위험한 소프트웨어 약점에 SQL Injection이 포함되어 있는 것을 확인할 수 있고 이를 통해 SQL Injection이 얼마나 위험하고 흔하게 발생하는 위험인지를 인지할 수 있다.

CWE Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

Home > CWE List > CWE- Individual Dictionary Definition (4.4)

Home | About | CWE List | Scoring | Community | News | Guidance | Search

CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Weakness ID: 89
Abstraction: Base
Structure: Simple

▼ Demonstrative Examples

Example 1
In 2008, a large number of web servers were compromised using the same SQL Injection attack string. This single string worked against many different programs. The SQL Injection was then used to modify the web sites to serve malicious code.

Example 2
The following code dynamically constructs and executes a SQL query that searches for items matching a specified name. The query restricts the items displayed to those where owner matches the user name of the currently-authenticated user.

```
Example Language: C#
...
string userName = ctx.GetAuthenticatedUserName();
string query = "SELECT * FROM items WHERE owner = '" + userName + "' AND itemname = '" + itemName.Text + "'";
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
...
```

The query that this code intends to execute follows:

```
SELECT * FROM items WHERE owner = <userName> AND itemname = <itemName>;
```

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if itemName does not contain a single-quote character. If an attacker with the user name wiley enters the string:

```
name' OR 'a'='a
```

아래쪽을 살펴보면 해당 유형의 SQL Injection에 대한 다양한 공격방법의 예시를 확인할 수 있다.

- NVD

Information Technology Laboratory
NATIONAL VULNERABILITY DATABASE

VULNERABILITIES

CVE-2021-33470 Detail

UNDERGOING ANALYSIS

This vulnerability is currently undergoing analysis and not all information is available. Please check back soon to view the completed vulnerability summary.

Description

COVID19 Testing Management System 1.0 is vulnerable to SQL Injection via the admin panel.

Severity

CVSS Version 3.x

CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD
Base Score: N/A

NVD score not yet provided.

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have not published a CVSS score for this CVE at this time. NVD Analysts use publicly available information at the time of analysis to associate CVSS vector strings.

QUICK INFO

CVE Dictionary Entry:
CVE-2021-33470

NVD Published Date:
05/26/2021

NVD Last Modified:
05/26/2021

Source:
MITRE

NVD를 통해 SQL Injection 사건인 CVE-2021-33470의 Score를 확인할 수 있다.

[CVE List](#)
[CNAs](#)
[WG's](#)
[Board](#)
[About](#)
[News & Blog](#)

[Search CVE List](#)
[Downloads](#)
[Data Feeds](#)
[Update a CVE Record](#)
[Request CVE IDs](#)

TOTAL CVE Records: 154789

[HOME](#) > [ABOUT CVE](#) > [CVE AND NVD RELATIONSHIP](#)

CVE and NVD Relationship

CVE and NVD Are Two Separate Programs


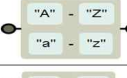




The [CVE List](#) was launched by [MITRE](#) as a community effort in 1999, and the [U.S. National Vulnerability Database \(NVD\)](#) was launched by the [National Institute of Standards and Technology \(NIST\)](#) in 2005.

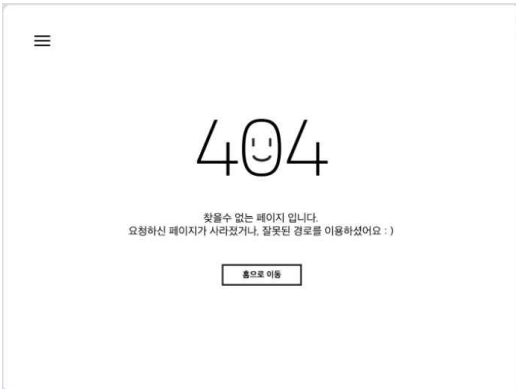
- CVE** - A list of records—each containing an identification number, a description, and at least one public reference—for publicly known cybersecurity vulnerabilities. CVE Records are used in numerous [cybersecurity products and services](#) from around the world, including NVD.
- NVD** - A vulnerability database built upon and fully synchronized with the CVE List so that any updates to CVE appear immediately in NVD.
- Relationship** - The CVE List feeds NVD, which then builds upon the information included in CVE Records to provide enhanced information for each record such as fix information, severity scores, and impact ratings. As part of its enhanced information, NVD also provides advanced searching features such as by OS; by vendor name, product name, and/or version number; and by vulnerability type, severity, related exploit range, and impact.

While separate, both CVE and NVD are sponsored by the [U.S. Department of Homeland Security \(DHS\) Cybersecurity and Infrastructure Security Agency \(CISA\)](#), and both are available to the public and free to use.

CVE는 NVD에게 공격 리스트를 공급하고 CVE 레코드에 포함된 정보를 기반으로 정보 수정, 심각도 점수 및 영향 등급과 같은 각 레코드에 대한 향상된 정보를 제공한다. 향상된 정보의 일부로 NVD는 OS와 공급 업체 이름, 제품 이름 및 / 또는 버전 번호 취약성 유형, 심각도, 관련 익스플로잇 범위 및 영향에 따른 고급 검색 기능도 제공한다.

(SQL 삽입 공격 유형 별 방어 대책)

정규표현식	표현	설명
[alnum:]		알파벳과 숫자를 찾습니다.
[alpha:]		알파벳을 찾습니다.
[digit:]		0~9사이를 찾습니다.
[lower:]		알파벳 소문자를 찾습니다.
[upper:]		알파벳 대문자를 찾습니다.
[blank:]		탭과 공백문자를 찾습니다.



SQL Injection은 웹 어플리케이션에서 DB에 쿼리 입력 시 입력된 쿼리의 유효성을 검증하지 않아 개발자가 의도하지 않은 동적 쿼리를 생성하여 DB정보를 열람하거나 조작할 수 있는 보안 취약점이다.

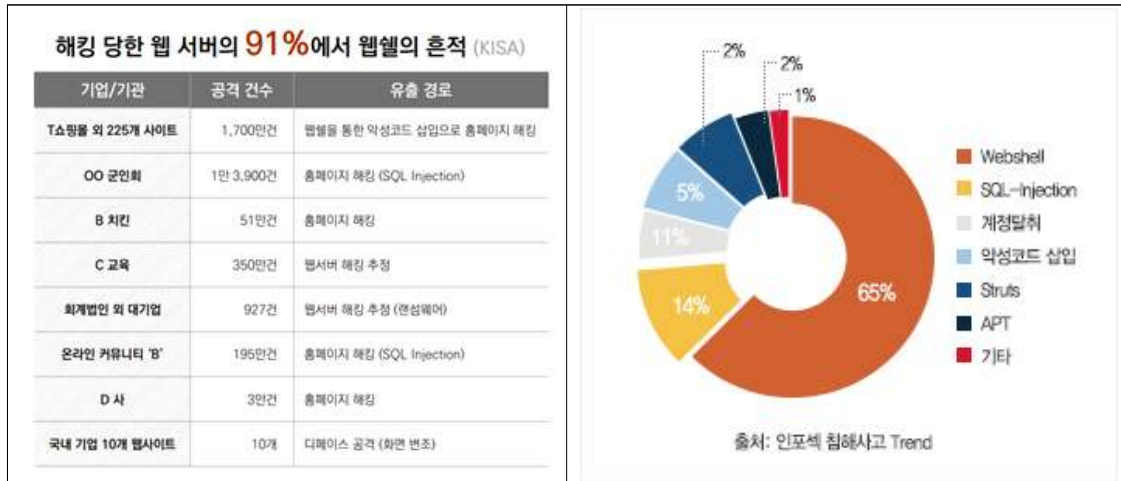
따라서 개발자는 소프트웨어를 개발할 때에 `[*, -, ' ", ?, #, (,), ;, @, =, *, +, union, select, drop, update, from, where, join, substr, user_tables, user_table_columns, information_schema, sysobject, table_schema, declare, dual]`등과 같이 자신이 의도하지 않은 입력값에 대해 자바스크립트를 통해 정규표현식을 사용하여 폼 입력값을 검증하고 차단해야 한다.

또한 에러메세지를 통해 의도치 않게 공격자에게 정보를 보여주는 상황을 막기 위해서는 각각의 에러에 따른 오류페이지를 따로 만들어 제공하거나 에러메세지를 만들어 제공하여야 한다.

✓ 파일 업로드 취약점

(파일 업로드 취약점 기반 공격 실제 사례)

파일 업로드 공격의 대표적인 예로는 웹셸(WebShell)이 있다. 웹셸은 공격자가 원격으로 웹서버를 제어할 수 있는 프로그램으로 이와 같은 웹셸 공격으로 인해 웹 서버에 저장된 개인정보가 유출되고, 웹사이트 변조, 악성코드 유포지로 악용되는 사례가 증가되고 있다.



〈 웹서버 해킹 주범 ‘웹셸’ 제거하기大作战 〉

해킹당한 웹서버 중 90% 이상 웹서버에서 웹셸 발견

웹 보안의 기본은 웹셸 탐지 및 방어...웹셸 방어 솔루션 살펴보니

수많은 웹 보안위협 중에서도 웹셸(WebShell)은 가장 기초적인 해킹 툴이다. 이는 공격자가 원격으로 웹서버를 제어할 수 있는 프로그램으로, 최근 이와 같은 웹셸 공격으로 인해 웹 서버에 저장된 개인정보가 유출되고, 웹사이트 변조·악성코드 유포지로 악용되는 사례가 증가하고 있다.

최근 이러한 웹셸에 의한 해킹 사고가 증가하고 있다. 지난해말 개인정보가 유출된 배달 앱 ‘배달통’의 해킹원인도 웹셸에 의한 것으로 밝혀졌다. 지난해 전 세계 개인정보 유출사고의 65%는 해킹에 의한 것으로 나타났다. 그리고 지난해 우리나라에서 발생한 3.20 사이버테러와 6.25 사이버테러도 해킹에 의한 침해사고였는데, 이들은 모두 웹셸 공격에서부터 시작됐다.

과거에도 이와 같은 웹셸 공격은 지속되어 왔다. 지난 2008년 옥션의 개인정보유출 사고와 2011년 현대캐피탈 정보유출, 2012년 EBS 정보유출 등도 웹셸에 의한 해킹 사건이었다.

KISA 인터넷침해대응센터에 따르면, 해킹당한 웹서버 중 웹셸이 발견된 웹서버는 90% 이상이다. 지난해 2월 26일 의사협회 8만명, 치과의사협회 5만 6천명, 한의사협회 2만명 등 총 15만 6천명의 개인정보가 유출됐다. 이 3개 협회 홈페이지는 공격자가 악성코드를 웹사이트에 심어서 관리자 권

한을 획득한 후, 웹셀을 이용해 개인정보를 탈취한 것으로 조사됐다. 지난해 3월 7일 113만명의 개인정보가 유출된 티켓몬스터의 경우에도 공격자가 홈페이지 게시판 등에 웹셀을 심어 해킹했다.

이와 같은 웹셀을 탐지하고 방어하기 위해서는 웹셀전용 보안 솔루션을 사용하는 것이 보다 효과적이다. 기존 네트워크와 서버 보안 체계에서 웹셀 탐지가 쉽지 않기 때문. 그 이유는 여러 가지인데, 일례로 방화벽의 경우에는 허용된 IP나 포트로부터의 공격은 방어하기 어렵다. 또 네트워크 계층에서의 유해성 검사를 진행하는 IDS/IPS의 경우에는 애플리케이션 취약성 공격에 대해서는 방어가 거의 불가능하다.

이에 많은 사람들이 이용하는 공공기관의 홈페이지는 웹 보안 강화는 필수적이다. 웹셀에 효과적으로 대응하기 위해서는 웹 애플리케이션 파일의 임의 생성 및 변조를 실시간으로 탐지해야 하고 탐지 즉시 처리할 수 있어야 한다. 특히 ASP, JSP, PHP 등 다양한 웹 스크립트 탐지를 지원하고 지속적인 R&D를 통해 진화하는 웹셀 탐지 패턴을 보유할 수 있어야 한다. 또한, 상시 모니터링으로 취약점을 제거하고 웹사이트의 개발·운영 전반에 걸쳐 보안을 강화해야 한다.

웹셀 솔루션 전문 기업 유엠브이기술 조혁래 이사는 “최근 해킹으로 인한 정보유출 사건이 대부분이 웹셀 공격으로 이루어졌다. 이에 고객들은 웹셀 전용 솔루션의 필요성은 인식하고는 있으나, 도입예산 문제로 아직까지 시장이 크게 확대되지는 않고 있다”고 말했다.

이어서 그는 “웹셀 방어 솔루션은 특히 공공이나 금융 분야에서 대부분 도입하고 있다. 많은 사용자들이 이용하고 있기 때문에 웹 보안 강화를 위해서는 필수적이지만 기관이나 기업에서의 웹셀 탐지 솔루션 도입이 확대되지 않고 있어 보안 위협에 여전히 노출되어 있다”라고 말했다. 이러한 웹셀 탐지 및 방어 솔루션은 국내 약 7개 정도의 솔루션이 시장에 나와 있다.

유엠브이기술의 통합웹보안 솔루션 ‘웹서버세이프가드(WSS)2.5’는 셸모니터(ShellMonitor)와 포저리모니터로 구성되어 있다. 셸모니터(ShellMonitor)는 웹서버 악성코드 탐지 방어를 위한 전용 보안 솔루션이고 WSS포저리모니터는 서비스 중인 홈페이지 위·변조가 발생하면 실시간으로 인지한다.

특히, 셸모니터는 위·변조가 되지 않은 원본파일로 즉시 복원하고 관리자에게 통보한다. 웹페이지 위·변조로 인한 기업이나 기관 신뢰도 하락을 막을 수 있다. 웹셀만 탐지하는 제품과 달리 통합적인 웹 보안의 방향을 제시한다. 또한 윈도우서버, 리눅스, 유닉스 등 모든 웹 서버 OS를 지원하며 ASP, JSP, PHP, HTML, JS 등의 웹 서비스 언어를 포함한 이미지 등 모든 소스 파일을 대상으로 웹 서버 악성코드인 웹셀, 악성코드 유포지 URL 및 웹 서버 개인정보를 탐지한다.

[출처: 보안뉴스(<https://www.boannews.com/media/view.asp?idx=46468&skind=0>)]

(파일 업로드 취약점 기반 공격의 유형)

파일 업로드에 대한 취약점은 웹 사이트 혹은 어플리케이션에 업로드 되는 파일의 크기, 유형, 개수를 제한하지 않는 경우 발생한다. 또한 업로드 된 파일을 외부에서 직접접근이 가능할 경우 역시 취약점이 될 수 있다.

언어	확장자
asp, aspx	asp, aspx, htm, html, asa
php	phtml, php, php3, php4, php5, inc, htm, html
jsp, java	jsp, jsp, jsw, js, jspf, htm, html
perl	pl, pm, cgi, lib, htm, html
coldfusion	cfm, cfml, cfc, dbm, htm, html

이러한 악성 파일이 업로드 된다면 접근권한, 정보유출, 악성코드 배포 등의 문제를 야기할 수 있다.

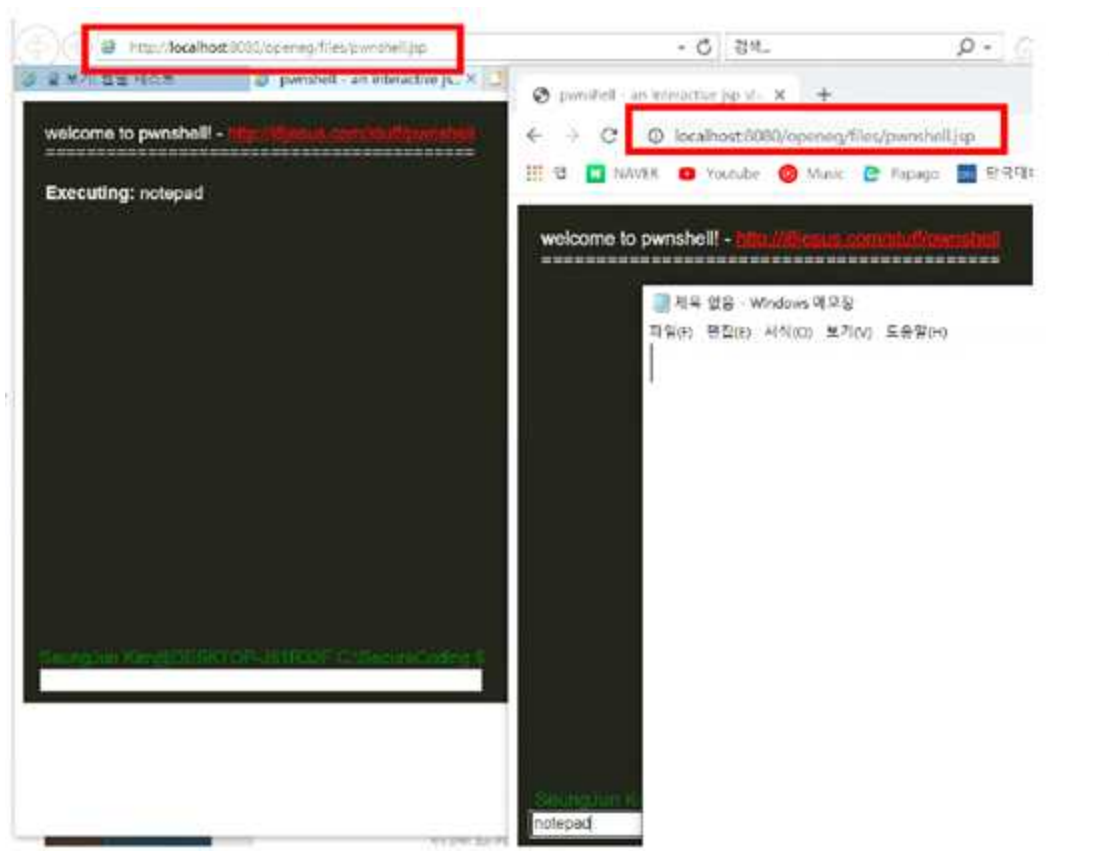
[직접접근이 가능한 경우]

업로드한 파일에 직접접근이 가능한 경우 공격자는 이를 통해 데이터베이스에 존재하는 다른 파일 또한 접근할 수 있게 되고 이는 개인정보 유출로 이어질 수 있다.

[웹셸 공격]

웹셸 공격은 파일 업로드 공격에 있어서 가장 많이 일어나고 위험한 공격이다. 파일 업로드 기능을 이용하여 시스템에 명령을 내릴 수 있는 웹 프로그램을 업로드 할 수 있는 취약점으로, 간단한 서버 스크립트 (jsp,php,asp)로 만드는 방법이 널리 사용되며 파일 업로드 시 확장자에 대한 검증을 제대로 하지 않게 되면 이 스크립트로 만든 파일이 업로드 되는 것이다. 웹셸 설치 시 해커들은 보안 시스템을 피하여 별도의 인증 없이 시스템에 쉽게 접속이 가능하므로 매우 위험하다.

즉, 공격자가 조작한 server site script 파일을 업로드하고, 서버상에 저장된 경로를 유추하여 파일을 실행, 셸을 획득할 수 있게 되며 셸 권한을 획득한 후에 시스템 명령어를 홈페이지를 통하여 실행하고 그 결과값을 보며 시스템 관리자 권한을 획득 또는 인접 서버에 침입을 시도할 수 있으므로, 업로드 되는 파일에 대한 검증 및 업로드 폴더에 대한 접근통제가 필요하다.



[NULL을 이용한 파일 업로드]

널(%00)문자가 문자의 끝을 의미하기 때문에 특정 확장자를 숨기기 위한 목적으로 사용될 수 있다. 해당 부분의 취약점은 내부 API를 호출할 때 발생할 가능성이 있다.

널바이트(%00)와 .jpeg 확장자를 함께 접목하여 업로드하면 %00 다음의 문자열은 무시하게 된다.

언어	우회패턴	처리패턴
php	test.php%00.jpeg	test.php
asp	test.asp%00.jpeg	test.asp
jsp	test.jsp%00.jpeg	test.jsp

(보안 정보 공유 체계(CVE, CWE, NVD))

- CVE

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	894	177	112	172			2	7		25	16	103			2
2000	1020	257	208	206		2	4	20		48	19	139			
2001	1677	403	403	297		7	34	123		83	36	220		2	2
2002	2156	498	553	435	2	41	200	103		127	76	199	2	14	1
2003	1527	381	477	372	2	50	129	60	1	62	69	144		16	5
2004	2451	580	614	408	3	148	291	111	12	145	96	134	5	38	5
2005	4935	838	1627	657	21	604	786	202	15	289	261	221	11	100	14
2006	6610	893	2719	664	91	967	1302	322	8	267	272	184	16	849	30
2007	6520	1101	2601	955	95	706	883	338	14	267	326	242	69	700	45
2008	5632	894	2310	699	128	1101	807	362	7	288	268	188	83	170	76
2009	5736	1035	2185	698	188	963	851	323	9	337	302	223	119	138	738
2010	4653	1102	1714	676	342	520	605	276	8	234	284	238	86	73	1501
2011	4155	1221	1334	735	351	294	470	108	7	197	411	206	58	17	557
2012	5297	1425	1459	833	423	243	759	122	13	344	392	250	166	14	623
2013	5191	1455	1186	858	366	156	650	110	7	352	512	274	123	1	208
2014	7939	1599	1572	841	420	304	1103	204	12	457	2106	239	264	2	403
2015	6504	1793	1830	1084	749	221	784	151	12	577	753	366	248	5	129
2016	6454	2029	1486	1313	717	94	498	99	15	444	870	602	86	7	1
2017	14714	3155	3004	2495	745	508	1518	279	11	629	1659	459	327	18	6
2018	16557	1853	3041	2121	400	517	2048	545	11	708	1239	247	461	31	4
2019	17344	1342	3201	1270	488	549	2390	465	10	710	983	202	535	57	13
2020	18325	1351	3248	1618	409	460	2178	401	14	966	1345	310	402	37	62
2021	7986	800	1663	680	143	233	935	190	1	339	404	123	167	16	
Total	154277	26182	38557	20085	6083	8688	19227	4921	187	7895	12699	5513	3226	2305	4423
% Of All		17.0	25.0	13.0	3.9	5.6	12.5	3.2	0.1	5.1	8.2	3.6	2.1	1.5	

전체 사이버 보안 공격 중 1.5% 차지하고 있는 것을 확인할 수 있었다. 파일 업로드 취약점은 2014년쯤 확산되었던 공격 방법이라 지금은 많이 보완되어 다른 유형의 사고에 비해 비중을 적게 차지하고 있다고 예상해 볼 수 있었다.

Security Vulnerabilities (File Inclusion)

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9


Sort Results By: CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2021-33408			File Inclusion	2021-05-27	2021-05-27	0.0	None	???	???	???	???	???	???
Local File Inclusion vulnerability in Ab Initio Control>Center before 4.0.2.6 allows remote attackers to retrieve arbitrary files. Fixed in v4.0.2.6 and v4.0.3.1.														
2	CVE-2021-32100			File Inclusion	2021-05-07	2021-05-14	4.0	None	Remote	Low	???	Partial	None	None
A remote file inclusion vulnerability exists in Artica Pandora FMS 742; exploitable by the lowest privileged user.														
3	CVE-2021-31783	345		File Inclusion	2021-04-26	2021-05-04	5.0	None	Remote	Low	Not required	Partial	None	None
show_default.php in the LocalFilesEditor extension before 11.4.0.1 for Piwigo allows Local File Inclusion because the file parameter is not validated with a proper regular-expression check.														
4	CVE-2021-30173	36		File Inclusion	2021-05-07	2021-05-18	4.0	None	Remote	Low	???	Partial	None	None
Local File Inclusion vulnerability of the omni-directional communication system allows remote authenticated attacker inject absolute path into Url parameter and access arbitrary file.														
5	CVE-2021-27236	94		Exec Code File Inclusion	2021-02-16	2021-02-22	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
An issue was discovered in Mutare Voice (EVM) 3.x before 3.3.8. getFile.asp allows Unauthenticated Local File Inclusion, which can be leveraged to achieve Remote Code Execution.														
6	CVE-2021-24242	22		Dir. Trav. File Inclusion	2021-04-22	2021-04-30	5.5	None	Remote	Low	???	Partial	Partial	None
The Tutor LMS 86" eLearning and online course solution WordPress plugin before 1.8.8 is affected by a local file inclusion vulnerability through the maliciously constructed sub_page parameter of the plugin's Tools, allowing high privilege users to include any local php file														
7	CVE-2021-23340	22		Dir. Trav. File Inclusion	2021-02-18	2021-02-25	5.5	None	Remote	Low	???	Partial	Partial	None
This affects the package pimcore/pimcore before 6.8.8. A Local File Inclusion vulnerability exists in the downloadCsvAction function of the CustomReportController class (bundles/AdminBundle/Controller/Reports/CustomReportController.php). An authenticated user can reach this function with a GET request at the following endpoint: /admin/reports/custom-report/download-csv?exportFile=891;filename). Since exportFile variable is not sanitized, an attacker can exploit a local file inclusion vulnerability.														
8	CVE-2020-35942	352		Exec Code XSS Bypass CSRF File Inclusion	2021-02-09	2021-02-12	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
A Cross-Site Request Forgery (CSRF) issue in the NextGEN Gallery plugin before 3.5.0 for WordPress allows File Upload and Local File Inclusion via settings modification, leading to Remote Code Execution and XSS. (It is possible to bypass CSRF protection by simply not including a nonce parameter.)														
9	CVE-2020-35580	522		File Inclusion	2021-05-20	2021-05-28	5.0	None	Remote	Low	Not required	Partial	None	None
A local file inclusion vulnerability in the FileServlet in all SearchBlox before 9.2.2 allows remote, unauthenticated users to read arbitrary files from the operating system via a /searchblox/servlet/FileServlet?col=url=request. Additionally, this may be used to read the contents of the SearchBlox configuration file (e.g., searchblox/WEB-INF/config.xml), which contains both the Super Admin's API key and the base64 encoded SHA1 password hashes of other SearchBlox users.														
10	CVE-2020-35566	706		File Inclusion	2021-02-16	2021-02-19	5.0	None	Remote	Low	Not required	Partial	None	None
An issue was discovered in MB CONNECT LINE mymbCONNECT24 and mbCONNECT24 through 2.6.2. An attacker can read arbitrary JSON files via Local File Inclusion.														
11	CVE-2020-29279			Exec Code File Inclusion	2020-12-02	2020-12-04	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
PHP remote file inclusion in the assign_resume_tpl method in Application/Common/Controller/BaseController.class.php in 74CMS before 6.0.48 allows remote code execution.														

CVE ID를 살펴보면 'CVE-년도-번호'인 형태로 저장되어 있는 것으로 보아 해당 사건 발생 년도와 해당 사건의 번호로 관리되어 지고 있는 것을 확인할 수 있다. 또한 CWE ID가 적혀있는 것으로 보아 CWE와 연개하여 관리되어지고 있다는 것 또한 확인할 수 있었고 Score를 통해 해당 사건의 위험도를 점수로 표시하고 있는 것을 알 수 있었다.

- CWE


Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

Home > CWE List > CWE- Individual Dictionary Definition (4.4)

Home | About | **CWE List** | Scoring | Community | News | Guidance | Search

CWE VIEW: Weaknesses in the 2020 CWE Top 25 Most Dangerous Software Weaknesses

View ID: 1350
Types: Graph

Objective
CWE entries in this view are listed in the 2020 CWE Top 25 Most Dangerous Software Weaknesses.

Audience

Stakeholder	Description
Software Developers	By following the CWE Top 25, developers are able to significantly reduce the number of weaknesses that occur in their software.
Product Customers	Customers can use the weaknesses in this view in order to formulate independent evidence of a claim by a product vendor to have eliminated / mitigated the most d
Educators	Educators can use this view to focus curriculum and teachings on the most dangerous weaknesses.


Relationships
The following graph shows the tree-like relationships between weaknesses that exist at different levels of abstraction. At the highest level, categories and pillars exist to group weaknesses; special CWE entries used to group weaknesses that share a common characteristic. Pillars are weaknesses that are described in the most abstract fashion. Below these top-level entries are still very abstract, typically independent of any specific language or technology. Base level weaknesses are used to present a more specific type of weakness. A variant is a weakness that a specific language or technology. A chain is a set of weaknesses that must be reachable consecutively in order to produce an exploitable vulnerability. While a composite is a set of weakn produce an exploitable vulnerability.

Expand All | Collapse All

1350 - Weaknesses in the 2020 CWE Top 25 Most Dangerous Software Weaknesses

- Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') - (79)
- Out-of-bounds Write - (787)
- Improper Input Validation - (20)
- Out-of-bounds Read - (125)
- Improper Restriction of Operations within the Bounds of a Memory Buffer - (119)
- Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') - (89)
- Exposure of Sensitive Information to an Unauthorized Actor - (200)
- Use After Free - (416)
- Cross-Site Request Forgery (CSRF) - (352)
- Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') - (78)
- Integer Overflow or Wraparound - (190)
- Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') - (22)
- NULL Pointer Dereference - (476)
- Improper Authentication - (287)
- **Unrestricted Upload of File with Dangerous Type - (434)**
- Incorrect Permission Assignment for Critical Resource - (732)
- Improper Control of Generation of Code ('Code Injection') - (94)
- Insufficiently Protected Credentials - (522)
- Improper Restriction of XML External Entity Reference - (611)

CWE를 살펴보면 2020 CWE의 약점 Top 25 가장 위험한 소프트웨어 약점에 File Upload가 포함되어 있는 것을 확인할 수 있고 이를 통해 File Upload가 얼마나 위험하고 흔하게 발생하는 위험인지를 인지할 수 있다.


Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

Home > CWE List > CWE- Individual Dictionary Definition (4.4)

Home | About | **CWE List** | Scoring | Community | News | Guidance | Search

ID Lookup: Go

CWE-434: Unrestricted Upload of File with Dangerous Type

Weakness ID: 434
Abstraction: Base
Structure: Simple
Status: Draft

Presentation Filter:

CWE에서는 File Upload 사고에 대해서 CWE-434번으로 번호를 부여한 것을 확인할 수 있었다.

▼ Demonstrative Examples

Example 1

The following code intends to allow a user to upload a picture to the web server. The HTML code that drives the form on the user end has an input field of type "file".

Example Language: **HTML**

(good code)

```
<form action="upload_picture.php" method="post" enctype="multipart/form-data">

Choose a file to upload:
<input type="file" name="filename"/>
<br/>
<input type="submit" name="submit" value="Submit"/>

</form>
```

Once submitted, the form above sends the file to upload_picture.php on the web server. PHP stores the file in a temporary location until it is retrieved (or discarded) by the server side code. In this example, the file is moved to a more permanent pictures/ directory.

Example Language: **PHP**

(bad code)

```
// Define the target location where the picture being
// uploaded is going to be saved.
$target = "pictures/" . basename($_FILES['uploadedfile']['name']);

// Move the uploaded file to the new location.
if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target))
{
    echo "The picture has been successfully uploaded.";
}
else
{
    echo "There was an error uploading the picture, please try again.";
}
```

The problem with the above code is that there is no check regarding type of file being uploaded. Assuming that pictures/ is available in the web document root, an attacker could upload a file with the name:

malicious.php

(attack code)

Since this filename ends in ".php" it can be executed by the web server. In the contents of this uploaded file, the attacker could use:

아래쪽을 살펴보면 해당 유형의 File Upload에 대한 다양한 공격방법의 예시를 확인할 수 있다.



VULNERABILITIES

CVE-2021-3395 Detail

Current Description

A cross-site scripting (XSS) vulnerability in Pryaniki 6.44.3 allows remote authenticated users to upload an arbitrary file. The JavaScript code will execute when someone visits the attachment.

[View Analysis Description](#)

Severity

CVSS Version 3.x

CVSS Version 2.0

CVSS 3.x Severity and Metrics:



NIST: NVD

Base Score: 5.4 MEDIUM

Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

QUICK INFO

CVE Dictionary Entry:

CVE-2021-3395

NVD Published Date:

02/02/2021

NVD Last Modified:

02/04/2021

Source:

MITRE

NVD를 통해 File Upload 사건인 CVE-2021-3395의 Score를 확인할 수 있다.



CVE and NVD Relationship

CVE and NVD Are Two Separate Programs

The [CVE List](#) was launched by [MITRE](#) as a community effort in 1999, and the [U.S. National Vulnerability Database \(NVD\)](#) was launched by the [National Institute of Standards and Technology \(NIST\)](#) in 2005.

- **CVE** - A list of records—each containing an identification number, a description, and at least one public reference—for publicly known cybersecurity vulnerabilities. CVE Records are used in numerous [cybersecurity products and services](#) from around the world, including NVD.
- **NVD** - A vulnerability database built upon and fully synchronized with the CVE List so that any updates to CVE appear immediately in NVD.
- **Relationship** - The CVE List feeds NVD, which then builds upon the information included in CVE Records to provide enhanced information for each record such as fix information, severity scores, and impact ratings. As part of its enhanced information, NVD also provides advanced searching features such as by OS; by vendor name, product name, and/or version number; and by vulnerability type, severity, related exploit range, and impact.

While separate, both CVE and NVD are sponsored by the [U.S. Department of Homeland Security \(DHS\) Cybersecurity and Infrastructure Security Agency \(CISA\)](#), and both are available to the public and free to use.

CVE는 NVD에게 공격 리스트를 공급하고 CVE 레코드에 포함 된 정보를 기반으로 수정 정보, 심각도 점수 및 영향 등급과 같은 각 레코드에 대한 향상된 정보를 제공한다. 향상된 정보의 일부로 NVD는 OS와 공급 업체 이름, 제품 이름 및 / 또는 버전 번호 취약성 유형, 심각도, 관련 익스플로잇 범위 및 영향에 따른 고급 검색 기능도 제공한다.

(파일 업로드 공격 유형 별 방어 대책)

1. 확장자 검사
2. 대소문자 구분하지 않고 확장자 비교
3. 특수문자가 포함된 경우 업로드 금지
4. 업로드된 파일명, 확장자를 난수화하여 변경
5. 업로드된 파일을 url 요청으로 직접 접근이 불가능한 위치에 저장

파일 업로드 취약점 기반 공격은 웹 APP에 업로드 되는 파일의 타입을 제한하는 것으로 파일 타입 검사나 확장자 검사를 통해 안전하게 업로드해야 한다.

외부에서 직접 접근이 불가능하도록 업로드 한 파일을 다시보거나 요청을 했을 때, 컨트롤러 컴포넌트에 의해 정확한 인증, 인가 정책을 수행하고 실행할 수 있도록 설계해야 한다.

업로드 파일의 크기를 제한하는 방법을 사용하여 서버에 업로드 되는 파일의 용량을 제한해야 한다. 소스코드에서 메소드를 사용하여 제한할 수 있다.

```
@RequestMapping(value="/write.do", method=RequestMethod.POST)
public String boardWriteProc(@ModelAttribute("BoardModel") BoardModel boardModel, MultipartHttpServletRequest request, HttpSession session){
    //파일저장 위치
    String uploadPath = session.getServletContext().getRealPath("/")
        + "WEB-INF/files/";
    System.out.println("uploadPath: " + uploadPath);
    MultipartFile file = request.getFile("file");
    //업로드 되는 파일 사이즈 제한
    if ( file != null && !"".equals(file.getOriginalFilename())
        && file.getSize() < 1024000 && file.getContentType().contains("image")){
        //업로드 파일명
        String fileName = file.getOriginalFilename();
        if ( fileName.toLowerCase().endsWith(".jpg")){
            //저장할 파일명을 랜덤하게 생성하여 사용한다
            String savedFileName = UUID.randomUUID().toString();
            File uploadFile = new File(uploadPath+savedFileName);
            try {
                file.transferTo(uploadFile); //실제 파일이 저장되는 리언
            } catch (Exception e) {
                System.out.println("upload error");
            }
            boardModel.setFileName(fileName);
            boardModel.setSavedFileName(savedFileName);
        }
    }

    String content = boardModel.getContent().replaceAll("<br>", "<br/>");
    boardModel.setContent(content);

    service.writeArticle(boardModel);

    return "redirect:list.do";
}
```

BoardController 소스코드에서 특정 게시물을 작성하는 부분에서 업로드 되는 파일의 크기를 특정 크기 이하로 제한하고 타입을 특정 타입만 업로드 할 수 있도록 제한한다. 여기서 1M 이하로 용량을 제한하고 확장자는 jpg로 설정한다.

```
//added code
@RequestMapping("/get_image.do")
public void getImage(HttpServletRequest request, HttpSession session,
    HttpServletResponse response){
    int idx=TestUtil.getInt(request.getParameter("idx"));
    // 읽어본 게시물인지 확인
    if ( session.getAttribute("idx") == null ||
        (Integer)session.getAttribute("idx") != idx ) {
        return;
    }
    // 저장된 파일명을 읽어오는 작업이 필요
    BoardModel board = service.getOneArticle(idx);

    String filePath=session.getServletContext().getRealPath("/") +
        "WEB-INF/files/" +board.getSavedFileName();

    System.out.println("filename: "+filePath);
    BufferedOutputStream out=null;
    InputStream in=null;
    try {
        response.setContentType("image/jpeg");
        response.setHeader("Content-Disposition", "inline;filename="+board.getFileName());
        File file=new File(filePath);
        in=new FileInputStream(file);
        out=new BufferedOutputStream(response.getOutputStream());
        int len;
        byte[] buf=new byte[1024];
        while ( (len=in.read(buf)) > 0) {
            out.write(buf,0,len);
        }
    }catch(Exception e){
        e.printStackTrace();
        System.out.println("파일 전송 에러");
    }finally {
        if ( out != null ) try { out.close(); }catch(Exception e){}
        if ( in != null ) try { in.close(); }catch(Exception e){}
    }
}
```

읽어본 게시물인지 확인하고 저장된 파일명을 읽어오는 작업이 필요하다.

업로드한 파일의 저장 위치 및 파일을 안전하게 얻어올 수 있도록 코드를 작성한다.

```
<td colspan="4" align="left">
<span class="date">첨부파일:&nbsp;  
<a href="get_image.do?idx=${board.idx}"
target="_blank">${board.fileName}</a></span></td>
```

view.jsp 파일에서 위에 get_image.do 명령이 수행될 수 있도록 설정한다.