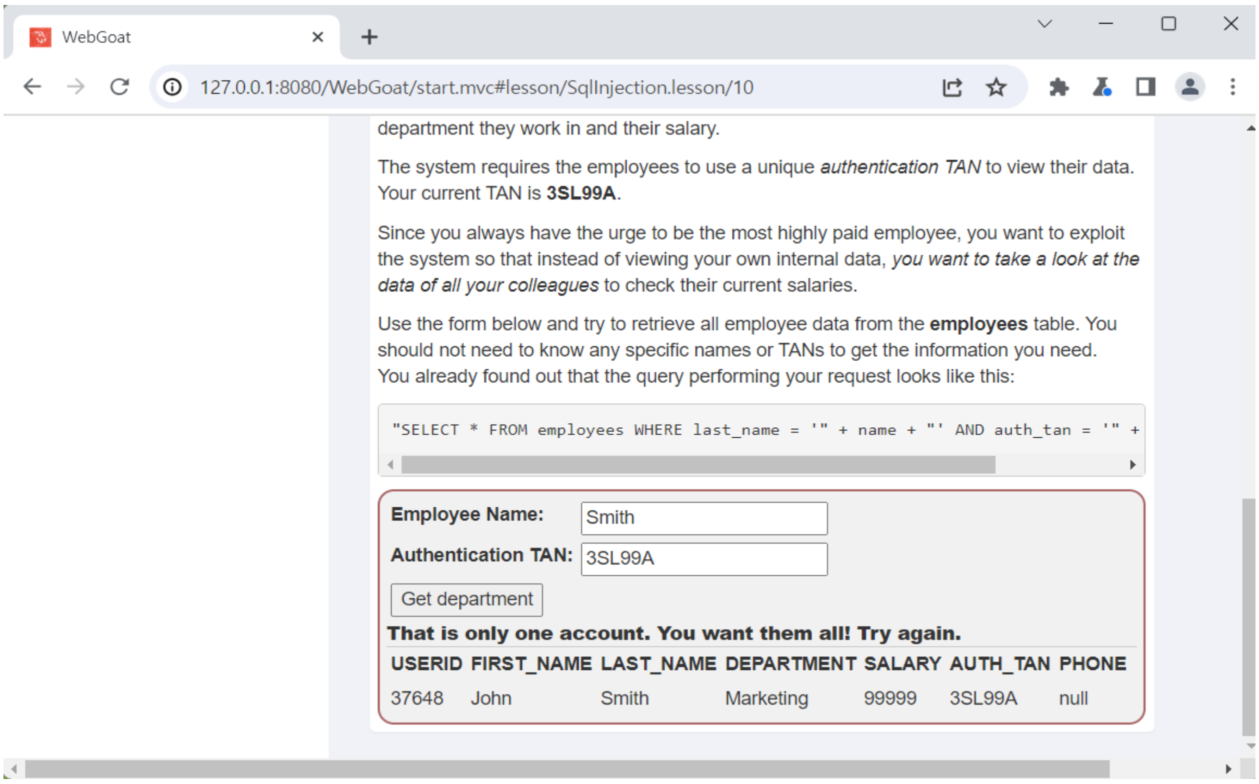
인터넷응용보안 6주차 과제

202121556 곽지현

SQL Injection (intro) 11번.

처음에 주어진 Smith 라는 이름과 3SL99A TAN 정보를 넣어 Smith 계정 정보를 출력



127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/10

department they work in and their salary.

The system requires the employees to use a unique *authentication TAN* to view their data. Your current TAN is **3SL99A**.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, *you want to take a look at the data of all your colleagues* to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.
You already found out that the query performing your request looks like this:

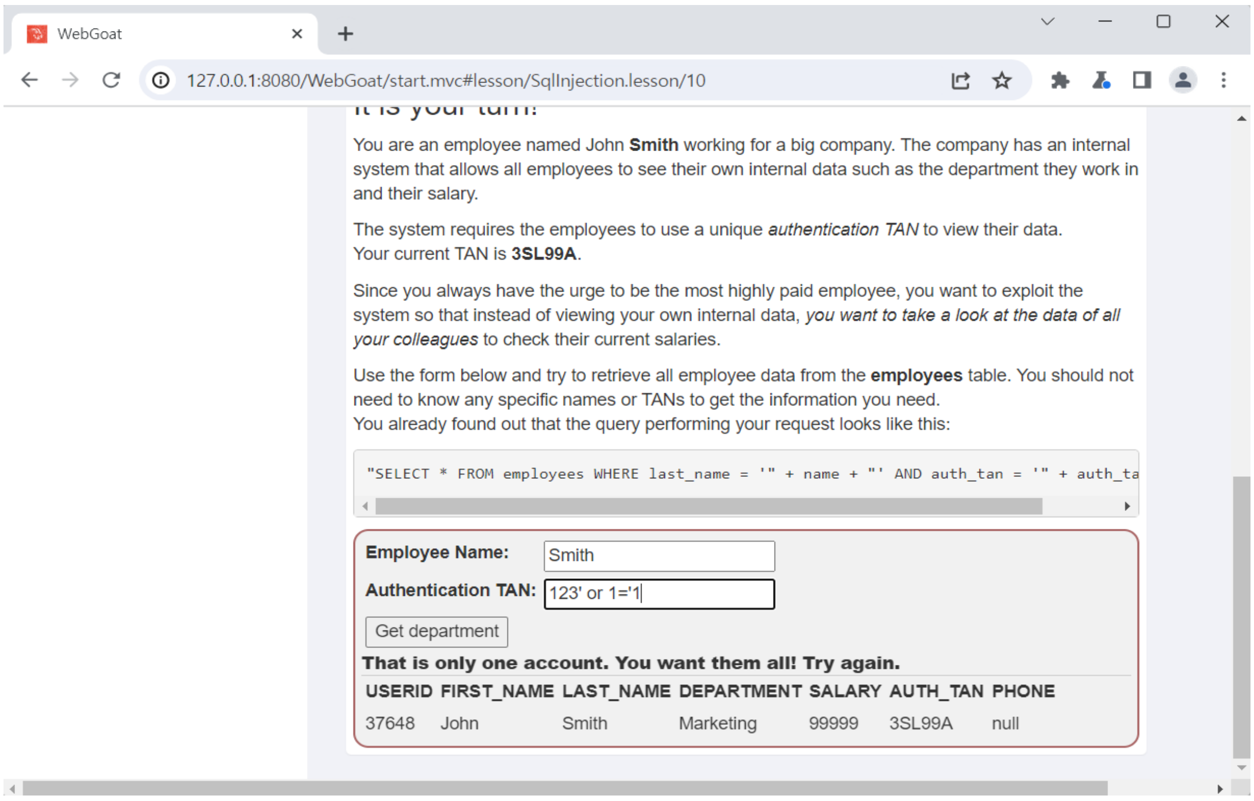
```
"SELECT * FROM employees WHERE last_name = '" + name + "' AND auth_tan = '" +
```

Employee Name:	Smith
Authentication TAN:	3SL99A

That is only one account. You want them all! Try again.

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	99999	3SL99A	null

Employee Name 에 Smith 를 넣고 Authentication TAN 에 123' or 1='1 넣기



127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/10

You are an employee named John **Smith** working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary.

The system requires the employees to use a unique *authentication TAN* to view their data.
Your current TAN is **3SL99A**.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, *you want to take a look at the data of all your colleagues* to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.
You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = '" + name + "' AND auth_tan = '" + auth_ta
```

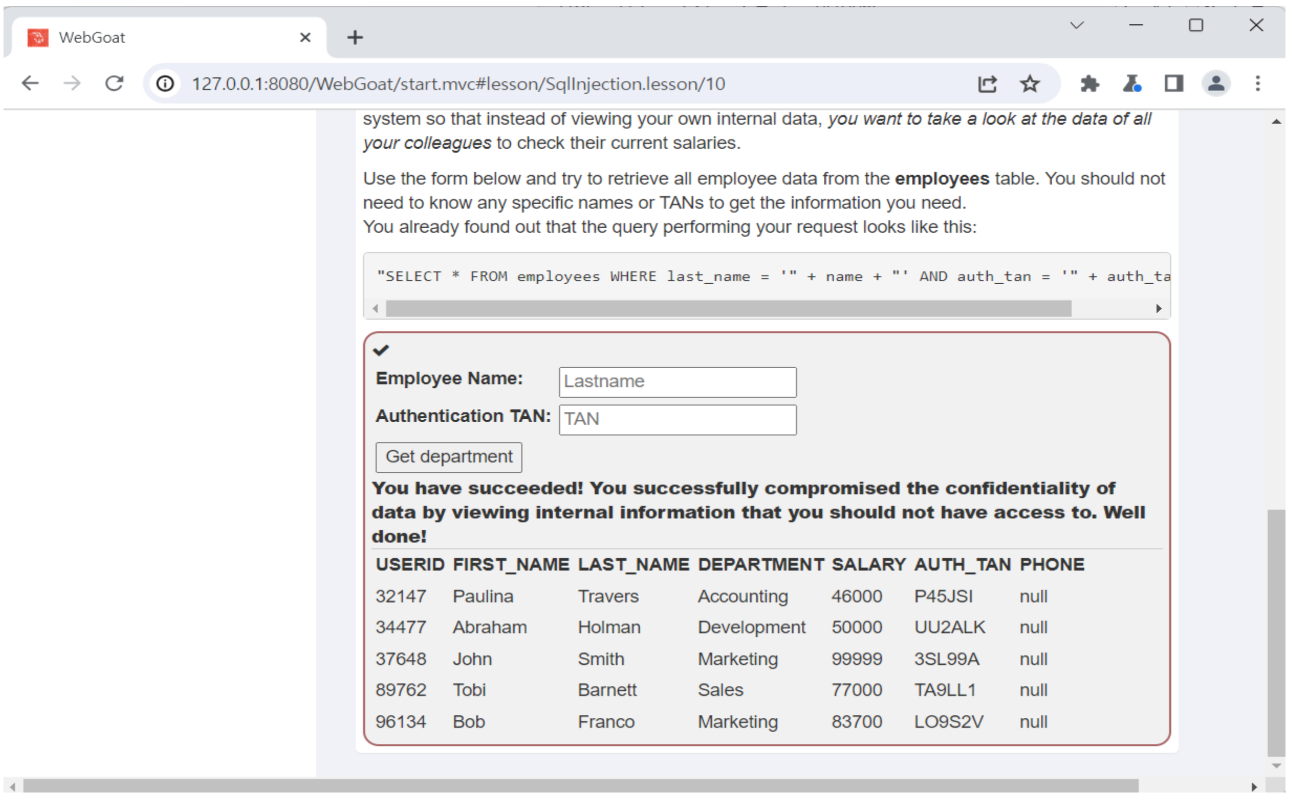
Employee Name:	Smith
Authentication TAN:	123' or 1='1

That is only one account. You want them all! Try again.

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	99999	3SL99A	null

직원 테이블에서 모든 직원 데이터를 출력 완료

11번 성공!


WebGoat
127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/10

system so that instead of viewing your own internal data, *you want to take a look at the data of all your colleagues* to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.
You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = '" + name + "' AND auth_tan = '" + auth_ta
```

✓

Employee Name:	Lastname
Authentication TAN:	TAN

Get department

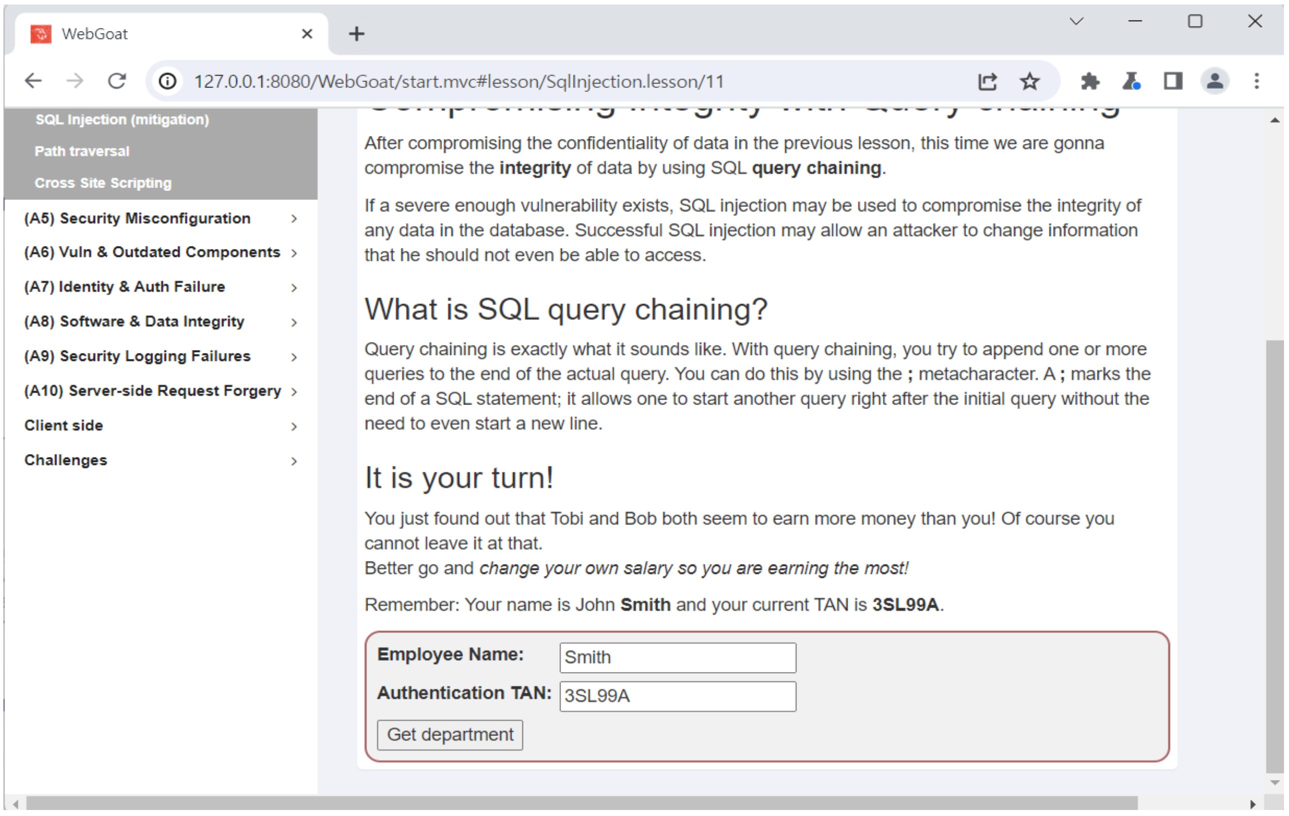
You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
37648	John	Smith	Marketing	99999	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

SQL Injection (intro) 12번.

처음에 주어진 Smith 라는 이름과 3SL99A TAN 정보를 넣어 Smith 계정 정보를 출력

중요한 정보인 DB 스키마가 표출


WebGoat
127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/11
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges

After compromising the confidentiality of data in the previous lesson, this time we are gonna compromise the **integrity** of data by using SQL **query chaining**.

If a severe enough vulnerability exists, SQL injection may be used to compromise the integrity of any data in the database. Successful SQL injection may allow an attacker to change information that he should not even be able to access.

What is SQL query chaining?

Query chaining is exactly what it sounds like. With query chaining, you try to append one or more queries to the end of the actual query. You can do this by using the **;** metacharacter. A **;** marks the end of a SQL statement; it allows one to start another query right after the initial query without the need to even start a new line.

It is your turn!

You just found out that Tobi and Bob both seem to earn more money than you! Of course you cannot leave it at that.
Better go and *change your own salary so you are earning the most!*

Remember: Your name is John **Smith** and your current TAN is **3SL99A**.

Employee Name:	Smith
Authentication TAN:	3SL99A

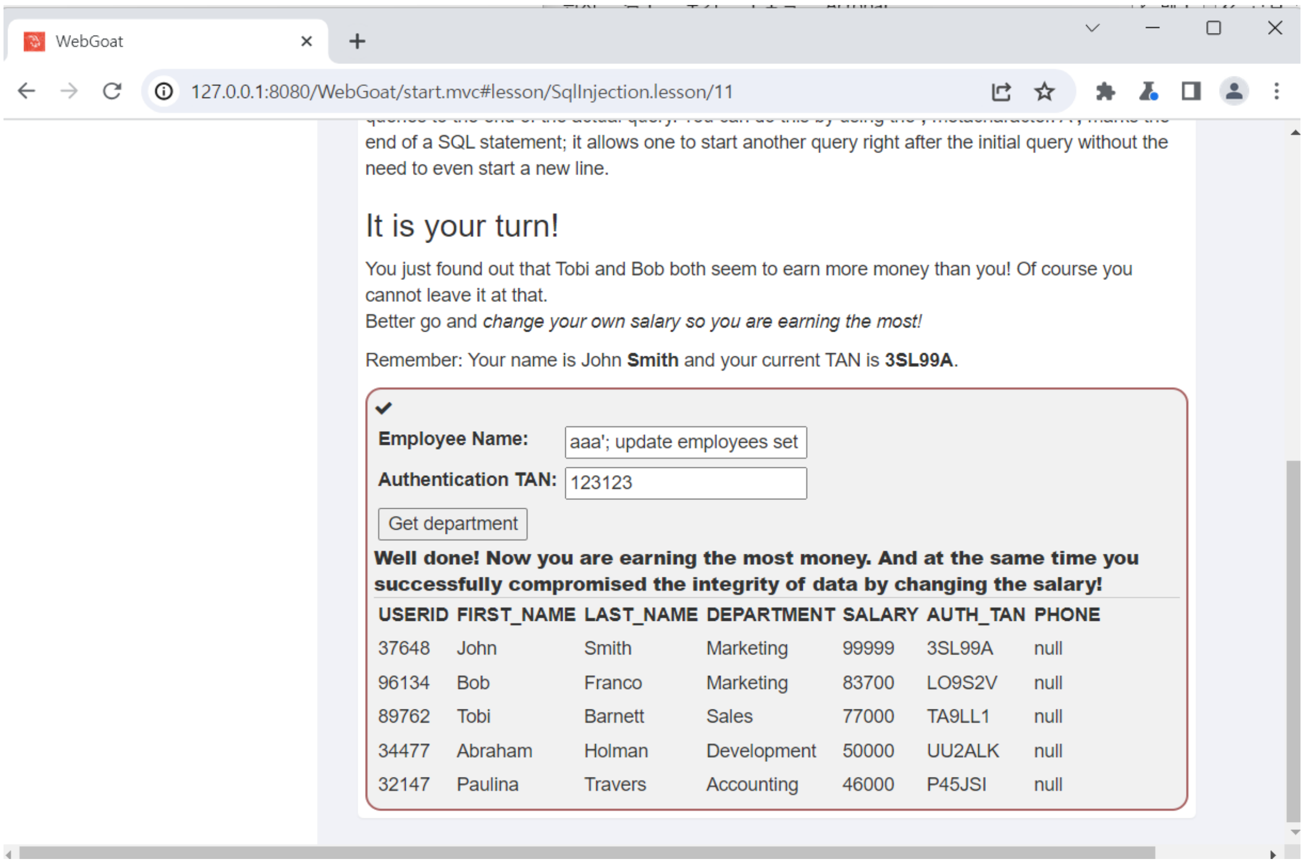
Get department

세미콜론을 써서 이전 문장을 수행, 추가적인 월급 변경 코드 삽입

aaa'; update employees set SALARY=99999 where LAST_NAME='Smith' and AUTH_TAN='3SL99A' –

Authentication TAN 에는 아무 데이터 입력 (123123)

12번 성공!



WebGoat
127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/11

end of a SQL statement; it allows one to start another query right after the initial query without the need to even start a new line.

It is your turn!

You just found out that Tobi and Bob both seem to earn more money than you! Of course you cannot leave it at that.
Better go and *change your own salary so you are earning the most!*

Remember: Your name is John **Smith** and your current TAN is **3SL99A**.

Employee Name:	aaa'; update employees set
Authentication TAN:	123123

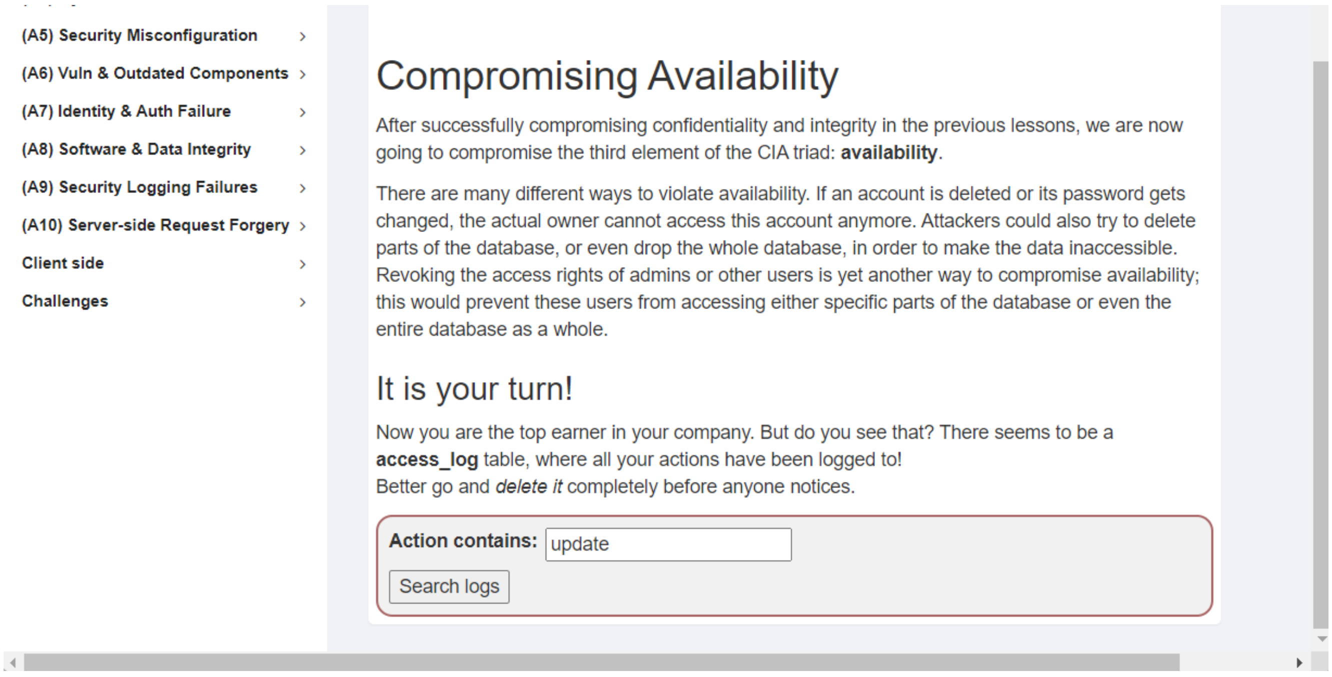
Get department

Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	99999	3SL99A	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null

SQL Injection (intro) 13번.

update 명령어 입력해 로그에서 update 기록 찾기 (월급을 변경한 로그가 남아있음)



(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges

Compromising Availability

After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: **availability**.

There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability; this would prevent these users from accessing either specific parts of the database or even the entire database as a whole.

It is your turn!

Now you are the top earner in your company. But do you see that? There seems to be a **access_log** table, where all your actions have been logged to!
Better go and *delete it* completely before anyone notices.

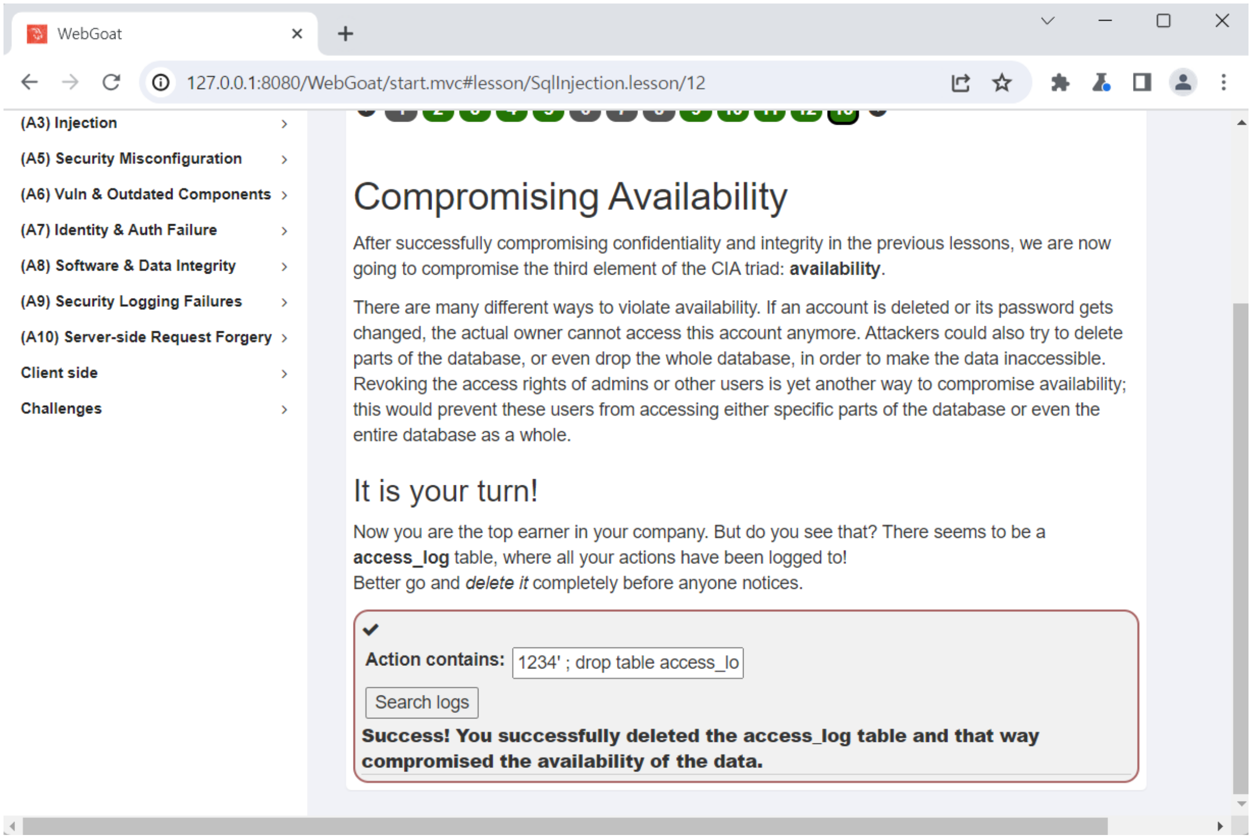
Action contains: update

Search logs

Drop 명령어를 이용해 전체 테이블을 삭제

1234' ; drop table access_log –

13번 성공!


WebGoat
127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/12
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges

Compromising Availability

After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: **availability**.

There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability; this would prevent these users from accessing either specific parts of the database or even the entire database as a whole.

It is your turn!

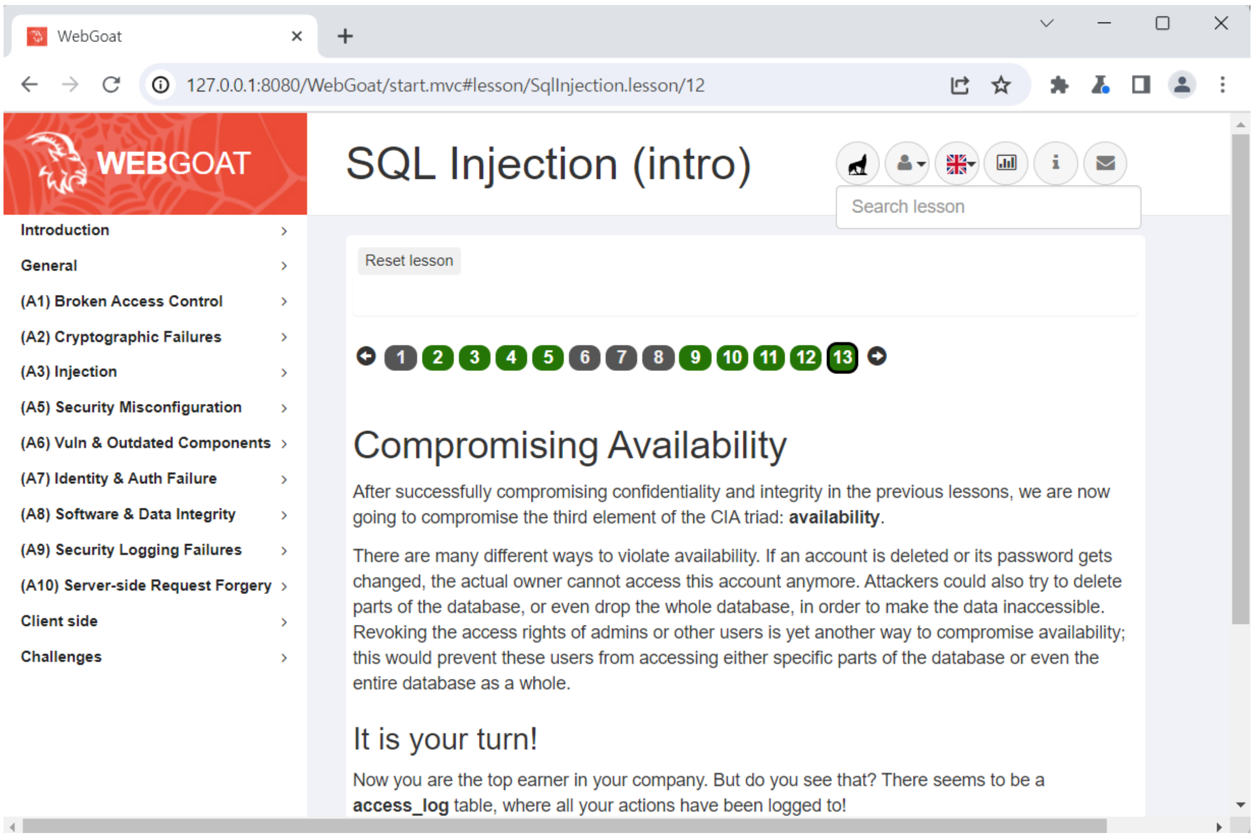
Now you are the top earner in your company. But do you see that? There seems to be a **access_log** table, where all your actions have been logged to!
Better go and *delete it* completely before anyone notices.

✓
Action contains: 1234' ; drop table access_lo

Search logs

Success! You successfully deleted the access_log table and that way compromised the availability of the data.

모든 문제 성공!


WebGoat
127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/12
WEBGOAT
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Search lesson
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13

SQL Injection (intro)

Compromising Availability

After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: **availability**.

There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability; this would prevent these users from accessing either specific parts of the database or even the entire database as a whole.

It is your turn!

Now you are the top earner in your company. But do you see that? There seems to be a **access_log** table, where all your actions have been logged to!