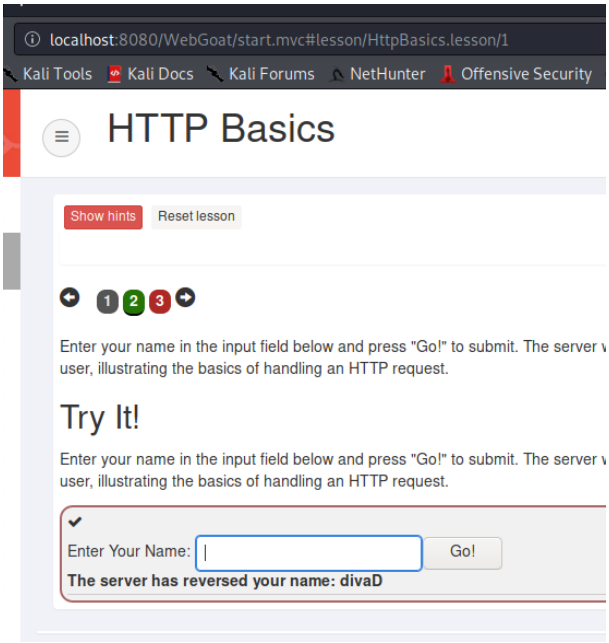
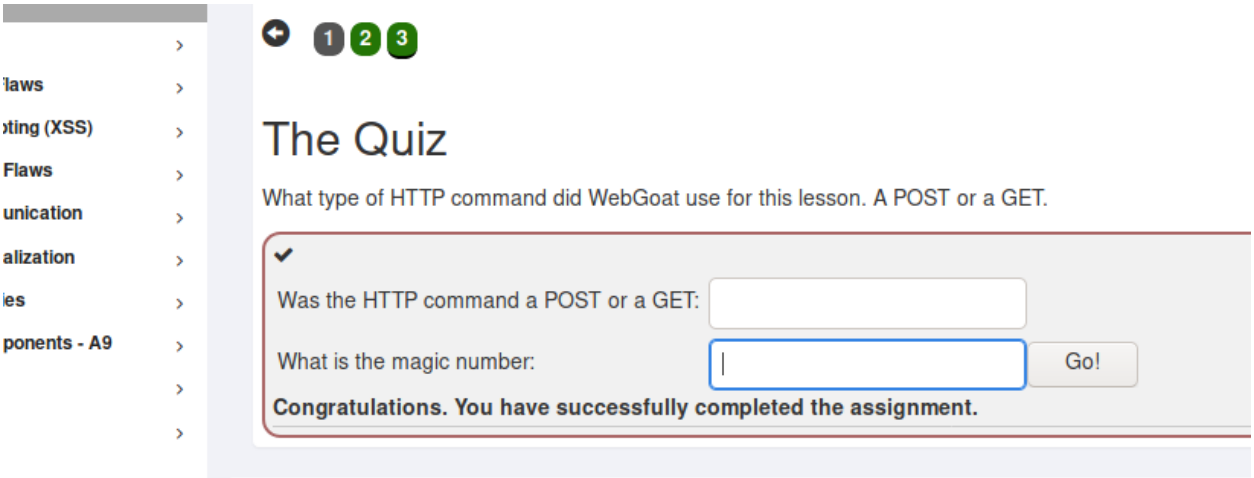
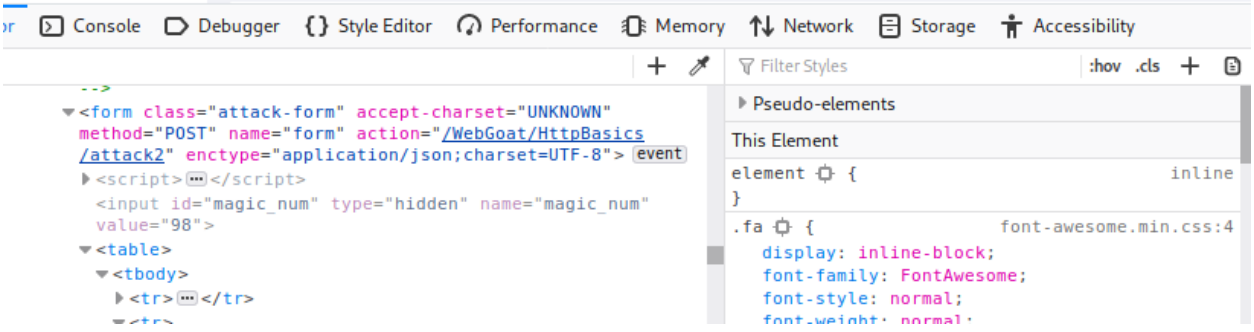
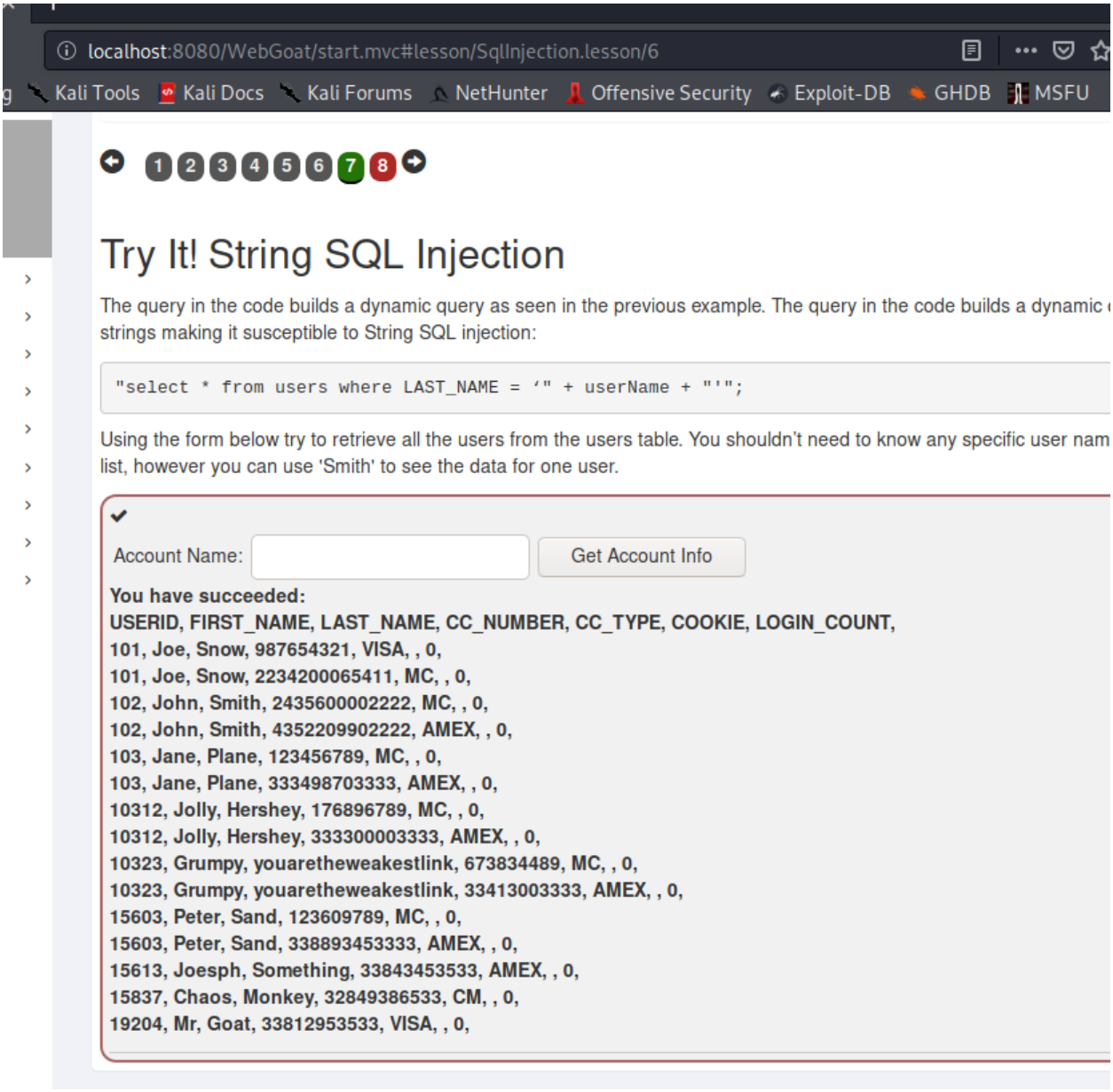
David Thomsen
SEC-260








localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/6
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
GHDB
MSFU
1 2 3 4 5 6 7 8
Try It! String SQL Injection
The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic
strings making it susceptible to String SQL injection:
"select * from users where LAST_NAME = '" + userName + "'";
Using the form below try to retrieve all the users from the users table. You shouldn't need to know any specific user nam
list, however you can use 'Smith' to see the data for one user.
Account Name:
Get Account Info
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,

Try It! Numeric SQL Injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dyna
a number making it susceptible to Numeric SQL injection:

```
"select * from users where USERID = "  + userID;
```

Using the form below try to retrieve all the users from the users table. You shouldn't need to know any specific user
list, however you can use '101' to see the data for one user.

✔

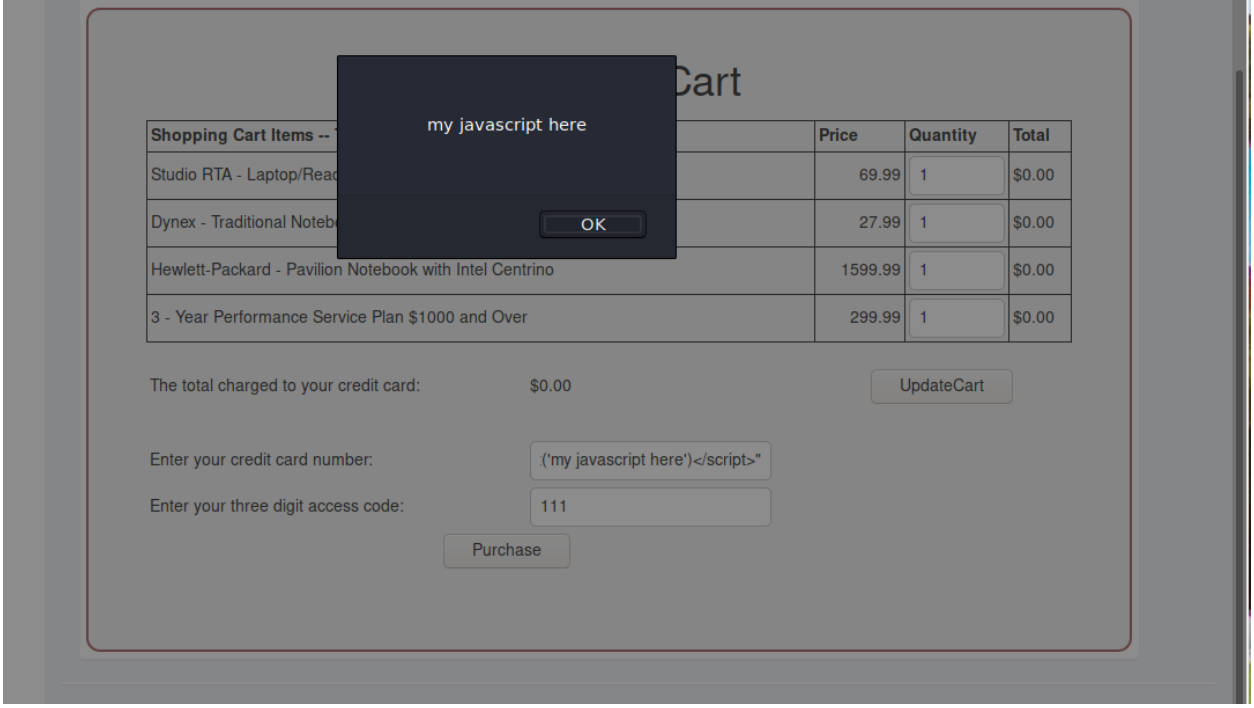
Name: Get Account Info

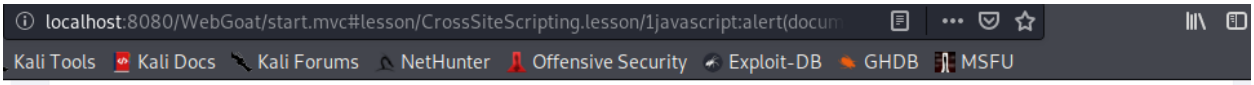
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,

Identify which field is susceptible to XSS

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

Make sure to include in your attack payload "<script>alert('my javascript here')</script>".





Cross-site script (also commonly known as XSS) is a vulnerability/flaw that combines ...# the allowance of html/script tags as input that are ...# rendred into a browser without encoding or sanitization

Cross site scripting (XSS) is the most prevalent and pernicious web application security issue

While there is a simple well-known defense for this attack, there are still many instances of it on the web. In terms of fixing it, coverage of fixes also tends to be a problem. We'll talk more about the defense in a little bit.

XSS has a significant impact

Especially as 'Rich Internet Applications' are more and more common place, privileged function calls linked to via javascript may be compromised. And if not properly protected, sensitive data (such as your authentication cookies) can be stolen and used for someone else's purpose.

Quick examples:

- From the browser address bar (chrome, Firefox)

```
javascript:alert("XSS Test");
javascript:alert(document.cookie);
```

- Any data field that is returned to the client is potentially injectable

```
<script>alert("XSS Test")</script>
```

Try It! Using Chrome or Firefox

- Open a second tab and use the same url as this page you are currently on (or any url within this instance of WebGoat)
- Then, in the address bar on each tab, type `javascript:alert(document.cookie);` **NOTE:** If you /cut/paste you'll need to add the `javascript:` back in.

Were the cookies the same on each tab? [ ] Submit

Congratulations. You have successfully completed the assignment.