

Roozah Khan
CYSE 450
Prof. Xu
Lab #2

a.)

☰

Cross Site Scripting

Logout

User: g01143245

Role: User

Version: 8.1.0

Build:

Show hints

Reset lesson

➡

1

2

3

4

5

6

7

8

9

10

11

12

➡

Try It! Reflected XSS

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.

An easy way to find out if a field is vulnerable to an XSS attack is to use the `alert()` or `console.log()` methods. Use one of them to find out which field is vulnerable.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tiltting Surface - Cherry	69.99	<input type="text" value="1"/>	\$0.00
Dynex - Traditional Notebook Case	27.99	<input type="text" value="1"/>	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	<input type="text" value="1"/>	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	<input type="text" value="1"/>	\$0.00

The total charged to your credit card:

\$0.00

UpdateCart

Enter your credit card number:

Enter your three digit access code:

Purchase

Well done, but alerts are not very impressive are they? Please continue.

Thank you for shopping at WebGoat.
You're support is appreciated

We have charged credit card:

.....

\$1997.96

b.)

Logout

User: g0t143245

Role: User

Version: 8.1.0

Build:

Show hints

Reset lesson

1

2

3

4

5

6

7

8

9

10

11

12

Cross Site Scripting

Identify potential for DOM-Based XSS

DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that are being "reflected" to the page.

For this example, you will want to look for some 'test' code in the route handlers (WebGoat uses backbone as its primary JavaScript library). Sometimes, test code gets left in production (and often times test code is very simple and lacks security or any quality controls).

Your objective is to find the route and exploit it. First though ... what is the base route? As an example, look at the URL for this lesson ... it should look something like /WebGoat/start.mvc#lesson/CrossSiteScripting/lesson/9. The 'base route' in this case is start.mvc#lesson/ The CrossSiteScripting.lesson/9 after that are parameters that are processed by the JavaScript route handler.

So, what is the route for the test code that stayed in the app during production? To answer this question, you have to check the JavaScript source.

GoatRouter.js

```
29     return $('#main-content');
30 };
31
32 function render(view) {
33     $('#div.pages').hide();
34     //TODO this works for now because we only have one page we should r
35     if (view != null) {
36         $('#report-card-page').show();
37     } else {
38         $('#lesson-title').show();
39         $('#lesson-page').show();
40     }
41 };
42
43 /*
44  * Definition of Goat App Router.
45  */
46 var GoatAppRouter = Backbone.Router.extend({
47
48     routes: {
49         'welcome': 'welcomeRoute',
50         'lesson/:name': 'lessonRoute',
51         'lesson/:name/:pageNum': 'lessonPageRoute',
52         'test/:param': 'testRoute',
53         'reportCard': 'reportCard'
54     },
55
56     lessonController: null,
57     menuController: null,
58     titleView: null,
59
60     setUpCustomJS: function () {
61
62 }
```

28 characters selected Coverage: n/a

Scope Watch

Logout

User: g0t143245

Role: User

Version: 8.1.0

Build:

Show hints

Reset lesson

1

2

3

4

5

6

7

8

9

10

11

12

Cross Site Scripting

Identify potential for DOM-Based XSS

DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that are being "reflected" to the page.

For this example, you will want to look for some 'test' code in the route handlers (WebGoat uses backbone as its primary JavaScript library). Sometimes, test code gets left in production (and often times test code is very simple and lacks security or any quality controls).

Your objective is to find the route and exploit it. First though ... what is the base route? As an example, look at the URL for this lesson ... it should look something like /WebGoat/start.mvc#lesson/CrossSiteScripting/lesson/9. The 'base route' in this case is start.mvc#lesson/ The CrossSiteScripting.lesson/9 after that are parameters that are processed by the JavaScript route handler.

So, what is the route for the test code that stayed in the app during production? To answer this question, you have to check the JavaScript source.

☒

Correct! Now, see if you can send in an exploit to that route in the next assignment.

GoatRouter.js

```
29     return $('#main-content');
30 };
31
32 function render(view) {
33     $('#div.pages').hide();
34     //TODO this works for now because we only have one page we should r
35     if (view != null) {
36         $('#report-card-page').show();
37     } else {
38         $('#lesson-title').show();
39         $('#lesson-page').show();
40     }
41 };
42
43 /*
44  * Definition of Goat App Router.
45  */
46 var GoatAppRouter = Backbone.Router.extend({
47
48     routes: {
49         'welcome': 'welcomeRoute',
50         'lesson/:name': 'lessonRoute',
51         'lesson/:name/:pageNum': 'lessonPageRoute',
52         'test/:param': 'testRoute',
53         'reportCard': 'reportCard'
54     },
55
56     lessonController: null,
57     menuController: null,
58     titleView: null,
59
60     setUpCustomJS: function () {
61
62 }
```

28 characters selected Coverage: n/a

Scope Watch

c.)

localhost:8080/WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome()<%2Fscript>

WEBGOAT

Introduction
General
(A1) Injection
(A2) Broken Authentication
(A3) Sensitive Data Exposure
(A4) XML External Entities (XXE)
(A5) Broken Access Control
(A7) Cross-Site Scripting (XSS)
(A8) Insecure Deserialization
(A9) Vulnerable Components
(A8-2013) Request Forgeries
Client side
Challenges

Show hints Reset lesson

test:

Logout
User: g0t1143245
Role: User
Version: 8.1.0
Build:

Logout
User: g0t1143245
Role: User
Version: 8.1.0
Build:

Elements Console Sources

top Filter Default levels 1 6 hidden

DevTools failed to load source map: Could not load content for <http://localhost:8080/WebGoat/js/libs/backbone-min.map>: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE

about to create app router
initialize goat app router
test handler
phoneHome invoked
phone home said
{"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","output":"phoneHome Response is -1462686731","assignment":"DOMCrossSiteScripting","attemptWasMade":true}

Show hints Reset lesson

test:

WEBGOAT

Cross Site Scripting

Show hints Reset lesson

1 2 3 4 5 6 7 8 9 10 11 12

Try It! DOM-Based XSS

Some attacks are "blind". Fortunately, you have the server running here so you will be able to tell if you are successful. Use the route you just found and see if you can use the fact that it reflects a parameter from the route without encoding to execute an internal function in WebGoat. The function you want to execute is ...

```
webgoat.customjs.phoneHome()
```

Sure, you could just use console/debug to trigger it, but you need to trigger it via a URL in a new tab. Once you do trigger it, a subsequent response will come to your browser's console with a random number. Put that random number in below.

1462686731 Submit

Logout
User: g0t1143245
Role: User
Version: 8.1.0
Build:



- Introduction
- General
- (A1) Injection
- (A2) Broken Authentication
- (A3) Sensitive Data Exposure
- (A4) XML External Entities (XXE)
- (A5) Broken Access Control
- (A7) Cross-Site Scripting (XSS)
- (A8) Insecure Deserialization
- (A9) Vulnerable Components
- (A8-2013) Request Forgeries
- Client side
- Challenges

Cross Site Scripting

Show hints Reset lesson



Try It! DOM-Based XSS

Some attacks are "blind". Fortunately, you have the server running here so you will be able to tell if you are successful. Use the route you just found and see if you can use the fact that it reflects a parameter from the route without encoding to execute an internal function in WebGoat. The function you want to execute is ...

`webgoat.customjs.phoneHome()`

Sure, you could just use console/debug to trigger it, but you need to trigger it via a URL in a new tab.

Once you do trigger it, a subsequent response will come to your browser's console with a random number. Put that random number in below.

Submit

Correct!

Logout

User: g01143245

Role: User

Version: 8.1.0

Build: