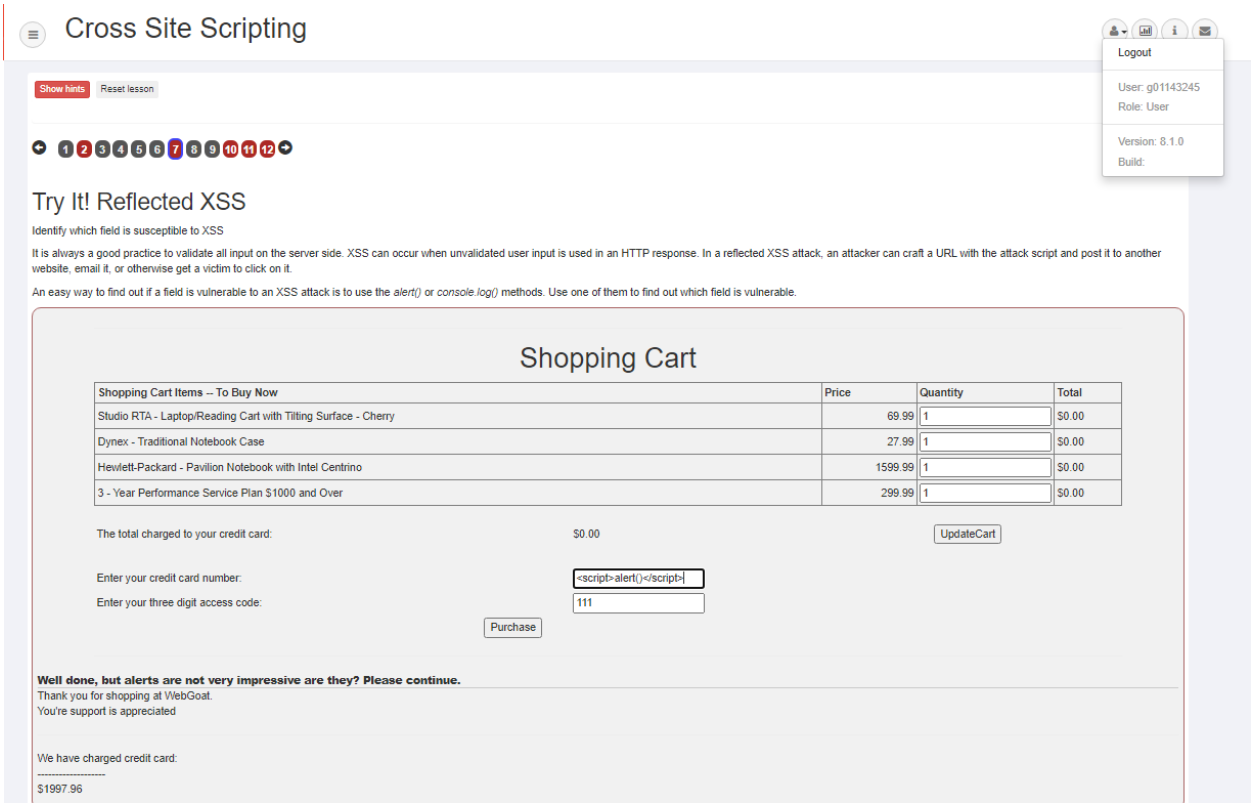
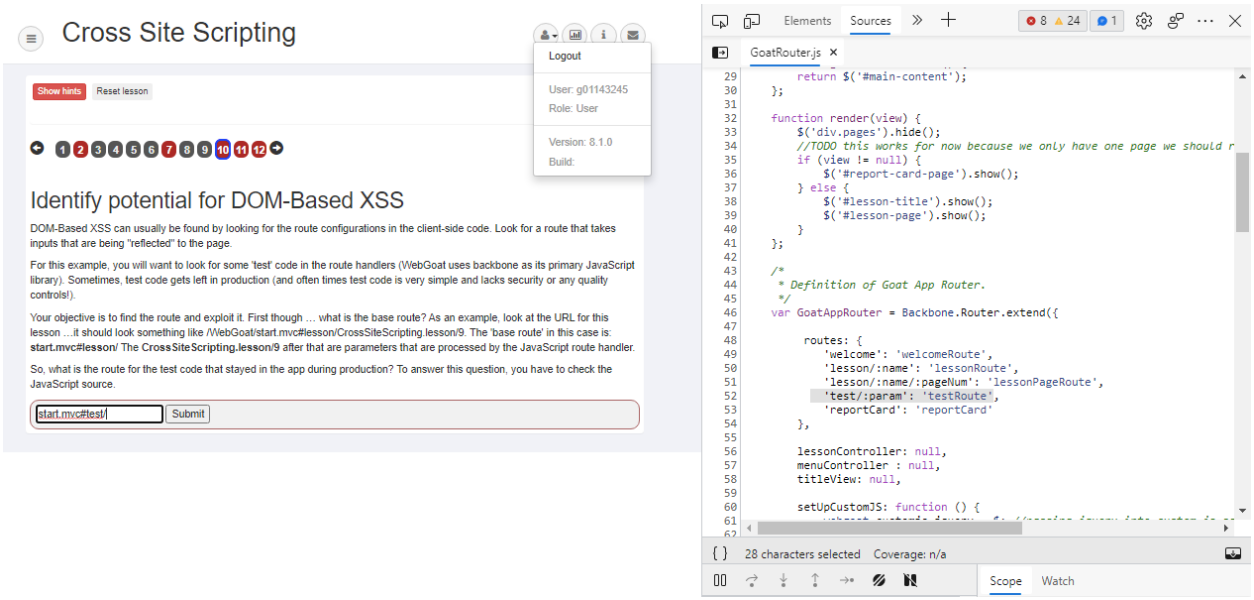
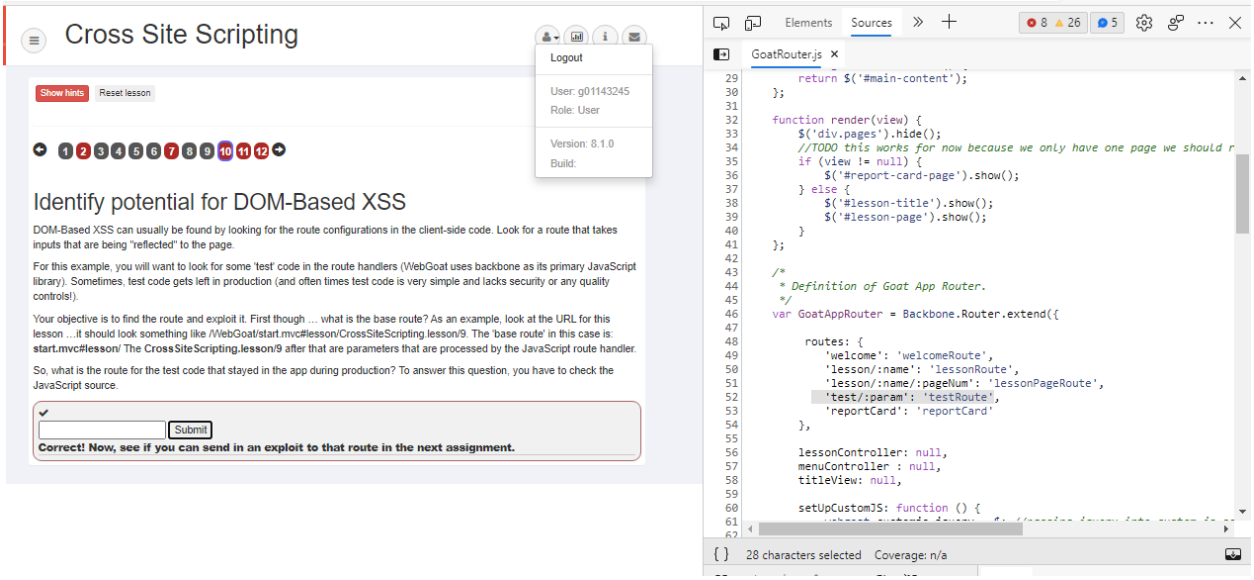
Roozah Khan
Lab #2

a.)

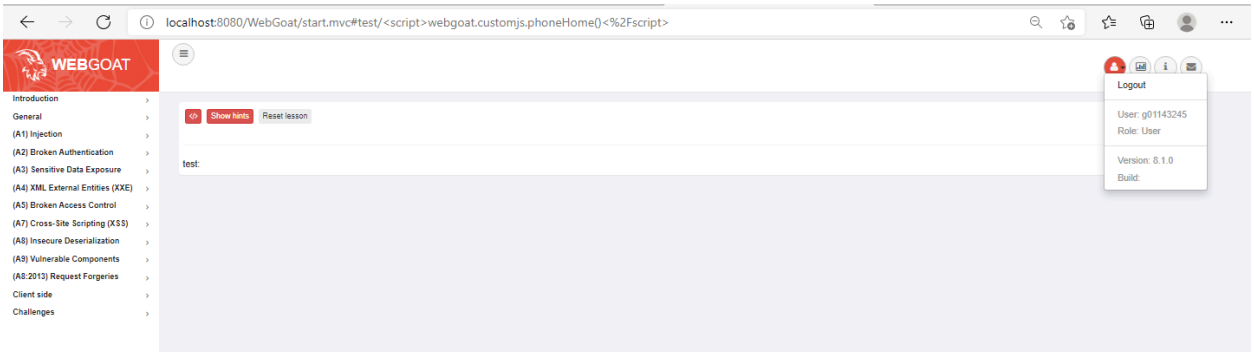


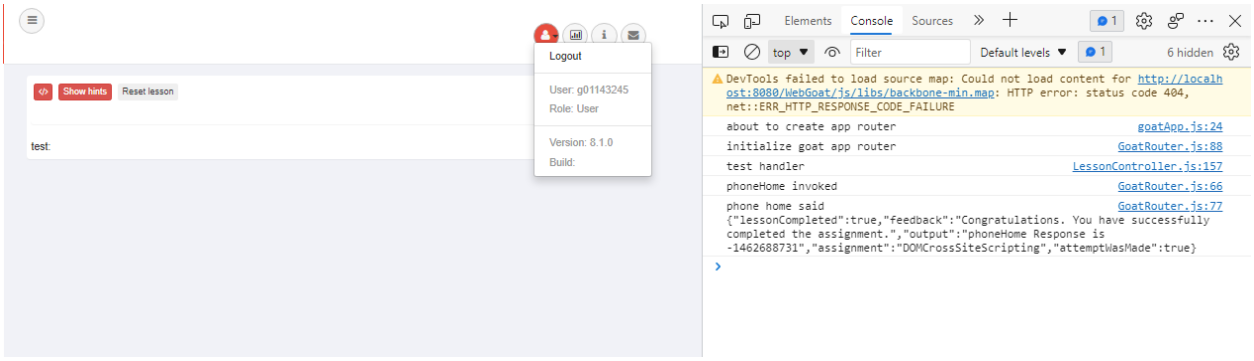
b.)

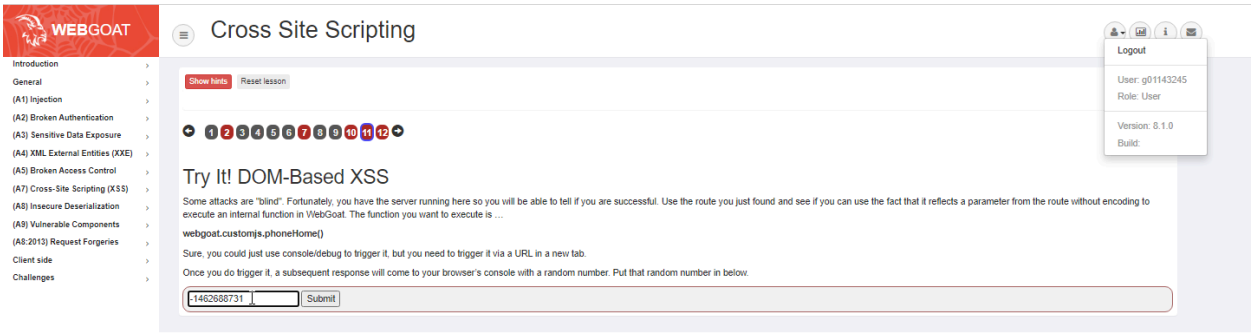

Cross Site Scripting
Logout
User: g01143245
Role: User
Version: 8.1.0
Build:
Show hints
Reset lesson
Identify potential for DOM-Based XSS
DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that are being "reflected" to 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.
start.mvc#test/
Submit
Elements
Sources
GoatRouter.js
return $('#main-content');
};
function render(view) {
$('div.pages').hide();
//TODO this works for now because we only have one page we should r
if (view != null) {
$('#report-card-page').show();
} else {
$('#lesson-title').show();
$('#lesson-page').show();
}
};
/*
* Definition of Goat App Router.
*/
var GoatAppRouter = Backbone.Router.extend({
routes: {
'welcome': 'welcomeRoute',
'lesson/:name': 'lessonRoute',
'lesson/:name/:pageNum': 'lessonPageRoute',
'test/:param': 'testRoute',
'reportCard': 'reportCard'
},
lessonController: null,
menuController : null,
titleView: null,
setUpCustomJS: function () {
28 characters selected
Coverage: n/a
Scope
Watch

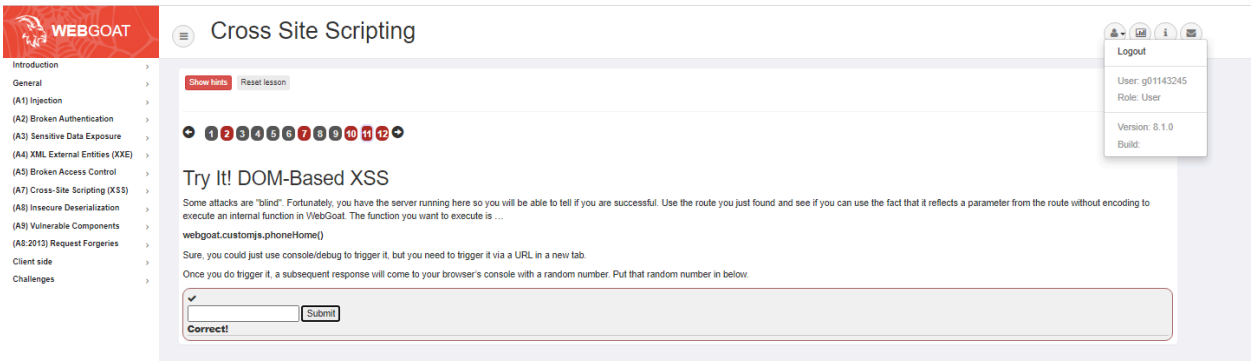

Cross Site Scripting
Logout
User: g01143245
Role: User
Version: 8.1.0
Build:
Show hints
Reset lesson
Identify potential for DOM-Based XSS
DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that are being "reflected" to 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.
Submit
Correct! Now, see if you can send in an exploit to that route in the next assignment.
Elements
Sources
GoatRouter.js
return $('#main-content');
};
function render(view) {
$('div.pages').hide();
//TODO this works for now because we only have one page we should r
if (view != null) {
$('#report-card-page').show();
} else {
$('#lesson-title').show();
$('#lesson-page').show();
}
};
/*
* Definition of Goat App Router.
*/
var GoatAppRouter = Backbone.Router.extend({
routes: {
'welcome': 'welcomeRoute',
'lesson/:name': 'lessonRoute',
'lesson/:name/:pageNum': 'lessonPageRoute',
'test/:param': 'testRoute',
'reportCard': 'reportCard'
},
lessonController: null,
menuController : null,
titleView: null,
setUpCustomJS: function () {
28 characters selected
Coverage: n/a

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: g01143245
Role: User
Version: 8.1.0
Build:


Show hints
Reset lesson
test:
Logout
User: g01143245
Role: User
Version: 8.1.0
Build:
Elements
Console
Sources
top
Filter
Default levels
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
goatApp.js:24
initialize goat app router
GoatRouter.js:88
test handler
LessonController.js:157
phoneHome invoked
GoatRouter.js:66
phone home said
GoatRouter.js:77
{"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","output":"phoneHome Response is -1462688731","assignment":"DOMCrossSiteScripting","attemptWasMade":true}


WEBGOAT
Cross Site Scripting
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
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.
-1462688731
Submit
Logout
User: g01143245
Role: User
Version: 8.1.0
Build:


WEBGOAT
Cross Site Scripting
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
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! 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!