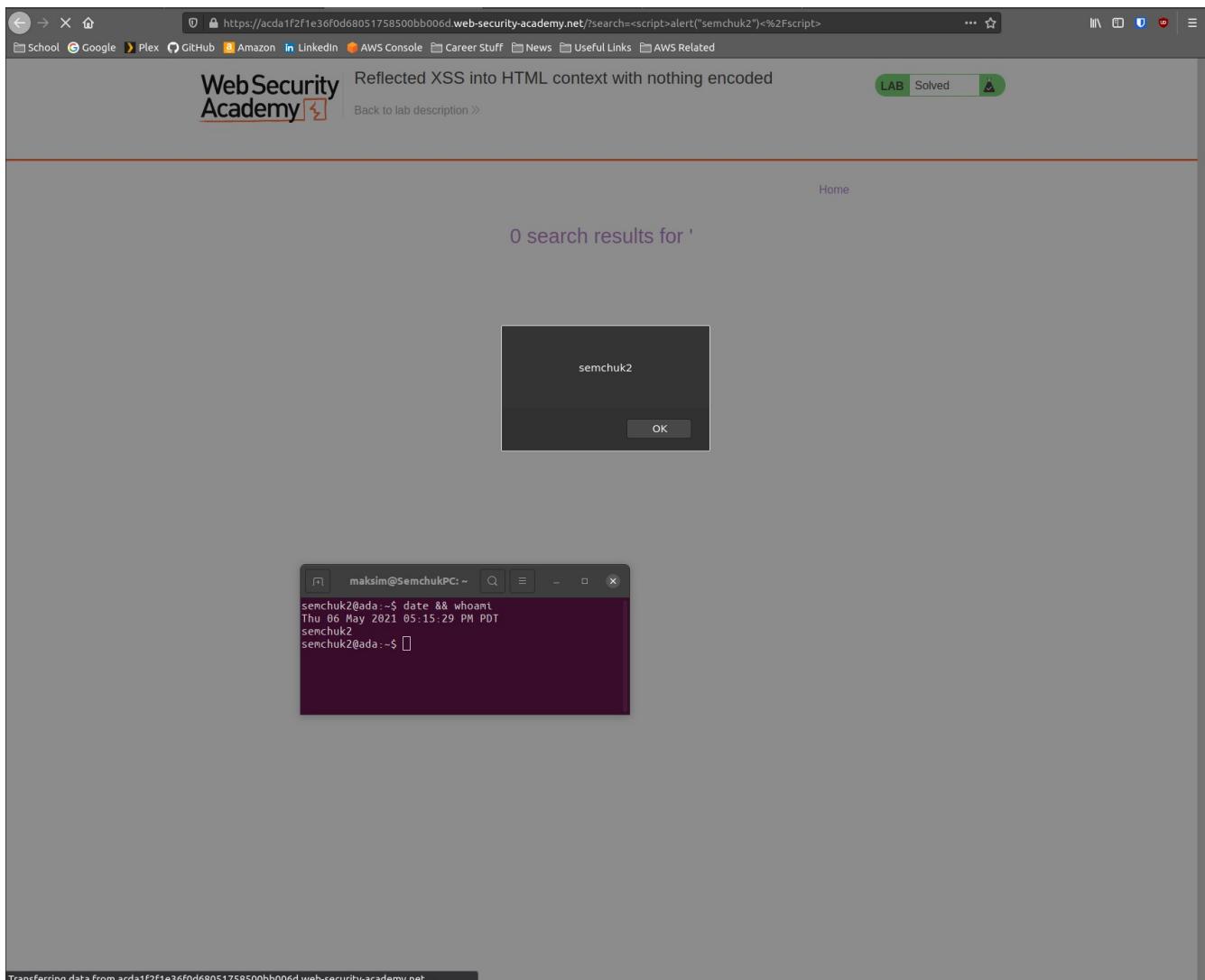


Table of Contents

Lab 3.1 html-context-nothing-encoded:.....	3
Lab 3.1 html-context-with-most-tags-and-attributes-blocked.....	4
3.6 some-svg-markup-allowed.....	13
Lab 3.1.7 attribute-angle-brackets-html-encoded.....	15
3.1.8 javascript-string-single-quote-backslash-escaped.....	16
3.1.9 javascript-string-angle-brackets-html-encoded.....	17
3.1.10 javascript-string-angle-brackets-double-quotes-encoded-single-quotes-escaped.....	18
3.1.11 javascript-template-literal-angle-brackets-single-double-quotes-backslash-backticks-escaped...19	19
3.1.12 document-write-sink.....	20
3.1.13 document-write-sink-inside-select-element.....	21

Lab 3.1 html-context-nothing-encoded:



This vulnerability comes from the fact that the user is able to pass in simple JS commands and have the remote server actually run them. This is shown by the fact that the web server displays an alert with my username.

Lab 3.1 html-context-with-most-tags-and-attributes-blocked

The ones ones allowed are on resize and onstorage.

School Google Plex GitHub Amazon LinkedIn AWS Console Career Stuff News Useful Links AWS Related

WebSecurity Academy 

Reflected XSS into HTML context with most tags and attributes blocked

Go to exploit server Back to lab description >

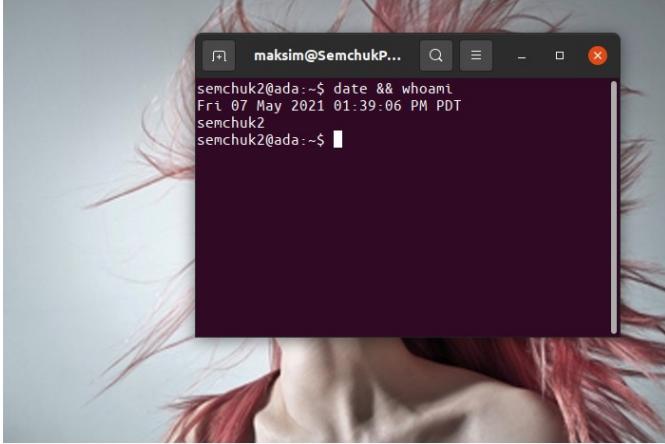
Home

WE LIKE TO 

BLOG

<body onresize=alert(document.cookie)></body>

Search



Smart Recognition

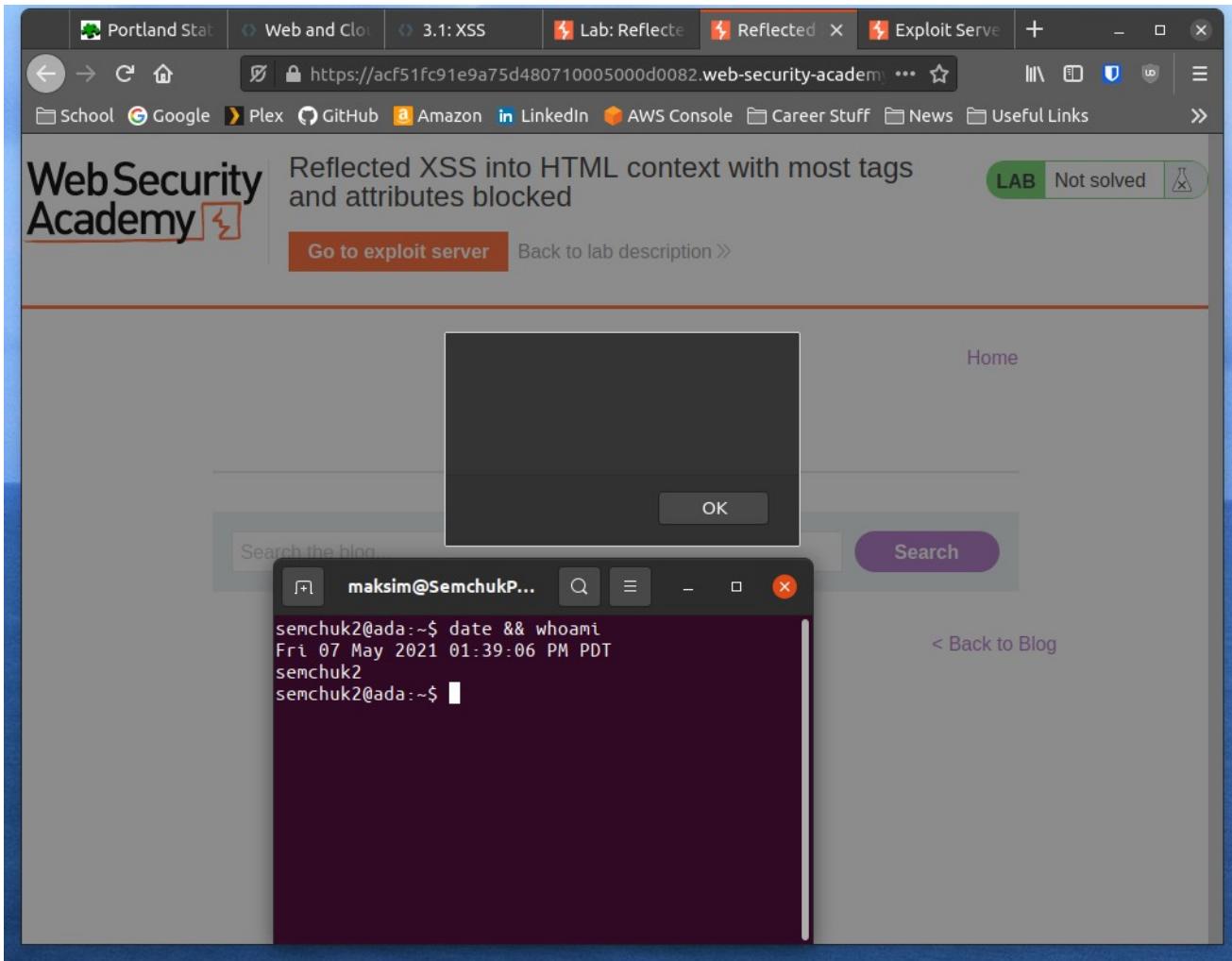
The techie folks have done it again. Yet another way to secure your cell, and other electronic gadgets. This Smart Recognition is like no other you've met with yet. It all comes down to the scent of your hair, your...

[View post](#)



This is before resize, I searched for: <body onresize=alert(document.cookie)></body>

Therefore it should pop up a alert after I resize the window.



And it does resize after I searched for the exploit and resized the window.



Hello semchuk2

A screenshot of a terminal window titled 'maksim@SemchukP...'. The terminal shows the command 'date && whoami' being run, with the output 'Fri 07 May 2021 01:39:06 PM PDT' and 'semchuk2'. The terminal prompt 'semchuk2@ada:~\$' is visible at the bottom.

This is the exploit being sent to the server.

```
50.53.222.79 - - [07/May/2021:10:02:33 +0000] "GET /exploit HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
50.53.222.79 - - [07/May/2021:10:05:18 +0000] "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0"
```

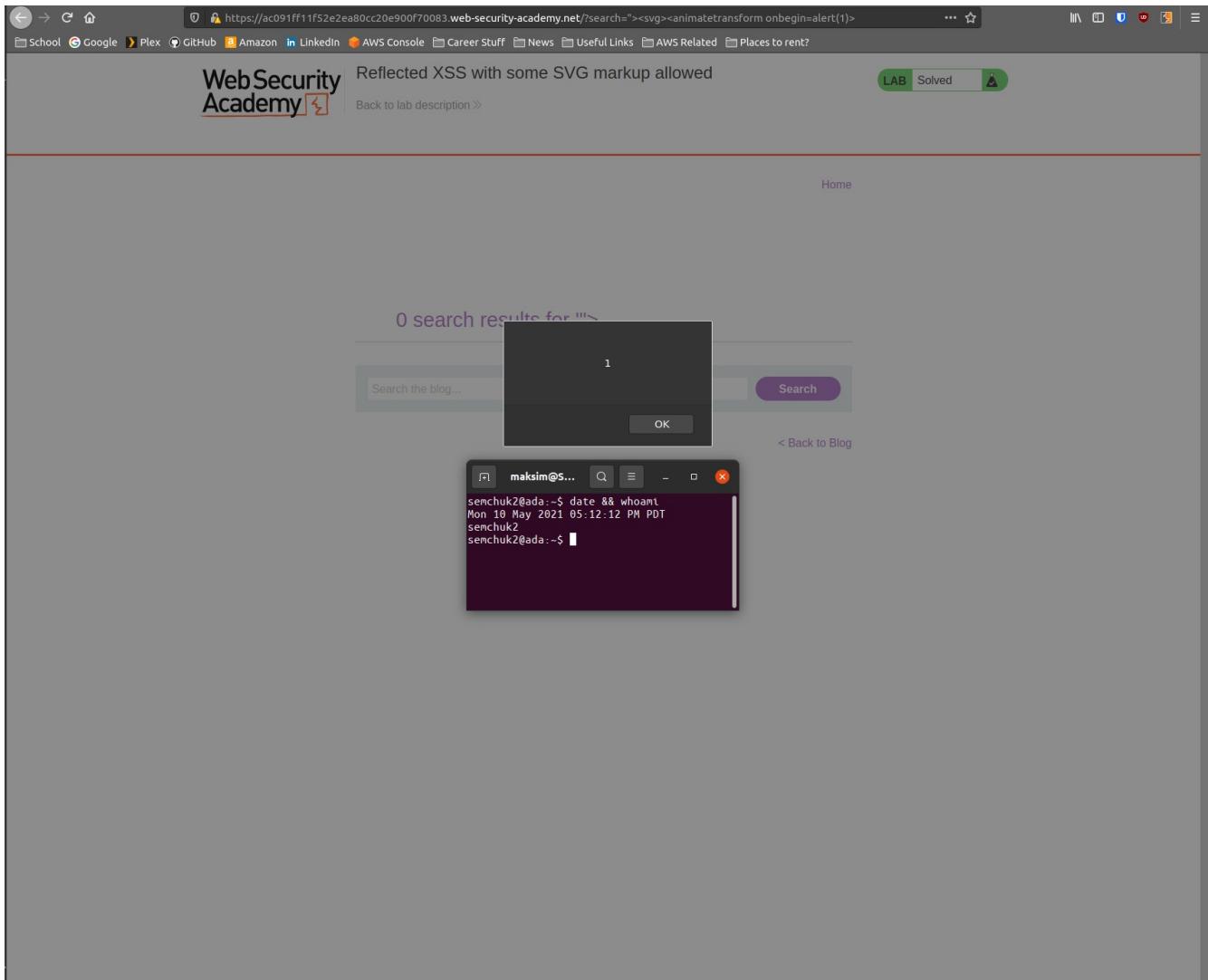
Here my my IP Address.

Here it is completed.

The screenshot shows a browser window with the URL <https://act21fb81eee6f64809a97ca00a10065.web-security-academy.net>. The title bar indicates the page is from Bitwar. The main content area shows a terminal window with the command `maksim@SemchukP... semchuk2@ada:~$ date && whoami` and the output `Fri 07 May 2021 03:26:43 PM PDT semchuk2`. Above the terminal is the text "Reflected XSS into HTML context with most tags and attributes blocked". Below the terminal is a link "Back to lab description >". A green button at the top right says "LAB Solved". A banner at the bottom says "Congratulations, you solved the lab!" with options to "Share your skills!" and "Continue learning >". To the left of the terminal is a blog post thumbnail titled "WE LIKE TO BLOG" featuring a bouquet of pink roses. The post title is "Say It With Flowers - Or Maybe Not" and the content is a short message about St. Valentine's Day. A "View post" button is at the bottom of the post.

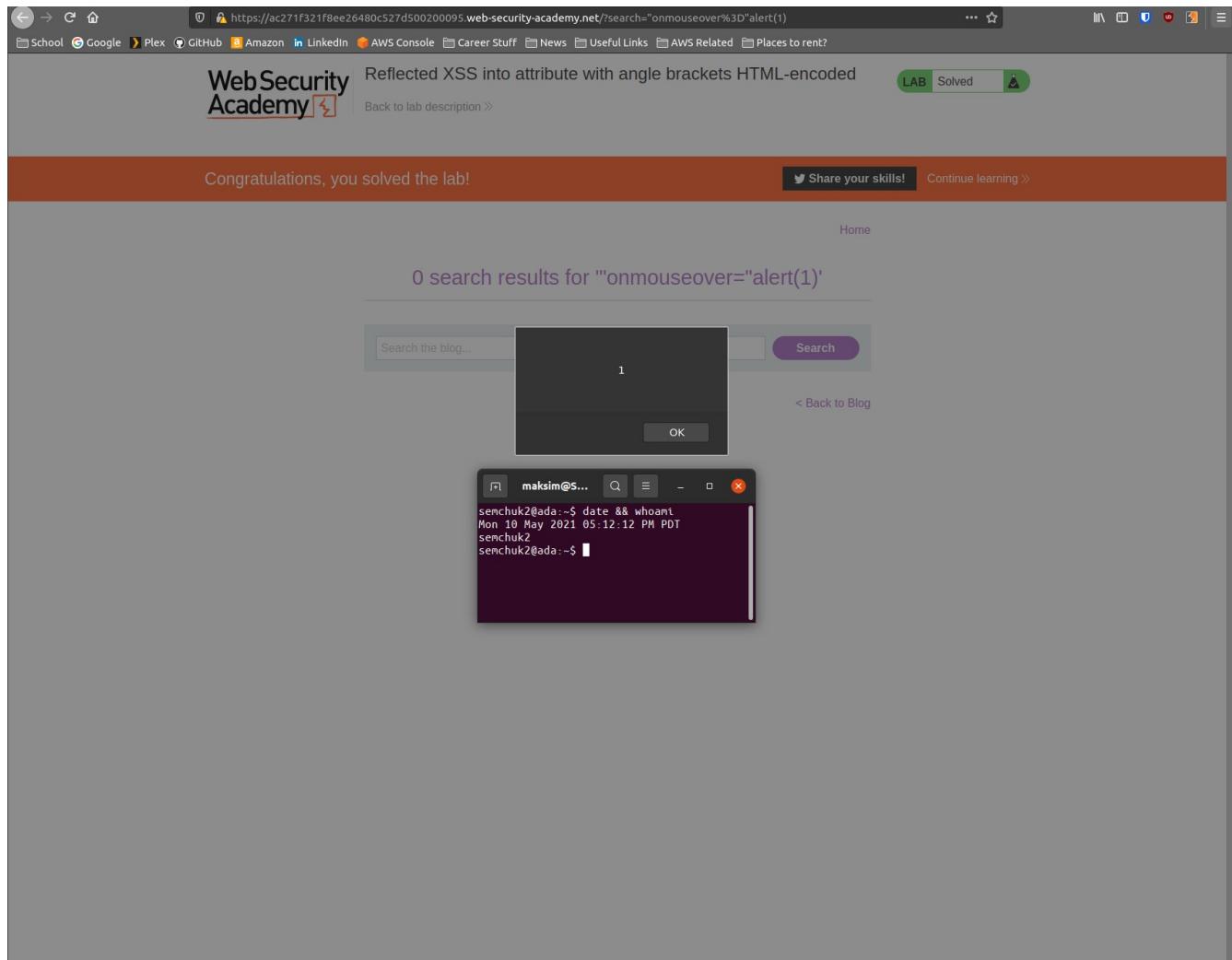
The reason why this worked, is because it does not protect body tags, and the places are execute the commands. The way to prevent this is to parse out any rags and take everything as a string, and not as a command.

3.6 some-svg-markup-allowed



Here we are able to see that certain tags are allowed. By doing a brute force attack which tests which of the tags are allowed (meaning get a 200 response back) we are able to target those weak tags to force through an alert.

Lab 3.1.7 attribute-angle-brackets-html-encoded



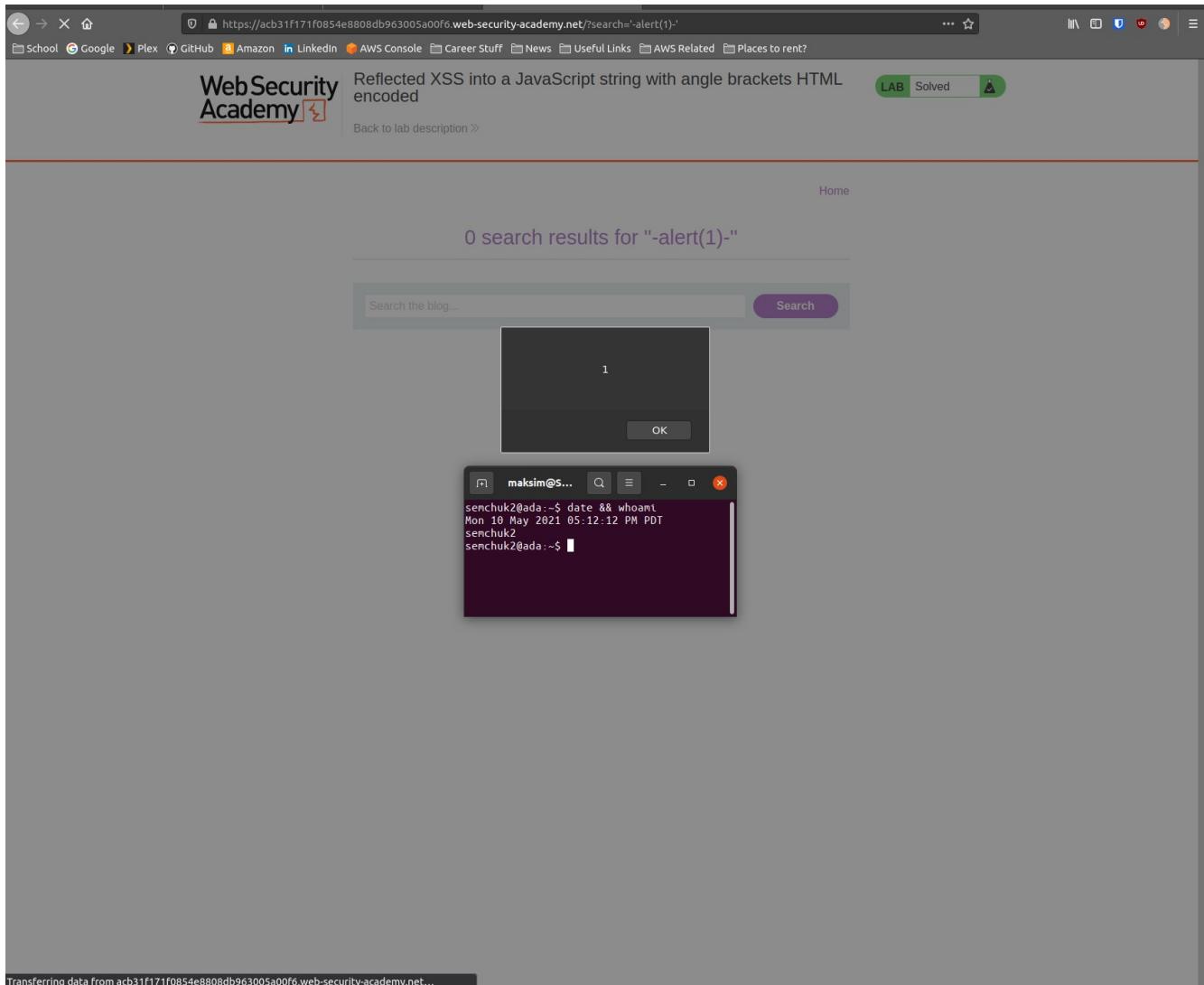
What happens here, is that since the search request is reflected, then it allows us to put the event in double quotes and have a action occur since when it get send to the reflected server it is formatted differently.

3.1.8 javascript-string-single-quote-backslash-escaped

The screenshot shows a browser window for the Web Security Academy. The URL is <https://ac091f9f1f369c538060fb2006a007c.web-security-academy.net/>. The page title is "Reflected XSS into a JavaScript string with single quote and backslash escaped". A green button indicates it's a "LAB" and "Solved" challenge. Below the title, there's a link "Back to lab description >". The main content area shows a search bar with the placeholder "Search the blog..." and a purple "Search" button. Below the search bar is a modal dialog box with the number "1" and an "OK" button. At the bottom of the page, there's a terminal window titled "maksim@S..." showing the command "date && whoami" and its output: "Mon 10 May 2021 05:12:12 PM PDT" and "semchuk2". The status bar at the bottom of the browser window says "Transferring data from ac091f9f1f369c538060fb2006a007c.web-security-academy.net...".

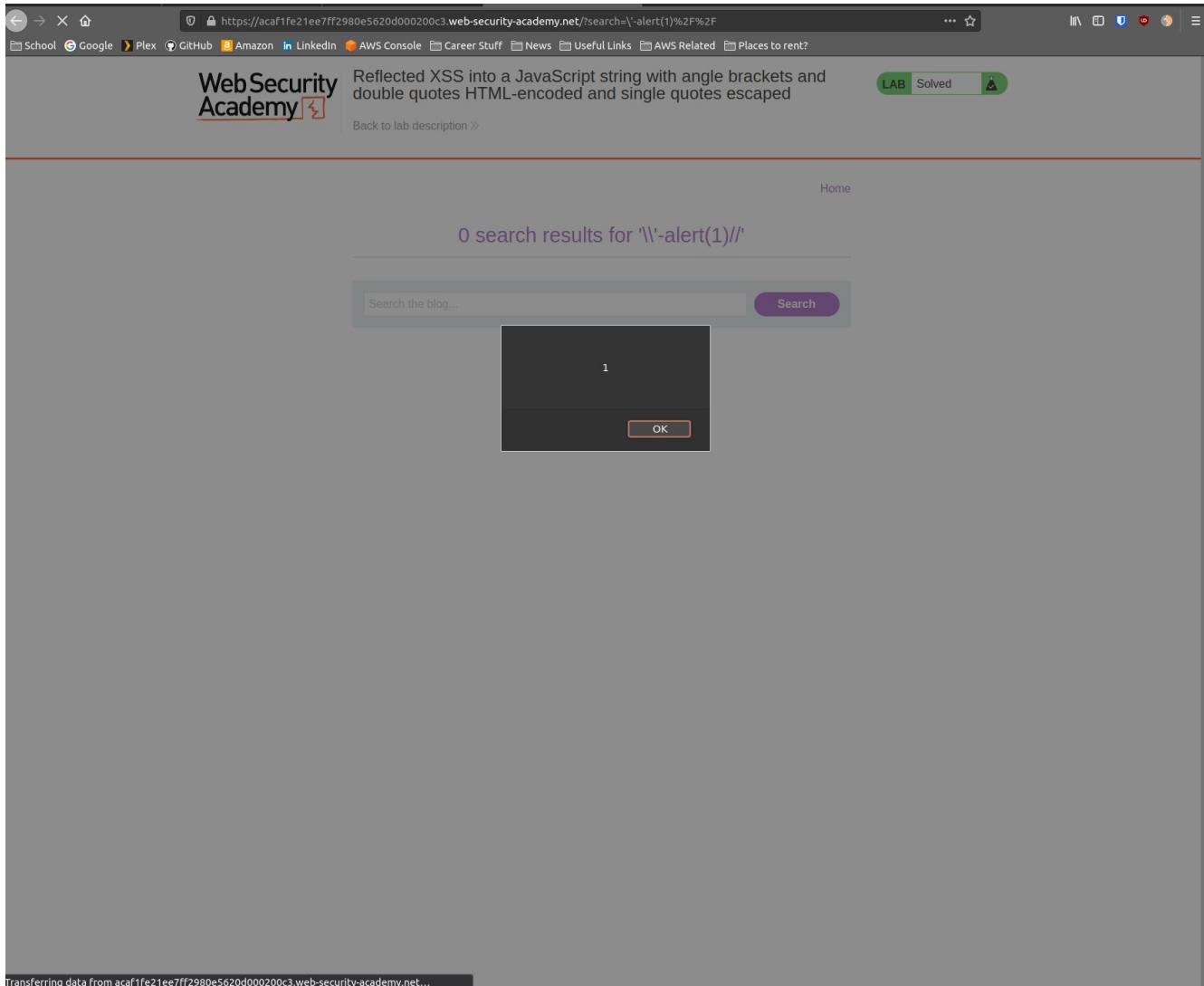
This is a vulnerability since it allows a user to pass in a script escape followed by a script tags to then inject a command into the search bar. The function is then called by javascript.

3.1.9 javascript-string-angle-brackets-html-encoded



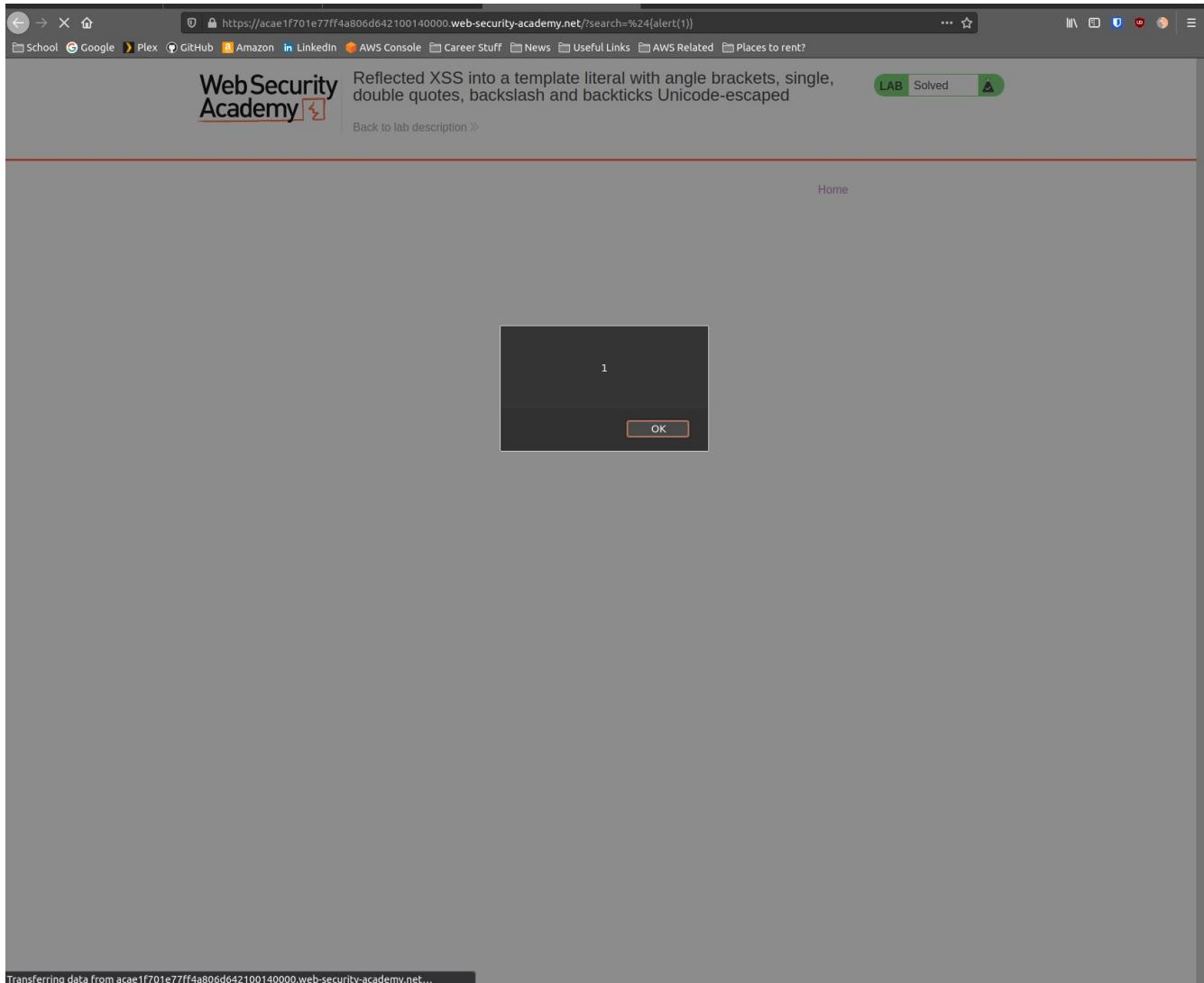
This allows a person to inject javascript into the searched bar, and run code. This allows the user to escape out of SQL, and then into java where they can call commands to do things, like an alert. This is done by having escape characters not be checked and assumed safe by the developer.

3.1.10 javascript-string-angle-brackets-double-quotes-encoded-single-quotes-escaped



This vulnerability is due to the fact that the HTML escape characters can be escaped and then are once again injected into the Javascript and run as a command.

3.1.11 javascript-template-literal-angle-brackets-single-double-quotes-backslash-backticks-escaped



These weaknesses come from the fact that the html characters are encoded, and sent to the refraction server. The server decodes them and then runs the code that was sent, resulting in a attack being successful.

3.1.12 document-write-sink

The screenshot shows a browser window for the WebSecurity Academy lab titled "DOM XSS in document.write sink using source location.search". The URL is https://ac821fd11fd1544c8082c0b000dd0073.web-security-academy.net/?search=""><svg+onload%3Dalert(1)>. The page displays search results for the query, showing a single result. Below the search bar is a modal dialog box with the number "1" and an "OK" button. At the bottom of the page is a terminal window showing a Linux shell session with the command "date && whoami" and output "Mon 18 May 2021 05:12:12 PM PDT" and "semchuk2@ada:~\$". A status bar at the bottom left indicates "Transferring data from ac821fd11fd1544c8082c0b000dd0073.web-security-academy.net...".

The vulnerability here, gets called from the document where the document in the webpage and gets executed on the server by doing escapes.

3.1.13 document-write-sink-inside-select-element

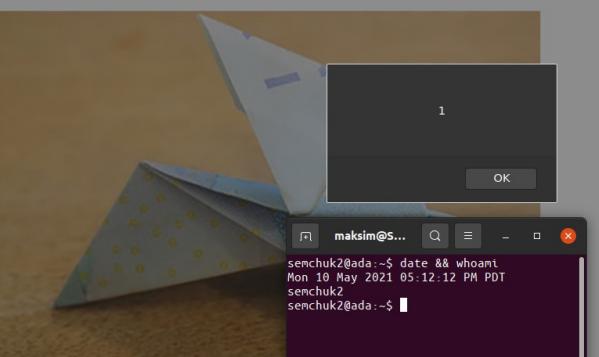
DOM XSS in document.write sink using source location.search inside a select element

Back to lab description »

Folding Gadgets

★ ★ ★ ★ ★

\$10.34



Description:

Folding smartphones that open up to reveal a handy little tablet are about to hit the market. Is folding the future of technology? As gadget trends go from large small, back to large again, small again, huge, I guess folding has to be the answer, the best of both worlds. They are still bulky though, once we start folding everything things have a tendency to get thicker. Purses and briefcases will need to be adjusted to accommodate these new convenient, but bulky items.

With this new concept, we can really make outside spaces and coffee houses our home offices. Pitch up in the park on a sunny day, and dig deep into your oversized carpet bag, with magician-like prowess you will be able to unfold your desk, PC, speakers, keyboards and mice until you have everything you need to start your days work. Even your travel mug and flask will conveniently unfold leaving you hydrated in that hot summers sun.

I was a bit of a trendsetter in this department, I have always folded my paper money, my grandmother used to do it and I guess the influence stuck with me. Little did granny know that 40 years on we would all be folding our money, and everything else we can attach minuscule hinges to. We have always folded our laundry as well, that goes back centuries. Like all good inventions, it takes time to bring these things to market.

To be honest I've been crying out for a tablet that makes phone calls ever since my eyesight deteriorated. Sadly it will probably only be affordable to those that can afford laser surgery, and they're just being greedy as they have no problems looking at a tiny cell phone screen. I hate touch screens and have had a folding keyboard for yonks, give me a giant Blackberry any day!

">

London

Paris

Milan

The javascript is exposed and is able to be injected into the search bar. The code is then executed

3.1.14 innerhtml-sink

The screenshot shows a browser window with the URL [https://ac891f591f8d5499806dcfc300fb001c.web-security-academy.net/?search=+<img+src%3D1+onerror%3Dalert\(1\)>](https://ac891f591f8d5499806dcfc300fb001c.web-security-academy.net/?search=+<img+src%3D1+onerror%3Dalert(1)>). The page title is "DOM XSS in innerHTML sink using source location.search". A modal dialog box is displayed, containing the number "1" and an "OK" button. Below the modal, a search results page is shown with the message "0 search results for !". A terminal window is also visible, displaying the command "date && whoami" and the output "Mon 10 May 2021 07:26:29 PM PDT" and "senchuk2@ada:~\$". At the bottom of the browser window, there is a status bar with the text "Transferring data from ac891f591f8d5499806dcfc300fb001c.web-security-academy.net...".

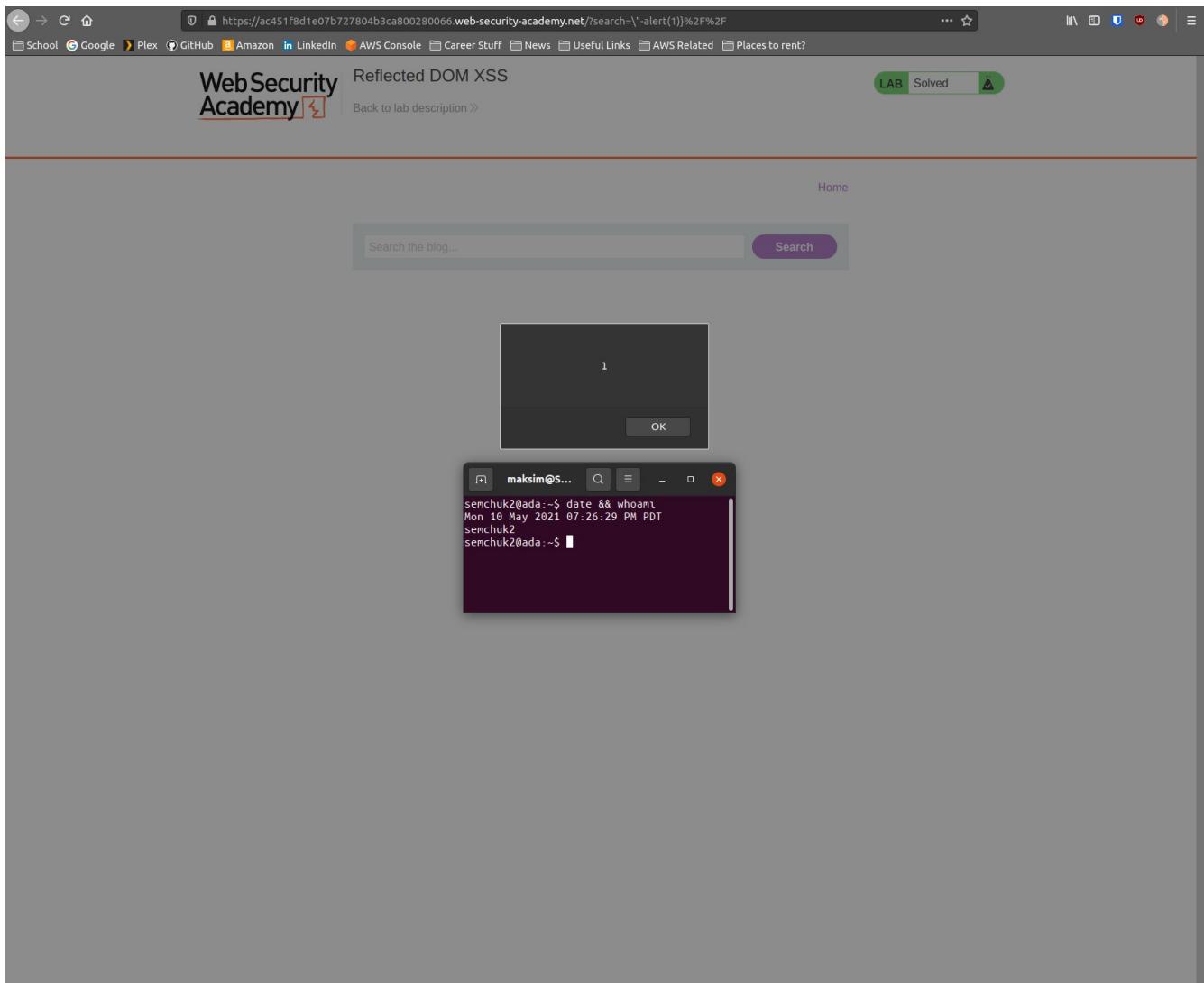
Here, we are able to show that the HTML is taken into a source as a javascript file. The second site is taken into the source file as a image, but contain the function to alert.

3.1.15 jquery-href-attribute-sink

The screenshot shows a browser window for the 'Web Security Academy' lab titled 'DOM XSS in jQuery anchor href attribute sink using location.search source'. The URL is [https://acdb1f9a1e9cfb280b18468006900e6.web-security-academy.net/feedback?returnPath=javascript:alert\(1\)](https://acdb1f9a1e9cfb280b18468006900e6.web-security-academy.net/feedback?returnPath=javascript:alert(1)). The page displays a success message: 'Congratulations, you solved the lab!' with options to 'Share your skills!' and 'Continue learning >'. A green button at the bottom says 'Submit feedback'. A modal dialog box is open, showing the number '1' and an 'OK' button. Below the modal, there's a terminal window showing a Linux session with the command 'date && whoami' and output 'Mon 10 May 2021 07:26:29 PM PDT' and 'senchuk2'. At the bottom left, there's a small note: 'javascript:alert(1)'.

This allows us to insert a javascript as a argument into the return link, so when a user clicks back it shows an alert function with a 1.

3.1.16 dom-xss-reflected



This is a cross site scripting attack that gets reflected by a server. Then the server runs the script.

3.1.18 html-context-nothing-encoded

The screenshot shows a web browser window with the URL <https://acad1f511f6e546680ddd36800740071.web-security-academy.net/post?postId=6>. The page title is "Stored XSS into HTML context with nothing encoded". A green "LAB" button and a "Solved" badge are visible. Below the title, there's a link "Back to lab description >". The main content features a cartoon illustration of a ninja holding a spear pointing at a computer monitor. On the monitor, a modal dialog box displays the number "1" and an "OK" button. In front of the monitor, a terminal window shows the command "maksim@S... date && whoami" followed by the output "Mon 10 May 2021 07:26:29 PM PDT" and "seanchuk2@ada:~\$". The terminal has a dark purple background. The page footer contains a note about transferring data and a copyright notice: "Transferring data from acad1f511f6e546680ddd36800740071.web-security-academy.net... Using history is an A-Z of boring research tasks from Acupuncture for".

The cross site script attack allows us to put html to the server, and when the comment part gets parsed and since the comment is not sanitized then it gets executed.

3.1.19 href-attribute-double-quotes-html-encoded

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home

maksim@S... date && whoami
Mon 10 May 2021 07:26:29 PM PDT
maksim
maksim@ada:~\$

Hobbies

Jock Sonyou | 14 April 2021

Hobbies are a massive benefit to people in this day and age, mainly due to the distractions they bring. People can often switch off from work, stress and family for the duration of their hobbies. Maybe they're playing sports, knitting or just having their normal two hour hideout in the toilet to avoid people, whatever your hobby is, embrace how it distracts you from other stresses.

However, some existing hobbies may be getting in the way of your life and are not helping you relax at all. If you're an aggressive sportsman or woman then chances are you're getting more riled up playing competitively than actually relaxing. This might be a perfect time in your life to find a new hobby, for example, photography may tickle your fancy. Does getting out and about and snapping the beauty of the world sound appealing to you? Maybe it could help you reconnect with nature, maybe you're a budding Instagram artist wanting to show off your pics, so why not pick up a camera and see if photography subdues some stress and even make some money? But be sure not to snap anything illegal or people without permission, the pending lawsuits will put a stop to your new hobby.

You could try your hand at writing! The beauty of writing forms like poetry and prose is they are very subjective. Poetry is the written equivalent of modern art. You can look at a dot that's worth millions for some reason, and you can read poetry that is not too dissimilar in unexplained greatness. If people dislike your poetry, you can dismiss them as just not understanding it. And, who knows, if people like your writing, again it could make you some extra money on the side. It

So here what happens is, if you do some filtering of the data, then you are able to intercept a packet and then edit it through burp. In this case, normally you cannot do a non formated website name but if you intercept the packet you can repeat the request with a custom website payload, sending javascript functions instead. This is dangerous since you bypass the check and gain access to the remote server.

3.1.20 onclick-event-angle-brackets-double-quotes-html-encoded-single-quotes-backslash-escaped

The screenshot shows a browser window for <https://ac941f921ea045cb803509fb00f300cd.web-security-academy.net/post?postId=2>. The title bar indicates the URL and the page is marked as 'Solved'. The main content area shows a success message: 'Congratulations, you solved the lab!' with options to 'Share your skills!' and 'Continue learning >'. Below this, there's a decorative graphic of several speech bubbles with question marks. A terminal window is overlaid on the graphic, showing a command-line session:

```
maksim@...:~$ date && whoami
Mon 10 May 2021 07:26:29 PM PDT
seanchuk2
seanchuk2@ada:~$
```

At the bottom left of the terminal window, there is a small text: 'foo?`-alert(1)`.'

Here, a comment gets forwarded through a one click redirect, and this is is really dangerous since you are able to then inject a javascript function into it and have it run on redirect.

3.1.23 dom-xss-stored

The screenshot shows a browser window for 'Web Security Academy' with the title 'Stored DOM XSS'. The main content area displays a woman performing a yoga pose against a blue sky background. A dark modal dialog box is overlaid on the image, containing the number '1' and an 'OK' button. Below the image, a terminal window shows a Linux session with the command 'date && whoami' run by user 'semchuk2' on 'Mon 10 May 2021 08:20:35 PM PDT'. At the bottom of the page, there is a post titled 'The Revers' by 'Kel Surprzee' with 14 likes.

I have yet to create a bucket list, mainly because I'm not very adventurous and don't want to do anything that will scare the pants off me. With my weekends wasting away with a huge dose of apathy and only a spoonful of activity, I felt it was time to try something new. I like to call it the Reverse Bucket List, doing things you think you should do, but don't really want to.

The first class I signed up to was freefall yoga. I wasn't sure of the freefall part but I'd seen great videos of people doing yoga in hammock type things and thought it would probably be a bit like that. At this point it's worth mentioning I hate yoga, I thought I'd start with the things I hate the most and work up.

Weeks of grueling body bending ensued and not a hammock in sight. The only falling taking place was when I tried to get out of bed in the mornings. Why would something that promotes suppleness and fitness leave you incapable of descending the stairs and leave you with the added inability to dress yourself as you can no longer lift your arms above your head?

The day of certification was drawing close and there seemed to be great excitement among the group. We were rather oddly, I thought, asked to meet the following weekend at a local airfield. It was to be a weekend affair with swanky overnight accommodation and Canap's. On arrival, we were greeted with a glass of champagne and asked to change into our yoga gear. What followed next will haunt me until the day I die.

If I didn't recognize my group and our yogini, I would have assumed I'd turned up at the wrong venue and made my excuses. It is not unheard of for me to turn up at the wrong venue. At the end of the weekend, I was still standing in the middle of the airfield, not having done a single thing.

This is a simple cross site attack where the website attempted to prevent this by doing a replace but by allowing more then one set of brackets to come through, the attack succeeds since it does not do this to all brackets, just the first one.

3.1.25 stealing-cookies

The screenshot shows the Network tab of a browser's developer tools. A specific request for 'accountDetails' is selected, revealing a JSON response. The response contains the user's API key ('apiKey'), session ID ('sessions[0]'), and session cookie ('P9ok').

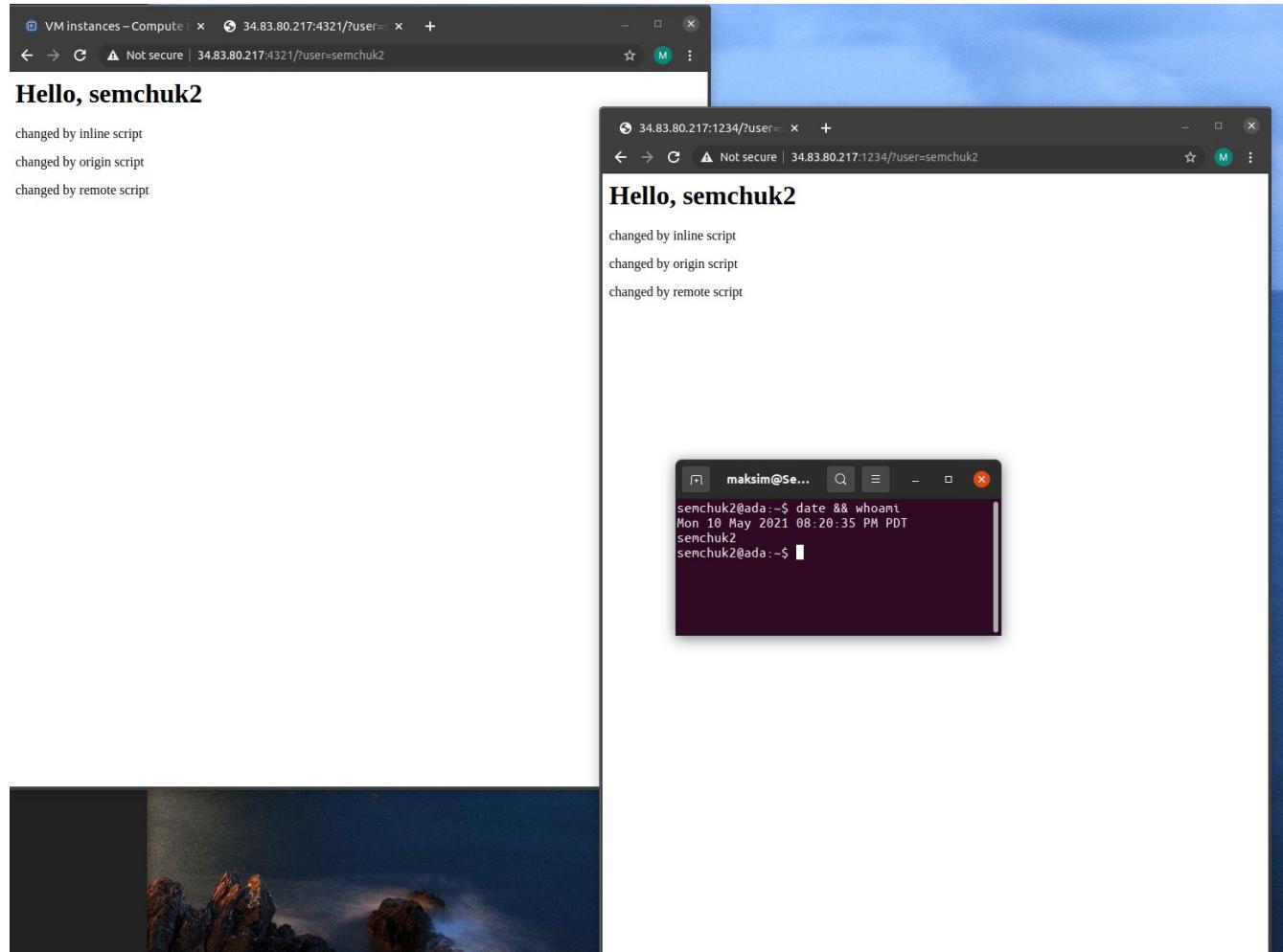
status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace	Security
200	GET	ac781f141f75f3d0800...	my-account	BrowseTabChild.jsm:92 (...	html	1.42 KB	3.80 KB				JSON			
200	GET	ac781f141f75f3d0800...	labHeader.js	script	js	529 B	739 B				username: "wiener"			
200	GET	ac781f141f75f3d0800...	submitSolution.js	script	js	700 B	1 KB				email: ""			
200	GET	ac781f141f75f3d0800...	academyLabHeader	labHeader.js:2 (websocket)	plain	249 B	0 B				apiKey: "bbr4K4MFANVCuK1pCp8Lep38BV8Ms3wi"			
200	GET	ac781f141f75f3d0800...	accountDetails	my-account:56 (fetch)	json	341 B	149 B				sessions: ["koZzSApK7iZqu67FQPB35ZQ4hpsP9ok"]			
200	GET	ac781f141f75f3d0800...	favicon.ico	FaviconLoader.jsm:191 (img)	x-icon	cached	15.04 KB				0: "koZzSApK7iZqu67FQPB35ZQ4hpsP9ok"			

The screenshot shows the 'My Account' page from the Web Security Academy. It displays a success message: 'Congratulations, you solved the lab!' and a 'Share your skills!' button. The user's session is marked as 'Solved'. Below the main content, there is a terminal window showing a successful login attempt.

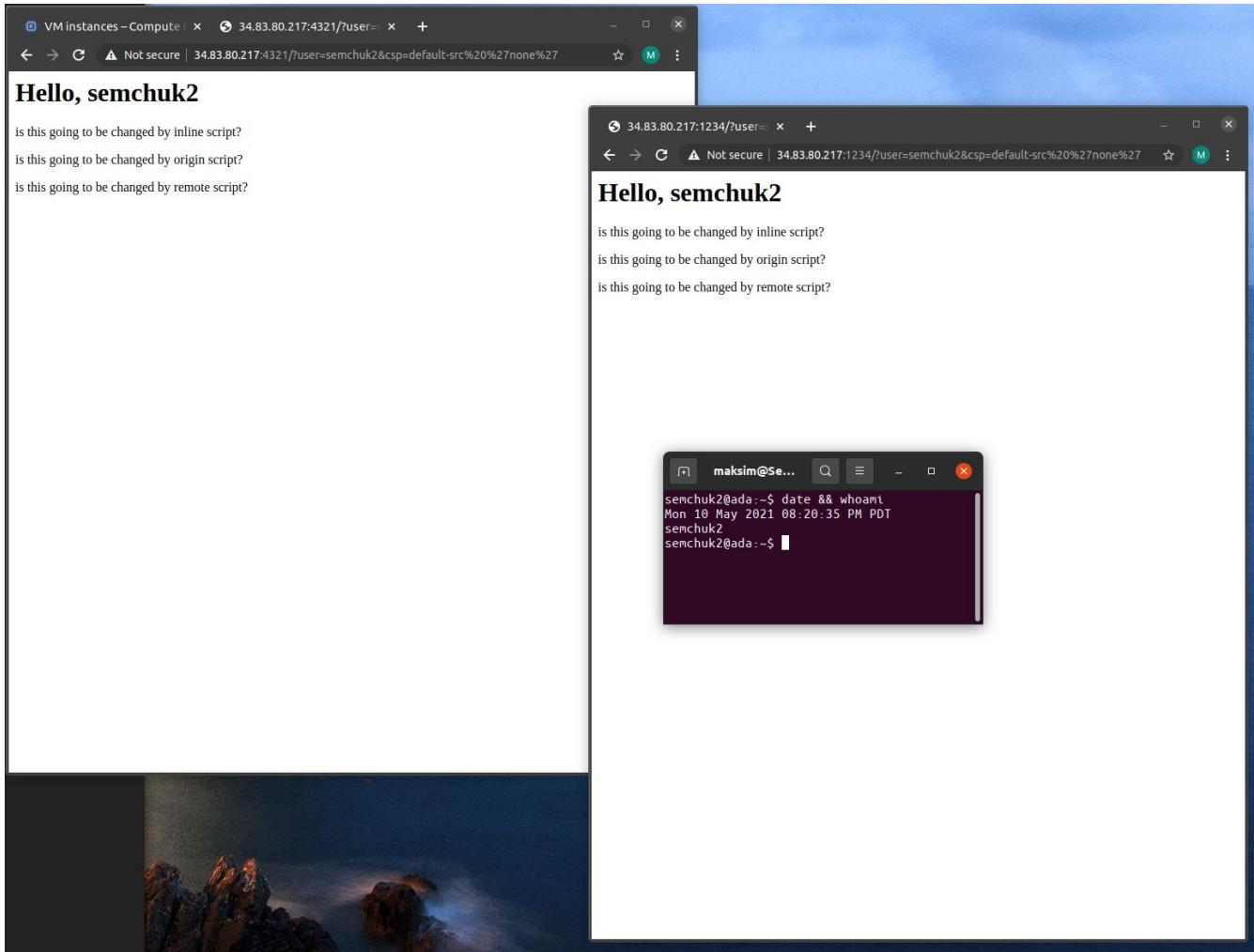
```
maksim@ada:~$ date && whoami
Mon 10 May 2021 08:20:35 PM PDT
semchuk2
semchuk2@ada:~$
```

This is a vulnerability because the server is allowing and allows unknown domains who are redirected to access API keys.

3.2.4

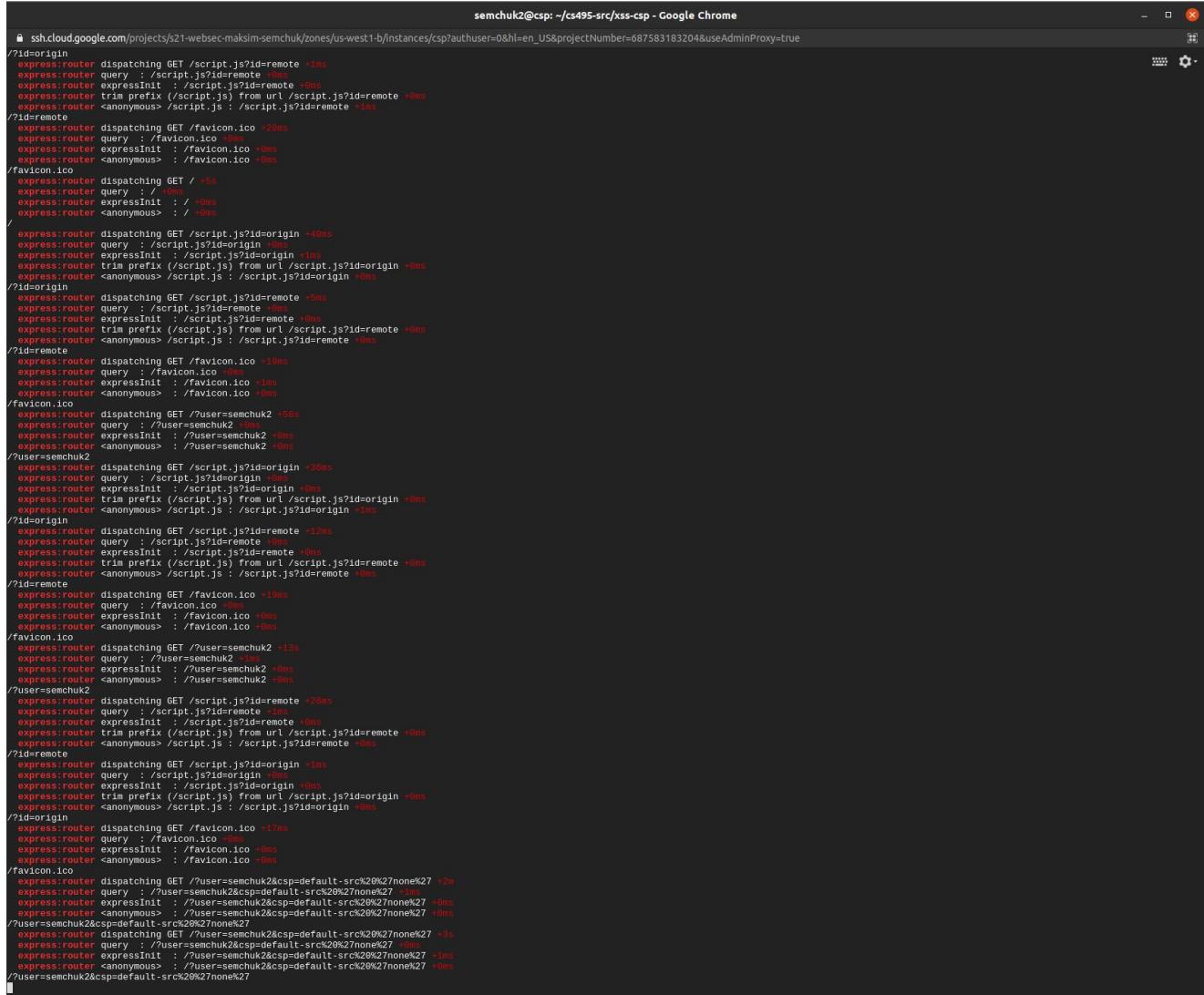


Submission of my username



here is the restriction being changed

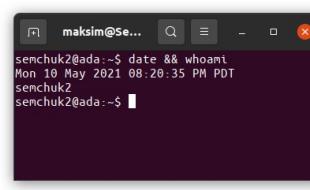
The console output for the last two steps:



A screenshot of a terminal window titled "semchuk2@csp: ~/c495-src/xss-csp - Google Chrome". The window displays a large amount of log output from a Node.js application. The logs are color-coded: green for standard output and red for error output. The log entries show various express:router dispatching events for different routes like '/favicon.ico' and '/script.js?id=origin'. It also shows express:router expressInit and express:router trimPrefix events. The logs indicate that the application is running on port 3001 and handling requests for files like favicon.ico and script.js. There are several instances of the same log entry repeated, suggesting a loop or a bug in the application's handling of certain requests.

```
semchuk2@csp: ~/c495-src/xss-csp - Google Chrome
ssh.cloud.google.com/projects/s21-websec-maksim-semchuk/zones/us-west1-b/instances/csp?authuser=0&hl=en_US&projectNumber=687583183204&useAdminProxy=true
/express:router dispatching GET /script.js?id=remote +1ms
/express:router query : /script.js?id=remote
/express:router expressInit : /script.js?id=remote +0ms
/express:router trim prefix /script.js from url /script.js?id=remote +0ms
/express:router <anonymous> /script.js : /script.js?id=remote +1ms
/express:router dispatching GET /favicon.ico +20ms
/express:router query : /favicon.ico
/express:router expressInit : /favicon.ico +0ms
/express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
/express:router dispatching GET / +5s
/express:router query : +5s
/express:router expressInit : / +5s
/express:router <anonymous> : / +0ms
/express:router dispatching GET /script.js?id=origin +40ms
/express:router query : /script.js?id=origin +0ms
/express:router expressInit : /script.js?id=origin +0ms
/express:router trim prefix /script.js from url /script.js?id=origin +0ms
/express:router <anonymous> /script.js : /script.js?id=origin +0ms
/express:router dispatching GET /script.js?id=remote +5ms
/express:router query : /script.js?id=remote +0ms
/express:router expressInit : /script.js?id=remote +0ms
/express:router trim prefix /script.js from url /script.js?id=remote +0ms
/express:router <anonymous> /script.js : /script.js?id=remote +0ms
/express:router dispatching GET /favicon.ico +10ms
/express:router query : /favicon.ico +0ms
/express:router expressInit : /favicon.ico +0ms
/express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
/express:router dispatching GET /user=semchuk2 +5s
/express:router query : /user=semchuk2 +0ms
/express:router expressInit : /user=semchuk2 +0ms
/express:router <anonymous> : /user=semchuk2 +0ms
/user=semchuk2
/express:router dispatching GET /script.js?id=origin +30ms
/express:router query : /script.js?id=origin +0ms
/express:router expressInit : /script.js?id=origin +0ms
/express:router trim prefix /script.js from url /script.js?id=origin +0ms
/express:router <anonymous> /script.js : /script.js?id=origin +0ms
/express:router dispatching GET /script.js?id=remote +12ms
/express:router query : /script.js?id=remote +0ms
/express:router expressInit : /script.js?id=remote +0ms
/express:router trim prefix /script.js from url /script.js?id=remote +0ms
/express:router <anonymous> /script.js : /script.js?id=remote +0ms
/express:router dispatching GET /favicon.ico +10ms
/express:router query : /favicon.ico +0ms
/express:router expressInit : /favicon.ico +0ms
/express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
/express:router dispatching GET /user=semchuk2 +13s
/express:router query : /user=semchuk2 +0ms
/express:router expressInit : /user=semchuk2 +0ms
/express:router <anonymous> : /user=semchuk2 +0ms
/express:router dispatching GET /script.js?id=remote +20ms
/express:router query : /script.js?id=remote +0ms
/express:router expressInit : /script.js?id=remote +0ms
/express:router trim prefix /script.js from url /script.js?id=remote +0ms
/express:router <anonymous> /script.js : /script.js?id=remote +0ms
/express:router dispatching GET /script.js?id=origin +1ms
/express:router query : /script.js?id=origin +0ms
/express:router expressInit : /script.js?id=origin +0ms
/express:router trim prefix /script.js from url /script.js?id=origin +0ms
/express:router <anonymous> /script.js : /script.js?id=origin +0ms
/express:router dispatching GET /favicon.ico +17ms
/express:router query : /favicon.ico +0ms
/express:router expressInit : /favicon.ico +0ms
/express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
/express:router dispatching GET /user=semchuk2&csp=default-src%27none%27 +2s
/express:router query : /user=semchuk2&csp=default-src%29%27none%27 +0ms
/express:router expressInit : /user=semchuk2&csp=default-src%29%27none%27 +0ms
/express:router <anonymous> : /user=semchuk2&csp=default-src%29%27none%27 +0ms
/user=semchuk2&csp=default-src%29%27none%27
/express:router query : /user=semchuk2&csp=default-src%29%27none%27 +3s
/express:router expressInit : /user=semchuk2&csp=default-src%29%27none%27 +0ms
/express:router <anonymous> : /user=semchuk2&csp=default-src%29%27none%27 +0ms
/user=semchuk2&csp=default-src%29%27none%27
```

Here is example #3:



The screenshot shows a terminal window titled "maksim@Se...". It displays the output of the "date" and "whoami" commands. The "date" command shows the current date and time as "Mon 18 May 2021 08:20:35 PM PDT". The "whoami" command shows the user name as "semchuk2".

```
semchuk2@cs: ~/cs495-src/xss-csp - Google Chrome
ssh.cloud.google.com/projects/s21-websc-maksim-semchuk/zones/us-west1-b/instances/csp?authuser=0&hl=en_US&projectNumber=687583183204&useAdminProxy=true

express:router dispatching GET /script.js?id=origin +40ms
express:router query : /script.js?id=origin +0ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +0ms
/2id=origin
express:router dispatching GET /script.js?id=remote +0ms
express:router query : /script.js?id=remote +0ms
express:router expressInit : /script.js?id=remote +0ms
express:router trim prefix (/script.js) from url /script.js?id=remote +0ms
express:router <anonymous> /script.js : /script.js?id=remote +0ms
/2id=remote
express:router dispatching GET /favicon.ico +10ms
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +10ms
express:router <anonymous> /favicon.ico +0ms
/favicon.ico
express:router dispatching GET /user=semchuk2 +58ms
express:router query : /user=semchuk2 +0ms
express:router expressInit : /user=semchuk2 +0ms
express:router <anonymous> /user=semchuk2 +0ms
/user=semchuk2
express:router dispatching GET /script.js?id=origin +36ms
express:router query : /script.js?id=origin +0ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +1ms
/2id=origin
express:router dispatching GET /script.js?id=remote +12ms
express:router query : /script.js?id=remote +0ms
express:router expressInit : /script.js?id=remote +0ms
express:router trim prefix (/script.js) from url /script.js?id=remote +0ms
express:router <anonymous> /script.js : /script.js?id=remote +0ms
/2id=remote
express:router dispatching GET /favicon.ico +10ms
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +0ms
express:router <anonymous> /favicon.ico +0ms
/favicon.ico
express:router dispatching GET /user=semchuk2 +13ms
express:router query : /user=semchuk2 +0ms
express:router expressInit : /user=semchuk2 +0ms
express:router <anonymous> /user=semchuk2 +0ms
/user=semchuk2
express:router dispatching GET /script.js?id=remote +26ms
express:router query : /script.js?id=remote +0ms
express:router expressInit : /script.js?id=remote +0ms
express:router trim prefix (/script.js) from url /script.js?id=remote +0ms
express:router <anonymous> /script.js : /script.js?id=remote +0ms
/2id=remote
express:router dispatching GET /script.js?id=origin +1ms
express:router query : /script.js?id=origin +0ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +0ms
/2id=origin
express:router dispatching GET /favicon.ico +17ms
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +0ms
express:router <anonymous> /favicon.ico +0ms
/favicon.ico
express:router dispatching GET /user=semchuk2&csp=default-src%20%27none%27 +2ms
express:router query : /user=semchuk2&csp=default-src%20%27none%27 +1ms
express:router expressInit : /user=semchuk2&csp=default-src%20%27none%27 +0ms
express:router <anonymous> /user=semchuk2&csp=default-src%20%27none%27 +0ms
/user=semchuk2&csp=default-src%20%27none%27
express:router dispatching GET /user=semchuk2&csp/default-src%20%27none%27 +3s
express:router query : /user=semchuk2&csp/default-src%20%27none%27 +0ms
express:router expressInit : /user=semchuk2&csp/default-src%20%27none%27 +1ms
express:router <anonymous> /user=semchuk2&csp/default-src%20%27none%27 +0ms
/user=semchuk2&csp=default-src%20%27self%27
express:router dispatching GET /script.js?id=origin +24ms
express:router query : /script.js?id=origin +0ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +0ms
/2id=origin
express:router dispatching GET /favicon.ico +10ms
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +0ms
express:router <anonymous> /favicon.ico +0ms
/favicon.ico
```

These differ because they only allow the user who owns the project to be able to connect.

Example #4:

The terminal window displays the source code for a middleware function named `script.js`. The code uses the `express` module to handle various HTTP requests, including `/favicon.ico`, `/?id=origin`, and `/?id=remote`. It includes logic for `query`, `expressInit`, and `expressInherit` handling, as well as `script.js?id=origin` and `script.js?id=remote` processing. The browser window shows a test page at `34.83.80.217:4321/?user=semchuk2&csp=default-src%20%27self%27%20script.js?id=origin+0ms` with the message "Hello, semchuk2". Below the message, there are three status logs: "changed by inline script", "changed by origin script", and "is this going to be changed by remote script?".

```
semchuk2@csp: ~/cs495-src/css-csp - Google Chrome
ssh.cloud.google.com/projects/s21-websec-maksim-semchuk/zones/us-west1-b/instances/csp?authuser=0&hl=en_US&projectNumber=687583183204&useAdminProxy=true

express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
express:router dispatching GET /user=semchuk2 +58s
express:router query : /user=semchuk2
express:router expressInit : /user=semchuk2 +0ms
express:router <anonymous> : /user=semchuk2 +0ms
/?user=semchuk2
express:router dispatching GET /script.js?id=origin +30ms
express:router query : /script.js?id=origin +10ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +1ms
/?id=origin
express:router dispatching GET /script.js?id=remote +12ms
express:router query : /script.js?id=remote +10ms
express:router expressInit : /script.js?id=remote +0ms
express:router trim prefix (/script.js) from url /script.js?id=remote +0ms
express:router <anonymous> /script.js : /script.js?id=remote +1ms
/?id=remote
express:router dispatching GET /favicon.ico +10ms
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +0ms
express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
express:router dispatching GET /user=semchuk2 +13s
express:router query : /user=semchuk2 +1ms
express:router expressInit : /user=semchuk2 +0ms
express:router <anonymous> : /user=semchuk2 +0ms
/?user=semchuk2
express:router dispatching GET /script.js?id=remote +20ms
express:router query : /script.js?id=remote +1ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=remote +0ms
/?2id=remote
express:router dispatching GET /script.js?id=origin +1ms
express:router query : /script.js?id=origin +0ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +0ms
/?2id=origin
express:router dispatching GET /favicon.ico +17s
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +0ms
express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
express:router dispatching GET /user=semchuk2&csp=default-src%20%27none%27 +2s
express:router query : /user=semchuk2&csp=default-src%20%27none%27 +1ms
express:router expressInit : /user=semchuk2&csp=default-src%20%27none%27 +0ms
express:router <anonymous> : /user=semchuk2&csp=default-src%20%27none%27 +0ms
/?user=semchuk2&csp=default-src%20%27none%27
express:router dispatching GET /user=semchuk2&csp=default-src%20%27none%27 +3s
express:router query : /user=semchuk2&csp=default-src%20%27none%27 +1ms
express:router expressInit : /user=semchuk2&csp=default-src%20%27none%27 +1ms
express:router <anonymous> : /user=semchuk2&csp=default-src%20%27none%27 +0ms
/?user=semchuk2&csp=default-src%20%27none%27
express:router dispatching GET /user=semchuk2&csp=default-src%20%27self%27 +3s
express:router query : /user=semchuk2&csp=default-src%20%27self%27 +1ms
express:router expressInit : /user=semchuk2&csp=default-src%20%27self%27 +0ms
express:router <anonymous> : /user=semchuk2&csp=default-src%20%27self%27 +0ms
/?user=semchuk2&csp=default-src%20%27self%27
express:router dispatching GET /script.js?id=origin +24ms
express:router query : /script.js?id=origin +1ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +0ms
/?id=origin
express:router dispatching GET /favicon.ico +17s
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +0ms
express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
express:router dispatching GET /user=semchuk2&csp=default-src%20%27self%27;%20script-src%20%27self%27%20%27unsafe-inline%27 +2s
express:router query : /user=semchuk2&csp=default-src%20%27self%27;%20script-src%20%27self%27%20%27unsafe-inline%27 +0ms
express:router expressInit : /user=semchuk2&csp=default-src%20%27self%27;%20script-src%20%27self%27%20%27unsafe-inline%27 +0ms
express:router <anonymous> : /user=semchuk2&csp=default-src%20%27self%27;%20script-src%20%27self%27%20%27unsafe-inline%27 +1ms
/?user=semchuk2&csp=default-src%20%27self%27;%20script-src%20%27self%27;%20script-src%20%27self%27%20%27unsafe-inline%27
express:router dispatching GET /script.js?id=origin +2ms
express:router query : /script.js?id=origin +0ms
express:router expressInit : /script.js?id=origin +0ms
express:router trim prefix (/script.js) from url /script.js?id=origin +0ms
express:router <anonymous> /script.js : /script.js?id=origin +0ms
/?id=script
express:router dispatching GET /favicon.ico +16s
express:router query : /favicon.ico +0ms
express:router expressInit : /favicon.ico +0ms
express:router <anonymous> : /favicon.ico +0ms
/favicon.ico
```

Here, it allows the unsafe inline arguments to be passed in. This means that this is less secure than the previous example.

3.3.1-2 no-defenses

The screenshot shows a browser window for the Web Security Academy. The URL is https://ac511f671e0579b3805409ae00df003c.web-security-academy.net/my-account. The page title is "CSRF vulnerability with no defenses". A green button at the top right says "LAB Solved". Below it, a message says "Congratulations, you solved the lab!". To the right are links for "Share your skills!" and "Continue learning >". At the bottom, there's a "My Account" section showing the username "wiener" and email "wiener@normal-user.net". An "Update email" button is present. To the right, a terminal window shows a shell session with the command "date && whoami" and output "Mon 10 May 2021 08:20:35 PM PDT semchuk2 semchuk2@ada:~\$".

This is a simple thing that allows the users to change email address, but since this does not protect from redirects then it allows a server to do this automatically allowing an attacker to gain access to a users account.

3.3.3 token-validation-depends-on-request-method

The screenshot shows a browser window for the 'WebSecurity Academy' website at <https://aceff10a1ec74c69808b090200fa004e.web-security-academy.net/my-account>. The page title is 'CSRF where token validation depends on request method'. A green button indicates the task is 'Solved'. Below the title, there's a message: 'Congratulations, you solved the lab!' with options to 'Share your skills!' or 'Continue learning >>'. At the bottom right, there are links to 'Home', 'My account', and 'Log out'. On the left, under 'My Account', it shows the user's current email as 'wiener@normal-user.net'. There is a text input field labeled 'Email' with a placeholder 'Email' and a green 'Update email' button below it. To the right, a terminal window titled 'maksim@Se...' shows a shell session: 'senchuk2@ada:~\$ date && whoami', followed by the output 'Mon 18 May 2021 10:33:51 PM PDT' and 'senchuk2'. This demonstrates a successful privilege escalation or information disclosure.

This is a simple thing that allows the users to change email address, but since this does not protect from redirects then it allows a server to do this automatically allowing an attacker to gain access to a users account. This is to show that you **MUST** secure both gets and posts.

3.3.4 token-not-tied-to-user-session

The screenshot shows a browser window with the URL <https://aca81ffe1e4fabd380b408de004e001c.web-security-academy.net>. The page title is "CSRF where token is not tied to user session". A green button indicates the task is "Solved". Below the title, there's a link "Back to lab description >>". A banner at the top says "Congratulations, you solved the lab!". To the right of the banner are links for "Share your skills!" and "Continue learning >>". At the bottom, there are links for "Home" and "My account". The main content area features a logo with the text "WE LIKE TO BLOG" and two coffee cups (one brown, one green) with matching sleeves. To the right of the logo is a terminal window showing a Linux command-line interface with the output:

```
maksim@Se... ~$ date && whoami
Mon 10 May 2021 10:33:51 PM PDT
semchuk2
semchuk2@maksim:~$
```

The weakness here, is that due to the fact that csrf tokens are not tied to the user, and the session leading to a vulnerability in allowing another user to claim the csrf token and present it as their own.

3.3.5 token-duplicated-in-cookie

Filter Headers

Block | Reser

▶ GET https://aced1fe31e4f450080da1e5300e80051.web-security-academy.net/?search=semchuk2

Status	200 OK ⓘ
Version	HTTP/1.1
Transferred	1.19 KB (3.08 KB size)
Referrer Policy	strict-origin-when-cross-origin

▼ Response Headers (200 B) Raw

- ② Connection: close
- ② Content-Encoding: gzip
- ② Content-Length: 1017
- ② Content-Type: text/html; charset=utf-8
- ② Set-Cookie: LastSearchTerm=semchuk2; Secure; HttpOnly
- ② X-XSS-Protection: 0

▼ Request Headers (615 B) Raw

- ② Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
- ② Accept-Encoding: gzip, deflate, br
- ② Accept-Language: en-US,en;q=0.5
- ② Cache-Control: no-cache
- ② Connection: keep-alive
- ② Cookie: session=5Yoq0moDVoWMUzDBgVrkml2b0Sg3YN4d; LastSearchTerm=semchuk2
- ② DNT: 1
- ② Host: aced1fe31e4f450080da1e5300e80051.web-security-academy.net
- ② Pragma: no-cache
- ② Referer: https://aced1fe31e4f450080da1e5300e80051.web-security-academy.net/?search=semchuk2
- ② Upgrade-Insecure-Requests: 1
- ② User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0

Picture of the request headers

How many cookies are returned? What are their names?

According to the content length, 1017.

How have foo and bar been interpreted?

They have been interpreted as a pair for usernames. And sets up for a new username.

How many cookies have been returned?

There was only 2 cookies returned.

Explain the results. Give one reason why a developer would choose to implement CSRF protection in this manner

This would be because they are trying to tie the user to the cookie, to prevent sharing of cookies.

What status code is returned? What is the value of the csrf field in the HTML form that is given back as a response?

A 200 , the cookie of semchuk2

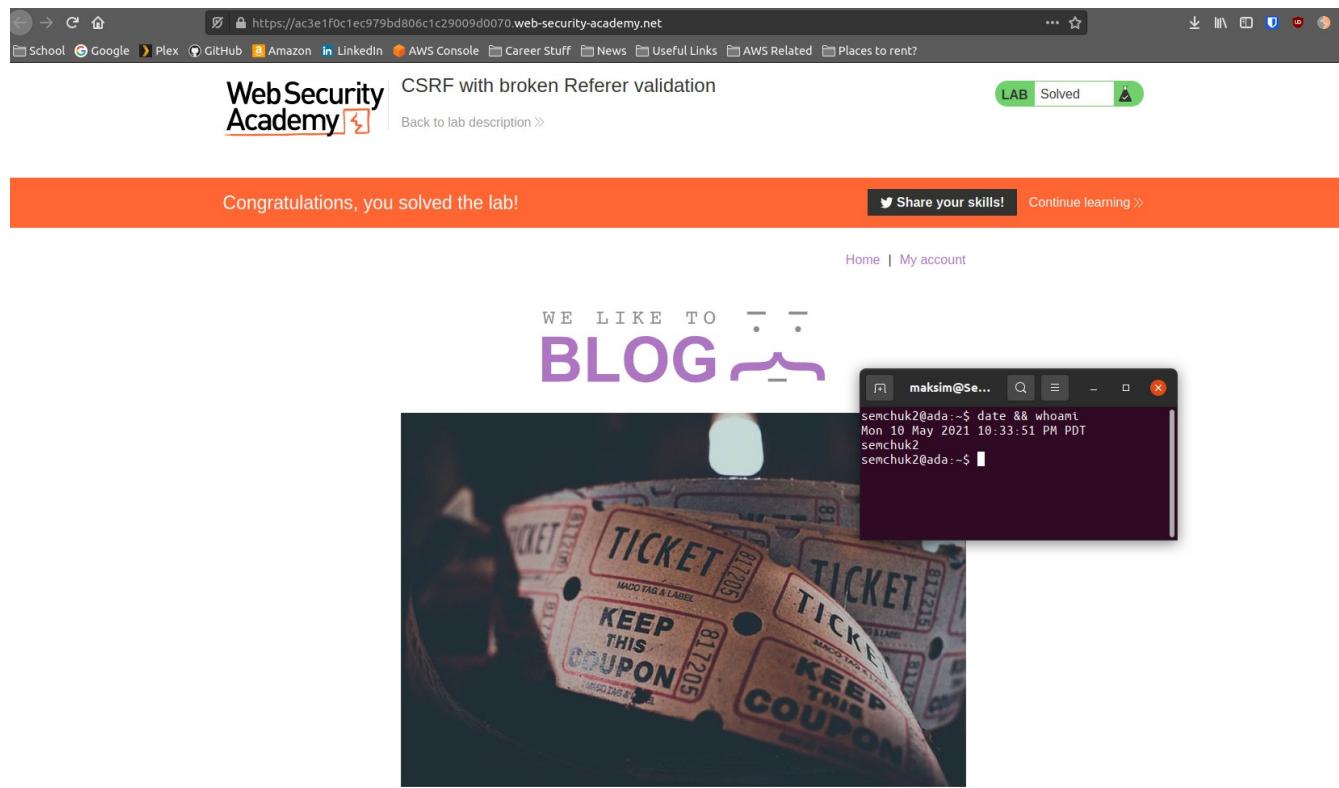
How might a developer use a keyed hash function on the server to prevent this request from succeeding without being forced to store each token?

They can create a hash to the key, and the users can provide the hash to ensure that it is correct.

3.3.9 referer-validation-broken

What status code and response text is returned?

HTTP status code: 400 with response text "Invalid referer header"



Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account

WE LIKE TO BLOG

TICKET COUPON

```
maksim@Se... maksm@Se... ~$ date && whoami
Mon 10 May 2021 10:33:51 PM PDT
maksim
maksim@ada:~$
```

The Peopleless Circus

When the leaflet dropped through my letterbox I thought it was a joke. The Peopleless Circus was in town. At the risk of sounding like a negative Nancy, I couldn't help thinking what is the world

The weakness here is that the referer can be set and overridden allowing a remote or 3rd party user to put their own and use it. This will allow a person to forge the identity to forge the csrf token and gain access since it never gets validated.

3.3.11 perform-csrf

The screenshot shows a browser window with the URL <https://ac9d1f561f26cf98809b11a5009200e9.web-security-academy.net/post?postId=3>. The page title is "Exploiting XSS to perform CSRF". A green button at the top right says "LAB Solved". Below it, a banner says "Congratulations, you solved the lab!" with options to "Share your skills!" and "Continue learning >". At the bottom, there's a "Home | My account" link and a terminal window showing a successful command execution.

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account

```
maksim@Se... ~
semchuk2@ada:~$ date && whoami
Tue 11 May 2021 12:14:41 AM PDT
semchuk2
semchuk2@ada:~$
```

21st Century Dreaming

This is due to the fact that you are able to use javascript to invoke functions and in fact use the javascript to parse the csrf token out and post it.

3.4.1 basic-csrf-protected

Congratulations, you solved the lab!

Share your skills! Continue learning >

Craft a response

URL: <https://ac231f971f825b6f80a810f701d80022.web-security-academy.net/exploit>

HTTPS

File: /exploit

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<style>
    iframe {
        position: relative;
        width: 700px;
        height: 500px;
        opacity: 0.3;
        z-index: 2;
    }
    div {
        position: absolute;
        top: 500px;
        left: 60px;
        z-index: 1;
    }
</style>
```

Store View exploit Deliver exploit to victim Access log

```
maksim@Se... ~$ date && whoami
Tue 11 May 2021 12:14:41 AM PDT
semchuk2
semchuk2@ada:~$
```

This works because we place a translucent button over the real one and then the user gets tricked into clicking it since they do not even know it is there,

3.4.4 prefilled-form-input

The screenshot shows a browser window for the 'WebSecurity Academy' lab titled 'Clickjacking with form input data prefilled from a URL parameter'. The URL is <https://ac3a1f011f08cf13805f188601980035.web-security-academy.net>. The page displays a message: 'Congratulations, you solved the lab!' with options to 'Share your skills!' and 'Continue learning >'. Below this, a section titled 'Craft a response' shows the exploit code. The code includes an 'HTTPS' checkbox checked, a 'File:' input field containing '/exploit', and a 'Head:' section with the following content:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

The 'Body:' section contains the following CSS and HTML:

```
<style>
  iframe {
    position: relative;
    width: 700px;
    height: 500px;
    opacity: 0.3;
    z-index: 2;
  }
  div {
    position: absolute;
    top: 450px;
    left: 60px;
    z-index: 1;
  }
</style>
```

On the right side of the interface, there is a terminal window showing the command `date && whoami` being run on a Linux system, with the output "Tue 11 May 2021 12:14:41 AM PDT" and the user "semchuk2". At the bottom of the interface are four buttons: 'Store', 'View exploit', 'Deliver exploit to victim', and 'Access log'.

This works because we place a translucent button over the real one and then the user gets tricked into clicking it since they do not even know it is there, except here the victims email address becomes pre filled by us.

3.4.5 trigger-dom-based-xss

The screenshot shows a web browser window with the URL <https://ac421f241fb2e68a80710e4c019c000e.web-security-academy.net/exploit>. The page displays an email form with 'Subject' set to 'foo' and 'Message' set to 'bar'. Below the form is a button labeled 'Test me'. To the right of the browser is a terminal window titled 'maksim@Se...'. The terminal output shows the command 'date && whoami' being run, with the response 'Tue 11 May 2021 12:14:41 AM PDT' and 'semchuk2'.

This did not work, despite it being right over the button. I am not sure why.s

3.5

Highlight the coordinates you entered previously via the UI in the decoded output for your lab notebook

```
maksim@SemchukPC:~$ base64 -d decode.txt
```

```
a:11:{i:0;a:4:{s:2:"x1";s:0:"";s:2:"y1";s:0:"";s:2:"x2";s:3:"100";s:2:"y2";s:3:"100";}i:1;a:4:  
{s:2:"x1";s:1:"1";s:2:"y1";s:1:"1";s:2:"x2";s:2:"99";s:2:"y2";s:2:"99";}i:2;a:4:  
{s:2:"x1";s:1:"1";s:2:"y1";s:1:"1";s:2:"x2";s:2:"99";s:2:"y2";s:2:"99";}i:3;a:4:  
{s:2:"x1";s:0:"";s:2:"y1";s:0:"";s:2:"x2";s:3:"100";s:2:"y2";s:3:"100";}i:4;a:0:{}i:5;a:4:  
{s:2:"x1";s:1:"0";s:2:"y1";s:1:"0"
```

3.6