IFS4103: Penetration Testing Practice

Lecture 4: Supplementary Slides on XSS

Outline

- Objectives
- Resources on Portswigger’s Site
- Covered XSS Concepts
- Using Burp Suite for Testing XSS
  (*With Similar Video Demos!)*

Objectives

Objectives of This Slide Deck

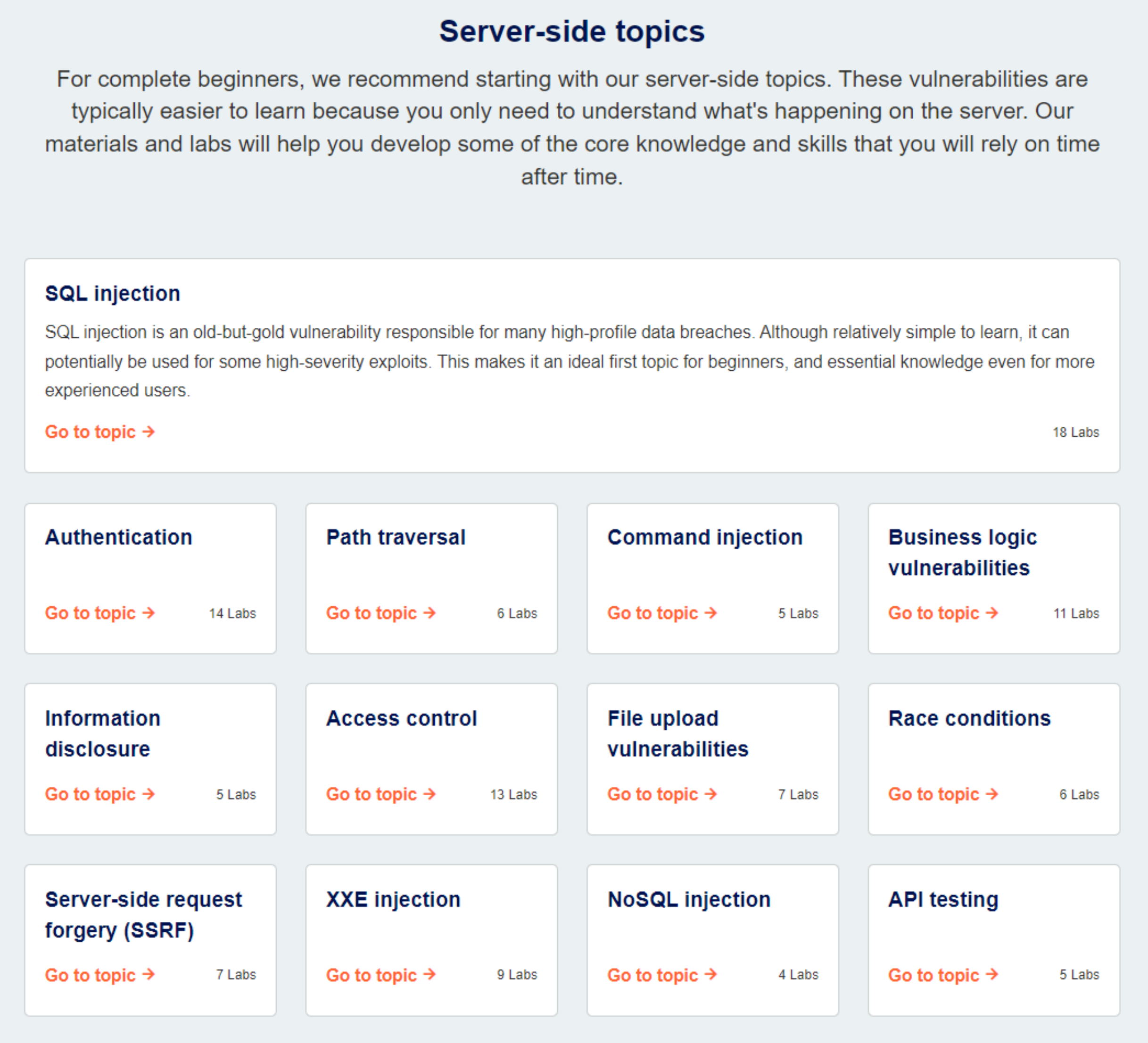
- To *supplement* the lecture & demonstration of XSS given by Ensign in Week 4:
  - Some of you mentioned that the given demo was **rather fast**
  - There were **no slides/notes** given on the demo
- To *share/highlight* **useful resources** that you can find within Portswigger's Web Security Academy site:
  - Review of various **web vulnerabilities**, including XSS
  - Steps and videos of using Burp Suite to **test & exploit XSS**
- To provide you with a *summary* of Lecture 4's class

Resources on Portswigger’s Site

Resources on Portswigger Site

- **Useful resources** within Portswigger's Web Security Academy:
  - https://portswigger.net/web-security/all-topics
    - **Server-side topics**:
      SQL injection, authentication, path traversal, command injection,
      business logic vulnerabilities, information disclosure, access control,
      file upload vulnerabilities, race conditions, server-side request forgery (SSRF),
      XXE injection, NoSQL injection, API testing
    - **Client-side topics**:
      Cross-site scripting, Cross-site request forgery (CSRF), Cross-origin resource
      sharing (CORS), clickjacking, DOM-based vulnerabilities, WebSockets
    - **Advanced topics**
  - https://portswigger.net/web-security/all-materials

Web Security Academy: All Topics



Server-side topics

For complete beginners, we recommend starting with our server-side topics. These vulnerabilities are typically easier to learn because you only need to understand what's happening on the server. Our materials and labs will help you develop some of the core knowledge and skills that you will rely on time after time.

SQL injection

SQL injection is an old-but-gold vulnerability responsible for many high-profile data breaches. Although relatively simple to learn, it can potentially be used for some high-severity exploits. This makes it an ideal first topic for beginners, and essential knowledge even for more experienced users.

Go to topic → 18 Labs

Authentication

Go to topic → 14 Labs

Path traversal

Go to topic → 6 Labs

Command injection

Go to topic → 5 Labs

Business logic vulnerabilities

Go to topic → 11 Labs

Information disclosure

Go to topic → 5 Labs

Access control

Go to topic → 13 Labs

File upload vulnerabilities

Go to topic → 7 Labs

Race conditions

Go to topic → 6 Labs

Server-side request forgery (SSRF)

Go to topic → 7 Labs

XXE injection

Go to topic → 9 Labs

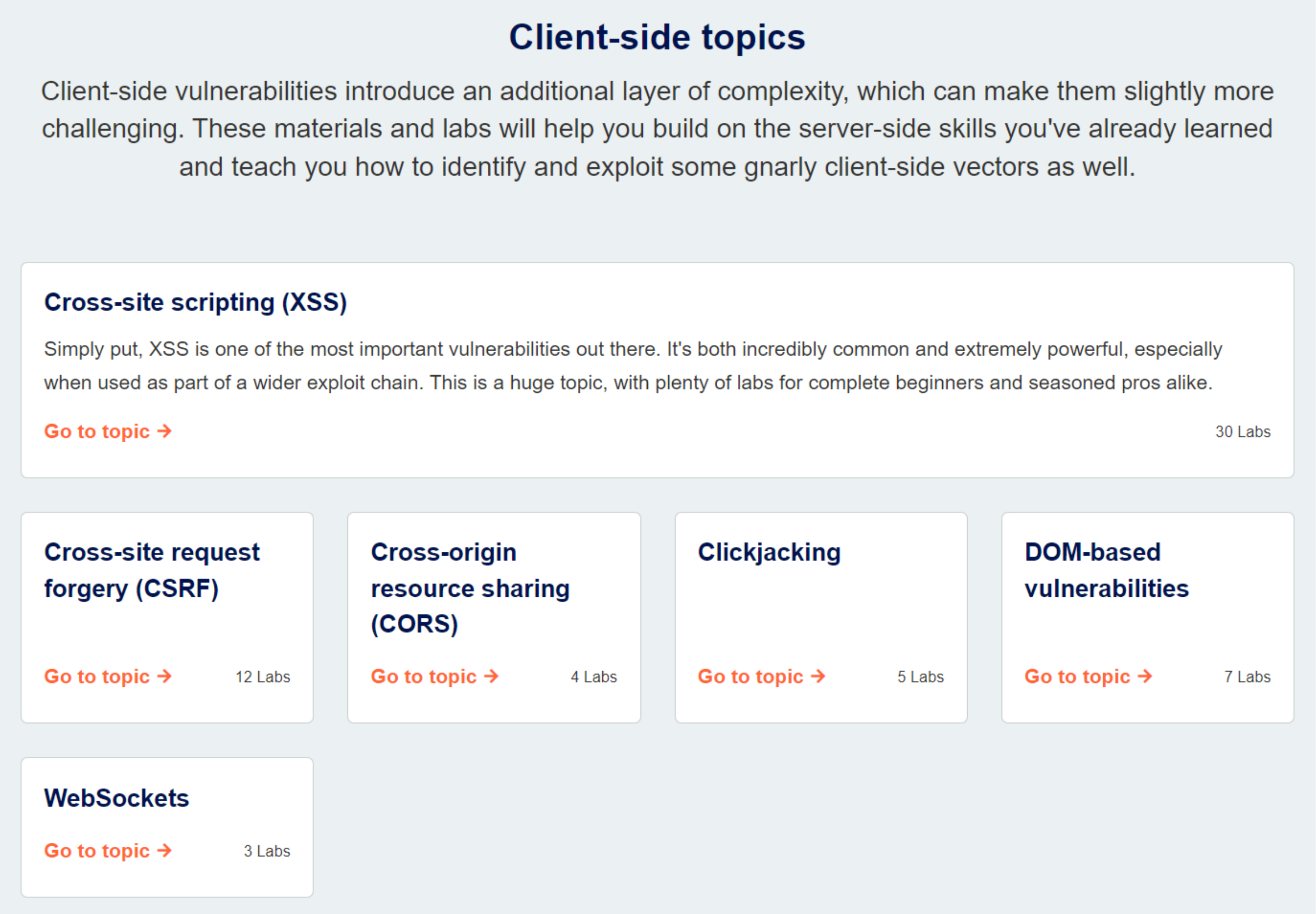
NoSQL injection

Go to topic → 4 Labs

API testing

Go to topic → 5 Labs

Web Security Academy: All Topics



Client-side topics

Client-side vulnerabilities introduce an additional layer of complexity, which can make them slightly more challenging. These materials and labs will help you build on the server-side skills you've already learned and teach you how to identify and exploit some gnarly client-side vectors as well.

Cross-site scripting (XSS)

Simply put, XSS is one of the most important vulnerabilities out there. It's both incredibly common and extremely powerful, especially when used as part of a wider exploit chain. This is a huge topic, with plenty of labs for complete beginners and seasoned pros alike.

Go to topic → 30 Labs

Cross-site request forgery (CSRF)

Go to topic → 12 Labs

Cross-origin resource sharing (CORS)

Go to topic → 4 Labs

Clickjacking

Go to topic → 5 Labs

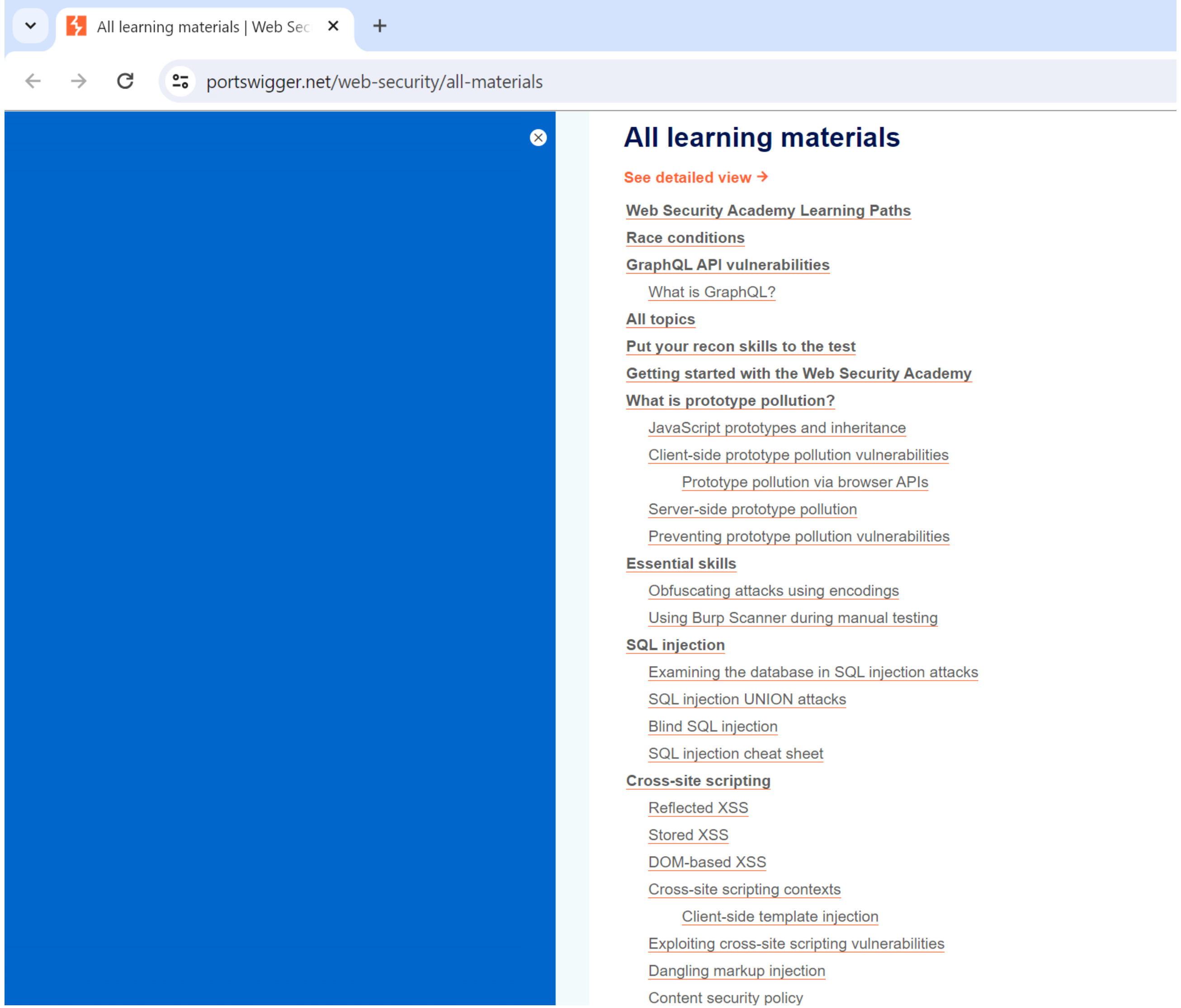
DOM-based vulnerabilities

Go to topic → 7 Labs

WebSockets

Go to topic → 3 Labs

Web Security Academy: All Materials



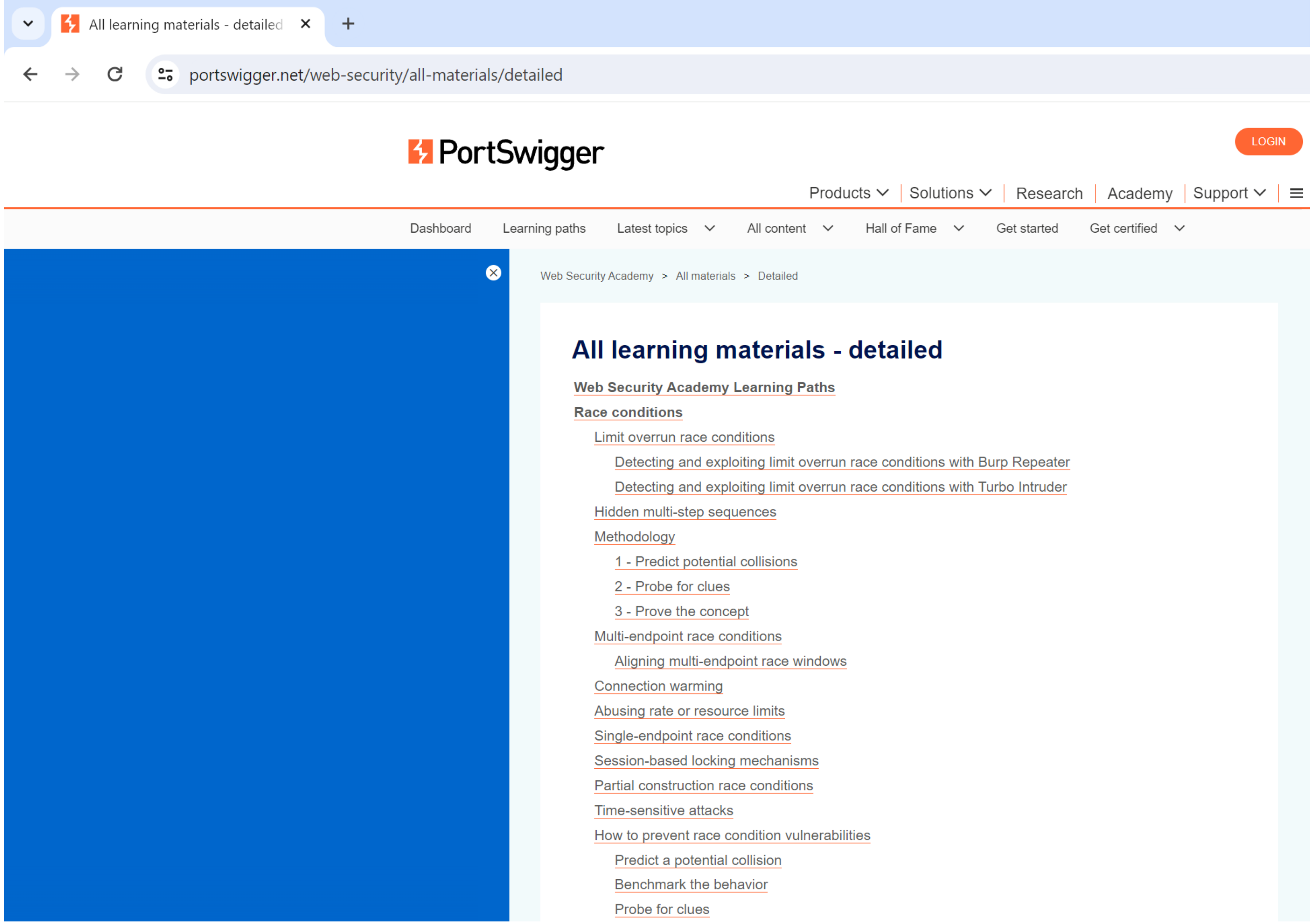
All learning materials | Web Sec
portswigger.net/web-security/all-materials

All learning materials

See detailed view →

- Web Security Academy Learning Paths
- Race conditions
- GraphQL API vulnerabilities
  - What is GraphQL?
- All topics
- Put your recon skills to the test
- Getting started with the Web Security Academy
- What is prototype pollution?
  - JavaScript prototypes and inheritance
  - Client-side prototype pollution vulnerabilities
    - Prototype pollution via browser APIs
  - Server-side prototype pollution
  - Preventing prototype pollution vulnerabilities
- Essential skills
  - Obfuscating attacks using encodings
  - Using Burp Scanner during manual testing
- SQL injection
  - Examining the database in SQL injection attacks
  - SQL injection UNION attacks
  - Blind SQL injection
  - SQL injection cheat sheet
- Cross-site scripting
  - Reflected XSS
  - Stored XSS
  - DOM-based XSS
  - Cross-site scripting contexts
    - Client-side template injection
  - Exploiting cross-site scripting vulnerabilities
  - Dangling markup injection
  - Content security policy

Web Security Academy: All Materials (Detailed View)



Web Security Academy > All materials > Detailed

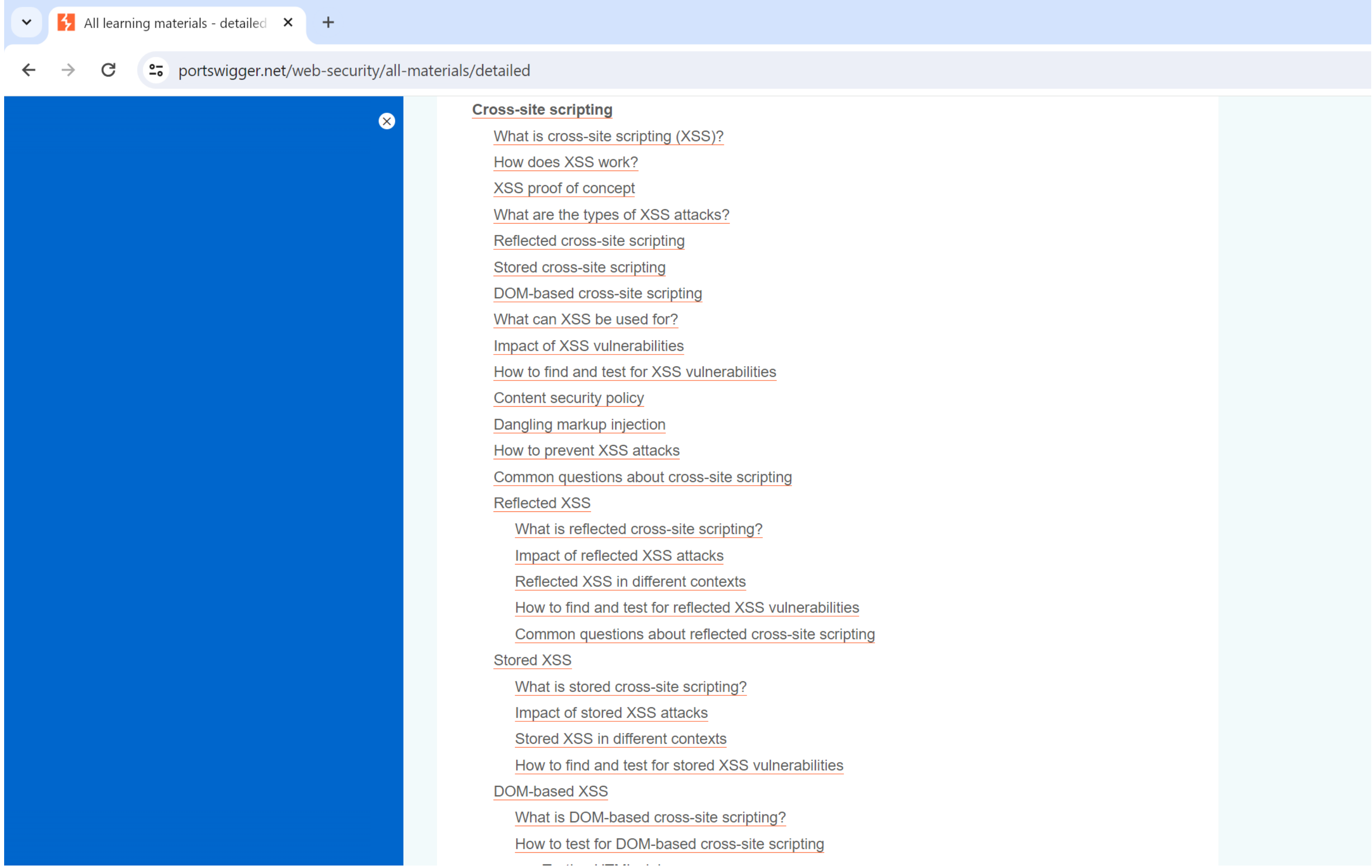
All learning materials - detailed

Web Security Academy Learning Paths

Race conditions

- Limit overrun race conditions
  - Detecting and exploiting limit overrun race conditions with Burp Repeater
  - Detecting and exploiting limit overrun race conditions with Turbo Intruder
- Hidden multi-step sequences
- Methodology
  - 1 - Predict potential collisions
  - 2 - Probe for clues
  - 3 - Prove the concept
- Multi-endpoint race conditions
  - Aligning multi-endpoint race windows
- Connection warming
- Abusing rate or resource limits
- Single-endpoint race conditions
- Session-based locking mechanisms
- Partial construction race conditions
- Time-sensitive attacks
- How to prevent race condition vulnerabilities
  - Predict a potential collision
  - Benchmark the behavior
  - Probe for clues

Web Security Academy: All Materials (Detailed View)

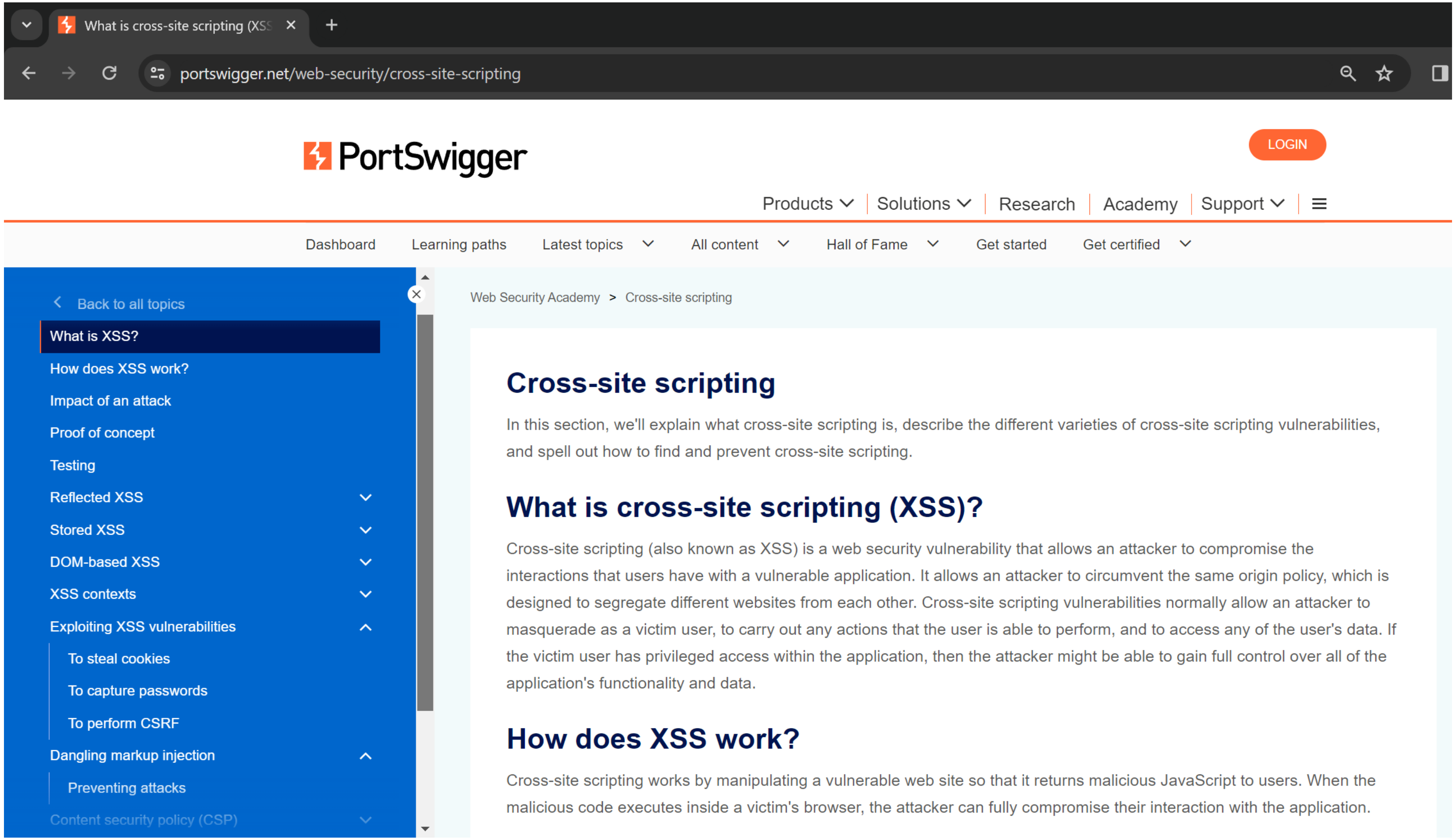

All learning materials - detailed
portswigger.net/web-security/all-materials/detailed

Cross-site scripting

- What is cross-site scripting (XSS)?
- How does XSS work?
- XSS proof of concept
- What are the types of XSS attacks?
- Reflected cross-site scripting
- Stored cross-site scripting
- DOM-based cross-site scripting
- What can XSS be used for?
- Impact of XSS vulnerabilities
- How to find and test for XSS vulnerabilities
- Content security policy
- Dangling markup injection
- How to prevent XSS attacks
- Common questions about cross-site scripting
- Reflected XSS
  - What is reflected cross-site scripting?
  - Impact of reflected XSS attacks
  - Reflected XSS in different contexts
  - How to find and test for reflected XSS vulnerabilities
  - Common questions about reflected cross-site scripting
- Stored XSS
  - What is stored cross-site scripting?
  - Impact of stored XSS attacks
  - Stored XSS in different contexts
  - How to find and test for stored XSS vulnerabilities
- DOM-based XSS
  - What is DOM-based cross-site scripting?
  - How to test for DOM-based cross-site scripting

Covered XSS Concepts

From Portswigger's All Topics/Materials



Cross-site scripting

In this section, we'll explain what cross-site scripting is, describe the different varieties of cross-site scripting vulnerabilities, and spell out how to find and prevent cross-site scripting.

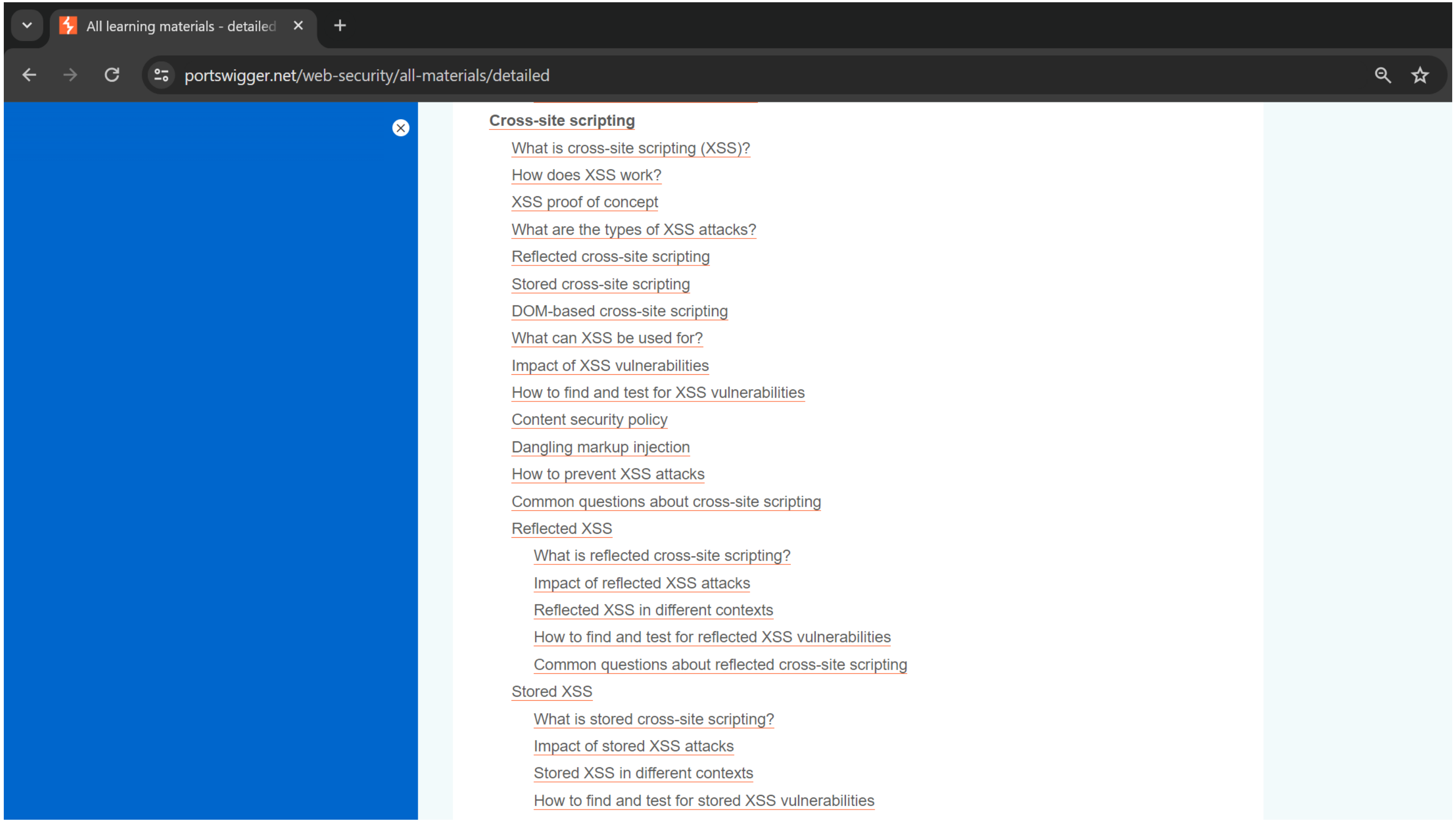
What is cross-site scripting (XSS)?

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data.

How does XSS work?

Cross-site scripting works by manipulating a vulnerable web site so that it returns malicious JavaScript to users. When the malicious code executes inside a victim's browser, the attacker can fully compromise their interaction with the application.

From Portswigger's All Topics/Materials

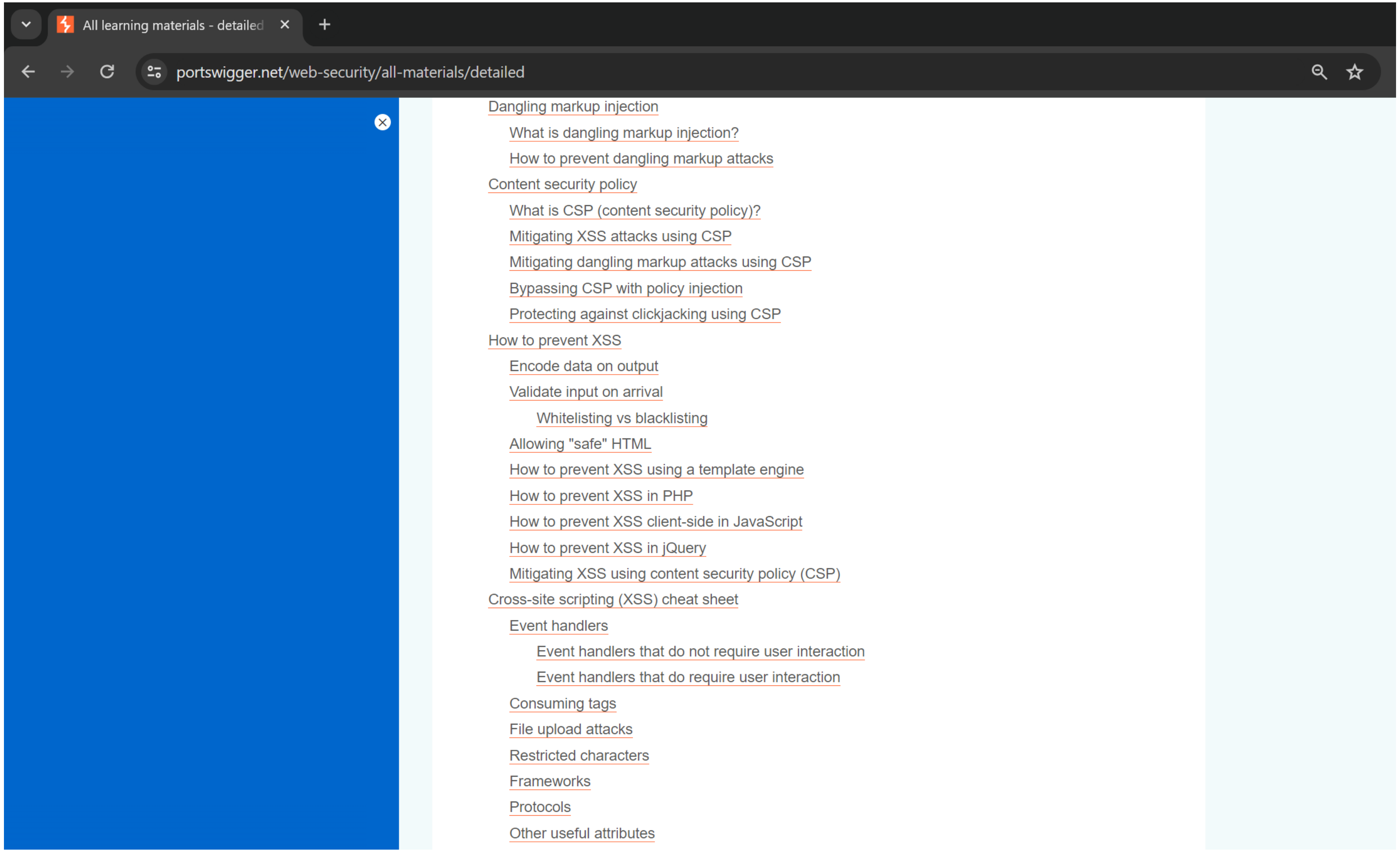


All learning materials - detailed
portswigger.net/web-security/all-materials/detailed

Cross-site scripting

- What is cross-site scripting (XSS)?
- How does XSS work?
- XSS proof of concept
- What are the types of XSS attacks?
- Reflected cross-site scripting
- Stored cross-site scripting
- DOM-based cross-site scripting
- What can XSS be used for?
- Impact of XSS vulnerabilities
- How to find and test for XSS vulnerabilities
- Content security policy
- Dangling markup injection
- How to prevent XSS attacks
- Common questions about cross-site scripting
- Reflected XSS
  - What is reflected cross-site scripting?
  - Impact of reflected XSS attacks
  - Reflected XSS in different contexts
  - How to find and test for reflected XSS vulnerabilities
  - Common questions about reflected cross-site scripting
- Stored XSS
  - What is stored cross-site scripting?
  - Impact of stored XSS attacks
  - Stored XSS in different contexts
  - How to find and test for stored XSS vulnerabilities

From Portswigger’s All Topics/Materials



All learning materials - detailed

portswigger.net/web-security/all-materials/detailed

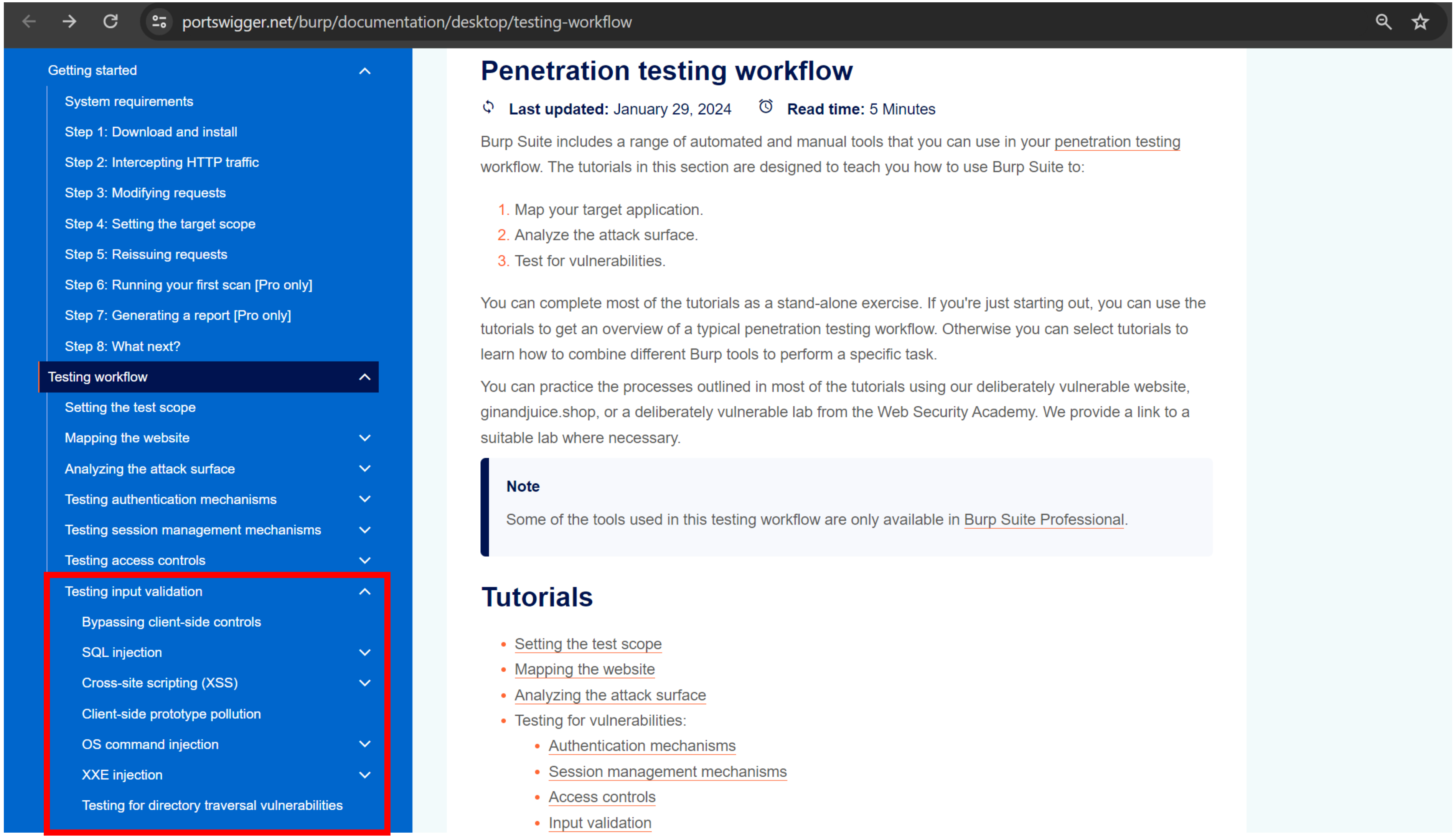
- Dangling markup injection
  - What is dangling markup injection?
  - How to prevent dangling markup attacks
- Content security policy
  - What is CSP (content security policy)?
  - Mitigating XSS attacks using CSP
  - Mitigating dangling markup attacks using CSP
  - Bypassing CSP with policy injection
  - Protecting against clickjacking using CSP
- How to prevent XSS
  - Encode data on output
  - Validate input on arrival
    - Whitelisting vs blacklisting
  - Allowing "safe" HTML
  - How to prevent XSS using a template engine
  - How to prevent XSS in PHP
  - How to prevent XSS client-side in JavaScript
  - How to prevent XSS in jQuery
  - Mitigating XSS using content security policy (CSP)
- Cross-site scripting (XSS) cheat sheet
  - Event handlers
    - Event handlers that do not require user interaction
    - Event handlers that do require user interaction
  - Consuming tags
  - File upload attacks
  - Restricted characters
  - Frameworks
  - Protocols
  - Other useful attributes

Using Burp Suite for Testing XSS: *With Similar Video Demos!*

Tutorials & Video Demos on Using Burp Suite

- You can refer to Burp Suite documentation – desktop editions:
- It contains "Penetration testing workflow", which covers testing for the following vulnerabilities:
  - Authentication mechanisms
  - Session management mechanisms
  - Access controls
  - **Input validation:** including SQLi, ***XSS***, OS command & XXE injections
  - Clickjacking
  - SSRF
  - WebSockets
  - Working with GraphQL in Burp Suite
  - Complementing your manual testing with Burp Scanner

From Portswigger's Burp Suite Documentation



Penetration testing workflow

Last updated: January 29, 2024 **Read time:** 5 Minutes

Burp Suite includes a range of automated and manual tools that you can use in your penetration testing workflow. The tutorials in this section are designed to teach you how to use Burp Suite to:

1. Map your target application.
2. Analyze the attack surface.
3. Test for vulnerabilities.

You can complete most of the tutorials as a stand-alone exercise. If you're just starting out, you can use the tutorials to get an overview of a typical penetration testing workflow. Otherwise you can select tutorials to learn how to combine different Burp tools to perform a specific task.

You can practice the processes outlined in most of the tutorials using our deliberately vulnerable website, ginandjuice.shop, or a deliberately vulnerable lab from the Web Security Academy. We provide a link to a suitable lab where necessary.

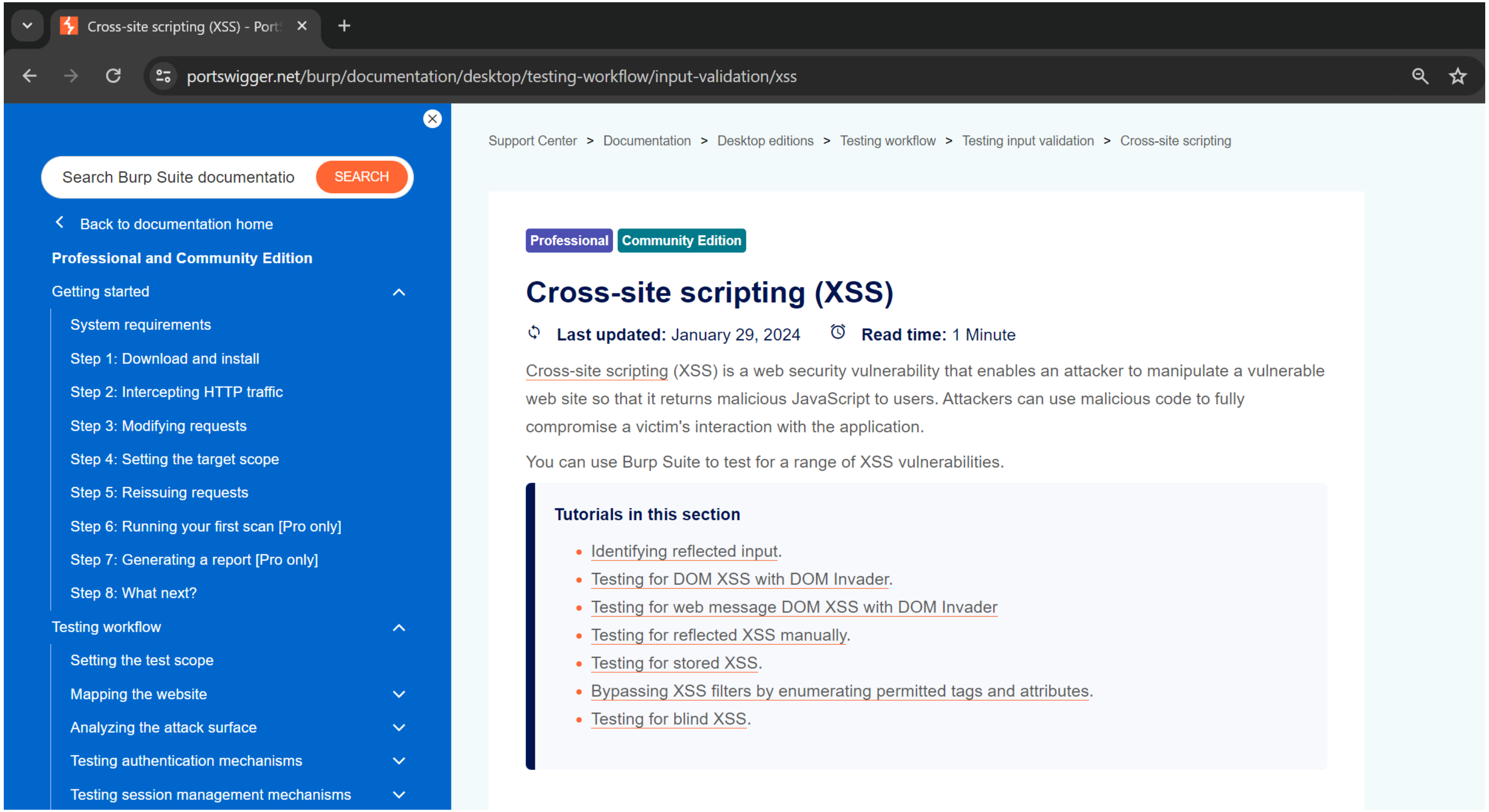
> **Note**
>
> Some of the tools used in this testing workflow are only available in Burp Suite Professional.

Tutorials

- Setting the test scope
- Mapping the website
- Analyzing the attack surface
- Testing for vulnerabilities:
  - Authentication mechanisms
  - Session management mechanisms
  - Access controls
  - Input validation

Testing XSS

Clickable link


Cross-site scripting (XSS) - Port
portswigger.net/burp/documentation/desktop/testing-workflow/input-validation/xss
Search Burp Suite documentatio
SEARCH
Back to documentation home
Professional and Community Edition
Getting started
System requirements
Step 1: Download and install
Step 2: Intercepting HTTP traffic
Step 3: Modifying requests
Step 4: Setting the target scope
Step 5: Reissuing requests
Step 6: Running your first scan [Pro only]
Step 7: Generating a report [Pro only]
Step 8: What next?
Testing workflow
Setting the test scope
Mapping the website
Analyzing the attack surface
Testing authentication mechanisms
Testing session management mechanisms
Support Center > Documentation > Desktop editions > Testing workflow > Testing input validation > Cross-site scripting
Professional
Community Edition

Cross-site scripting (XSS)

Last updated: January 29, 2024 **Read time:** 1 Minute

Cross-site scripting (XSS) is a web security vulnerability that enables an attacker to manipulate a vulnerable web site so that it returns malicious JavaScript to users. Attackers can use malicious code to fully compromise a victim's interaction with the application.

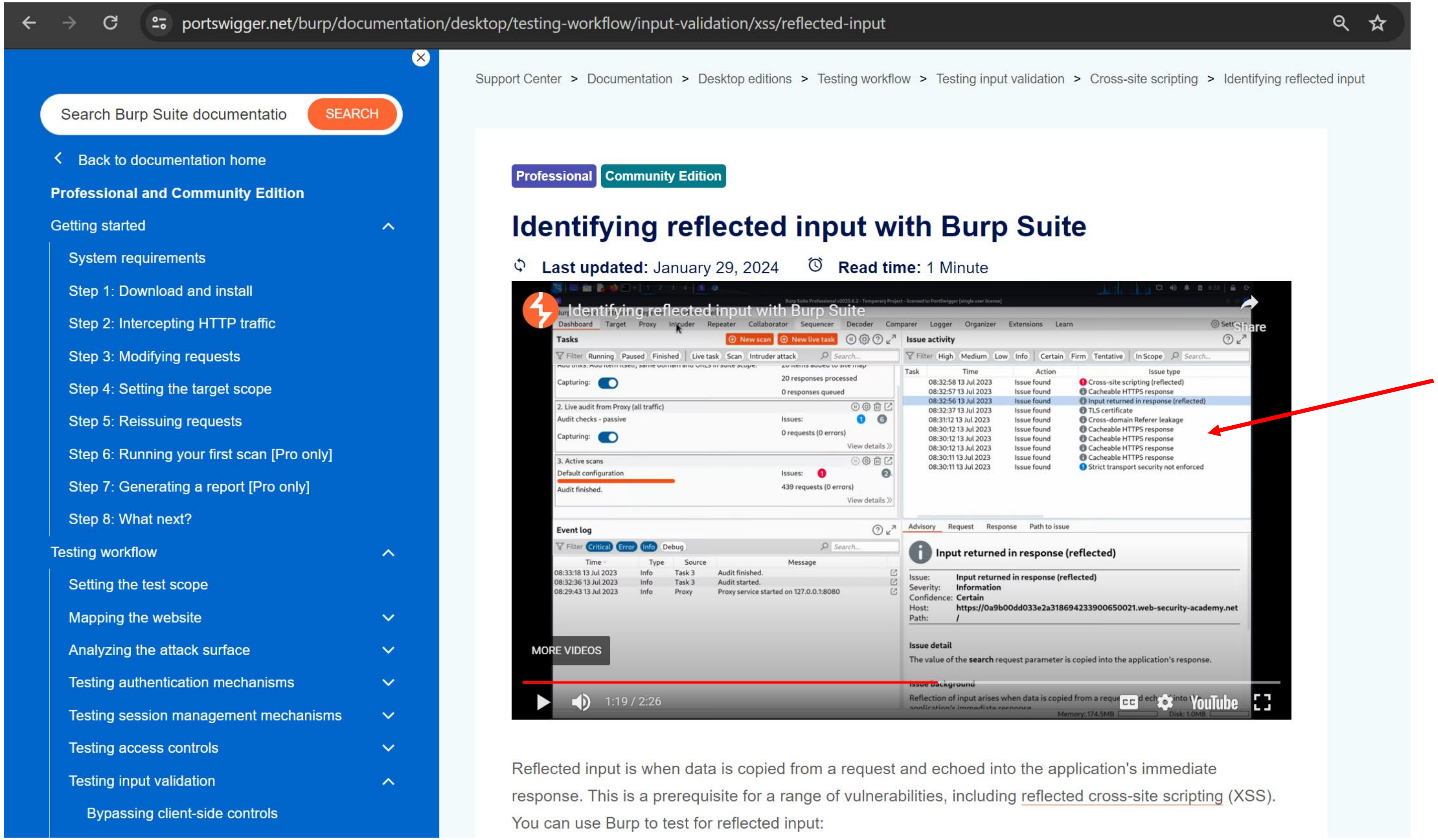
You can use Burp Suite to test for a range of XSS vulnerabilities.

Tutorials in this section

- Identifying reflected input.
- Testing for DOM XSS with DOM Invader.
- Testing for web message DOM XSS with DOM Invader
- Testing for reflected XSS manually.
- Testing for stored XSS.
- Bypassing XSS filters by enumerating permitted tags and attributes.
- Testing for blind XSS.

Identifying Reflected Input with Burp Suite

Clickable link



portswigger.net/burp/documentation/desktop/testing-workflow/input-validation/xss/reflected-input

Search Burp Suite documentatio SEARCH

- Back to documentation home
- **Professional and Community Edition**
- Getting started
  - System requirements
  - Step 1: Download and install
  - Step 2: Intercepting HTTP traffic
  - Step 3: Modifying requests
  - Step 4: Setting the target scope
  - Step 5: Reissuing requests
  - Step 6: Running your first scan [Pro only]
  - Step 7: Generating a report [Pro only]
  - Step 8: What next?
- Testing workflow
  - Setting the test scope
  - Mapping the website
  - Analyzing the attack surface
  - Testing authentication mechanisms
  - Testing session management mechanisms
  - Testing access controls
  - Testing input validation
    - Bypassing client-side controls

Support Center > Documentation > Desktop editions > Testing workflow > Testing input validation > Cross-site scripting > Identifying reflected input

Professional Community Edition

Identifying reflected input with Burp Suite

Last updated: January 29, 2024 **Read time:** 1 Minute

Issue activity

Task	Time	Action	Issue type
	08:32:58 13 Jul 2023	Issue found	Cross-site scripting (reflected)
	08:32:57 13 Jul 2023	Issue found	Cacheable HTTPS response
	08:32:56 13 Jul 2023	Issue found	Input returned in response (reflected)
	08:32:37 13 Jul 2023	Issue found	TLS certificate
	08:31:12 13 Jul 2023	Issue found	Cross-domain Referer leakage
	08:30:12 13 Jul 2023	Issue found	Cacheable HTTPS response
	08:30:12 13 Jul 2023	Issue found	Cacheable HTTPS response
	08:30:12 13 Jul 2023	Issue found	Cacheable HTTPS response
	08:30:11 13 Jul 2023	Issue found	Cacheable HTTPS response
	08:30:11 13 Jul 2023	Issue found	Strict transport security not enforced

Time	Type	Source	Message
08:33:18 13 Jul 2023	Info	Task 3	Audit finished.
08:32:36 13 Jul 2023	Info	Task 3	Audit started.
08:29:43 13 Jul 2023	Info	Proxy	Proxy service started on 127.0.0.1:8080

Advisory Request Response Path to issue

Input returned in response (reflected)

Issue: Input returned in response (reflected)
Severity: Information
Confidence: Certain
Host: https://0a9b00dd033e2a318694233900650021.web-security-academy.net
Path: /

Issue detail

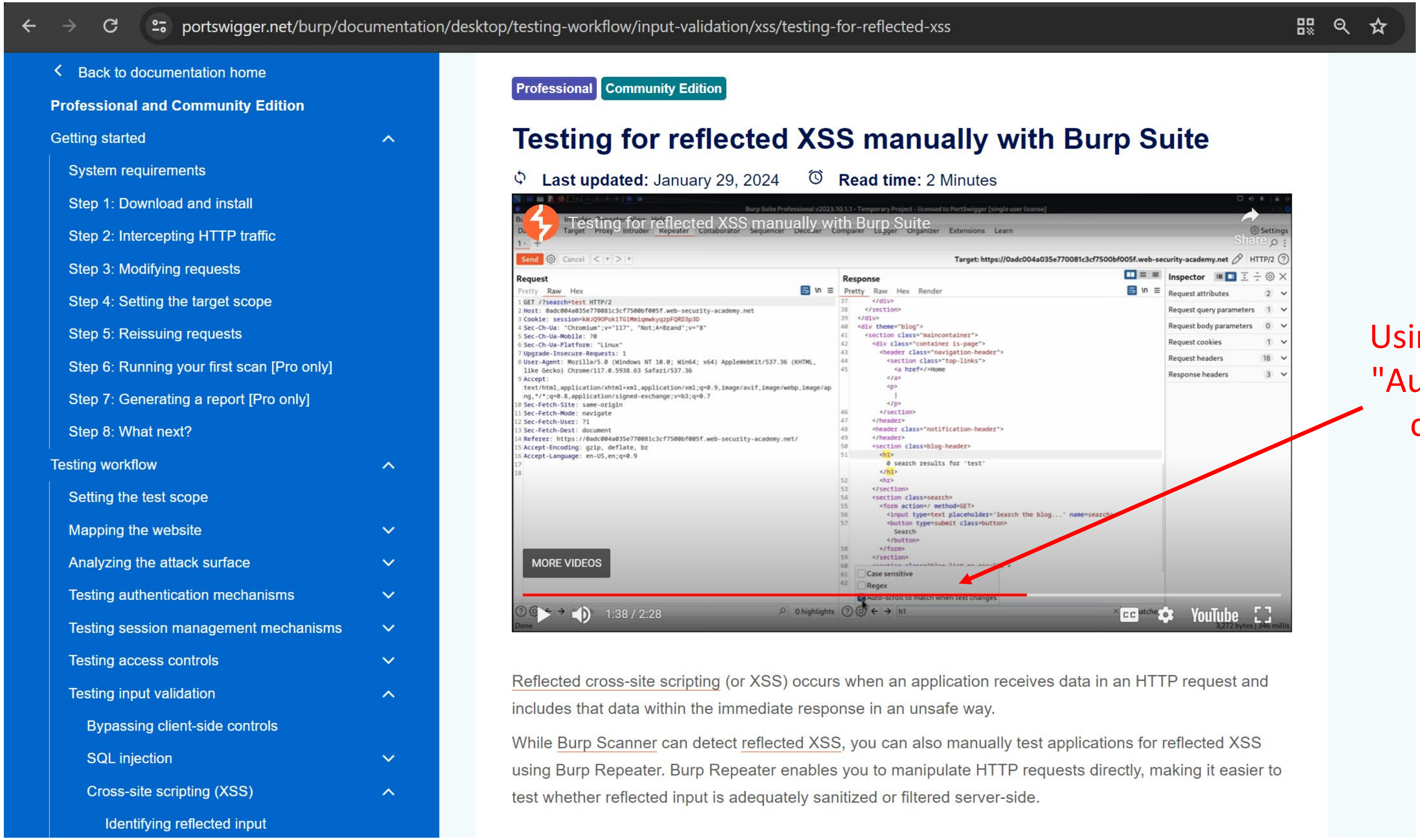
The value of the **search** request parameter is copied into the application's response.

Reflected input is when data is copied from a request and echoed into the application's immediate response. This is a prerequisite for a range of vulnerabilities, including reflected cross-site scripting (XSS). You can use Burp to test for reflected input:

Burp Scanner is used!

Testing for Reflected XSS Manually with Burp Suite

Clickable link



portswigger.net/burp/documentation/desktop/testing-workflow/input-validation/xss/testing-for-reflected-xss
Back to documentation home
Professional and Community Edition
Getting started
System requirements
Step 1: Download and install
Step 2: Intercepting HTTP traffic
Step 3: Modifying requests
Step 4: Setting the target scope
Step 5: Reissuing requests
Step 6: Running your first scan [Pro only]
Step 7: Generating a report [Pro only]
Step 8: What next?
Testing workflow
Setting the test scope
Mapping the website
Analyzing the attack surface
Testing authentication mechanisms
Testing session management mechanisms
Testing access controls
Testing input validation
Bypassing client-side controls
SQL injection
Cross-site scripting (XSS)
Identifying reflected input
Professional
Community Edition
Testing for reflected XSS manually with Burp Suite
Share
MORE VIDEOS
1:38 / 2:28
Auto-scroll to match when text changes
YouTube

Testing for reflected XSS manually with Burp Suite

Last updated: January 29, 2024 **Read time:** 2 Minutes

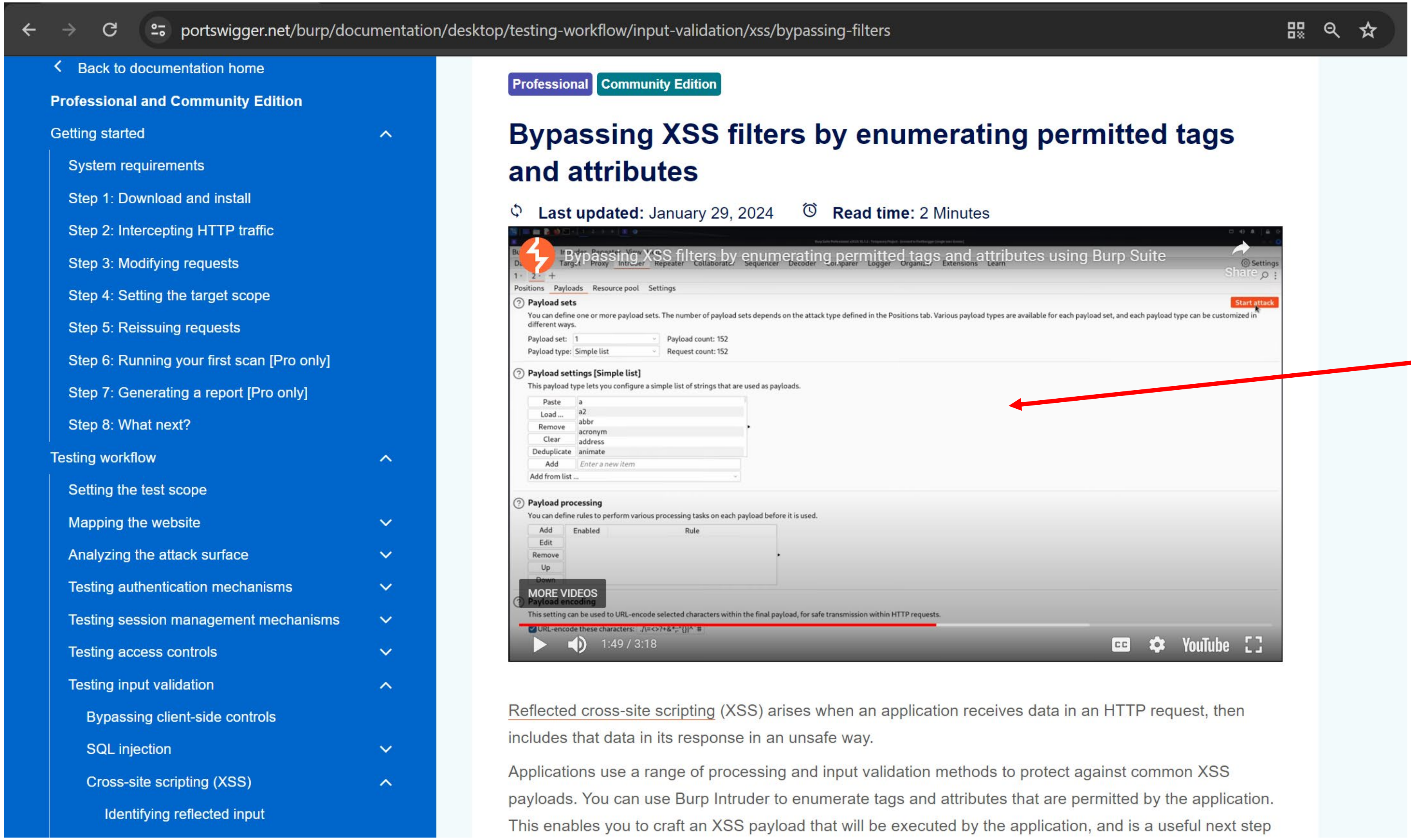
Reflected cross-site scripting (or XSS) occurs when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

While Burp Scanner can detect reflected XSS, you can also manually test applications for reflected XSS using Burp Repeater. Burp Repeater enables you to manipulate HTTP requests directly, making it easier to test whether reflected input is adequately sanitized or filtered server-side.

Using **Burp Repeater's** "Auto scroll when text changes" feature

Bypassing XSS Filters by Enumerating Permitted Tags and Attributes

Clickable link



portswigger.net/burp/documentation/desktop/testing-workflow/input-validation/xss/bypassing-filters

Back to documentation home

Professional and Community Edition

- Getting started
  - System requirements
  - Step 1: Download and install
  - Step 2: Intercepting HTTP traffic
  - Step 3: Modifying requests
  - Step 4: Setting the target scope
  - Step 5: Reissuing requests
  - Step 6: Running your first scan [Pro only]
  - Step 7: Generating a report [Pro only]
  - Step 8: What next?
- Testing workflow
  - Setting the test scope
  - Mapping the website
  - Analyzing the attack surface
  - Testing authentication mechanisms
  - Testing session management mechanisms
  - Testing access controls
  - Testing input validation
    - Bypassing client-side controls
    - SQL injection
    - Cross-site scripting (XSS)
      - Identifying reflected input

Professional | Community Edition

Bypassing XSS filters by enumerating permitted tags and attributes

Last updated: January 29, 2024 **Read time:** 2 Minutes

Bypassing XSS filters by enumerating permitted tags and attributes using Burp Suite

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 152

Payload type: Simple list Request count: 152

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

a
a2
abbr
acronym
address
animate

Enter a new item

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☑ URL-encode these characters: ./\=<>?+&*;:"{}|^`#

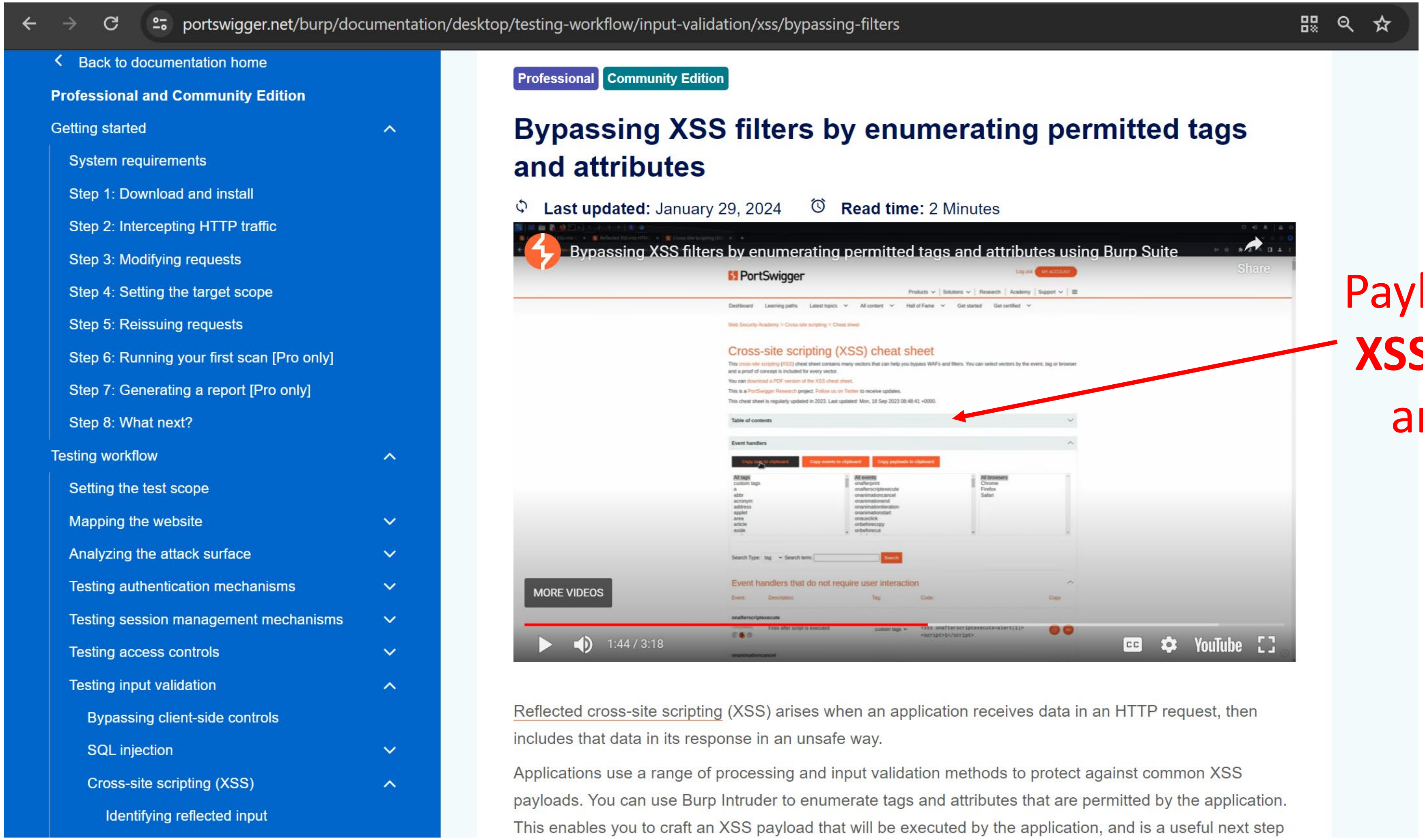
[Reflected cross-site scripting](#) (XSS) arises when an application receives data in an HTTP request, then includes that data in its response in an unsafe way.

Applications use a range of processing and input validation methods to protect against common XSS payloads. You can use Burp Intruder to enumerate tags and attributes that are permitted by the application. This enables you to craft an XSS payload that will be executed by the application, and is a useful next step

Burp Intruder is used!

Bypassing XSS Filters by Enumerating Permitted Tags and Attributes

Clickable link



portswigger.net/burp/documentation/desktop/testing-workflow/input-validation/xss/bypassing-filters
Back to documentation home
Professional and Community Edition
Getting started
System requirements
Step 1: Download and install
Step 2: Intercepting HTTP traffic
Step 3: Modifying requests
Step 4: Setting the target scope
Step 5: Reissuing requests
Step 6: Running your first scan [Pro only]
Step 7: Generating a report [Pro only]
Step 8: What next?
Testing workflow
Setting the test scope
Mapping the website
Analyzing the attack surface
Testing authentication mechanisms
Testing session management mechanisms
Testing access controls
Testing input validation
Bypassing client-side controls
SQL injection
Cross-site scripting (XSS)
Identifying reflected input
Professional
Community Edition
Bypassing XSS filters by enumerating permitted tags and attributes using Burp Suite
Share
PortSwigger
Cross-site scripting (XSS) cheat sheet
Table of contents
Event handlers
Copy tags to clipboard
Copy events to clipboard
Copy payloads to clipboard
Event handlers that do not require user interaction
MORE VIDEOS
1:44 / 3:18
YouTube

Bypassing XSS filters by enumerating permitted tags and attributes

Last updated: January 29, 2024 **Read time:** 2 Minutes

Reflected cross-site scripting (XSS) arises when an application receives data in an HTTP request, then includes that data in its response in an unsafe way.

Applications use a range of processing and input validation methods to protect against common XSS payloads. You can use Burp Intruder to enumerate tags and attributes that are permitted by the application. This enables you to craft an XSS payload that will be executed by the application, and is a useful next step

Payload lists from **XSS Cheat Sheet** are also used

Please utilize the available resources: You can become a Burp Expert too

Thanks!