Labs Overview

[Document subtitle]

# Table of Contents

[Table of Contents 1](#_Toc93479620)

[1. Lab01 2](#_Toc93479621)

[2. Lab02 2](#_Toc93479622)

[3. Lab03 2](#_Toc93479623)

[3.1. Lab01.php 2](#_Toc93479624)

[3.2. Lab02.php 3](#_Toc93479625)

[3.3. Lab03.php 3](#_Toc93479626)

[3.4. Lab04.php 3](#_Toc93479627)

[3.5. Lab05.php 3](#_Toc93479628)

[3.6. Lab06.php 4](#_Toc93479629)

[3.7. Lab07.php 4](#_Toc93479630)

[3.8. Lab08.php 4](#_Toc93479631)

[3.9. Lab09.php 4](#_Toc93479632)

[3.10. Lab10.php 5](#_Toc93479633)

[4. Lab04.php 5](#_Toc93479634)

[5. Lab05.php 5](#_Toc93479635)

[6. Lab06.php 5](#_Toc93479636)

[7. Lab08.php 5](#_Toc93479637)

[8. Lab09.php 6](#_Toc93479638)

[9. Lab11.php 6](#_Toc93479639)

# Lab01

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
| 13/10/21 | Firstly I changed the max length of one of the inputs:  <script>document.getElementsByTagName("input")[0].setAttribute("maxlength", 1000);</script>  Then used the following code in the input box:  <script>alert("This is an alert");</script> | Able to run any code through the browser by increasing the character limits of the inputs that are reflected out to the user.  FINDING: Inputs are not sanitised properly |
| **Mitigation** | | |
| **Sanitise all inputs for problem characters e.g. <,>,[,]…** | | |

# Lab02

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

# Lab03

## Lab01.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
| 27/10/21 | Payloads added to the address bar are reflected to the user:  ?name=<script>alert("this is an alert")</script> | Any code can be running by setting the name variable in the address bar and submitting.  FINDING: Reflective XSS |
| **Mitigation** | | |
| **Use POST instead of GET for requests will stop the address bar being able to be used to change variables** | | |

## Lab02.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
| 27/10/21 | Payloads added to the address bar are reflected to the user:  ?name=<<script>script>alert("this is an alert")<</script>/script> | As 3.1 but payload is sanitise to look for “<script>” and “</script>”. Inserting these tags inside the original tags will result in the original script tags not being found.  FINDING:Reflective XSS |
| **Mitigation** | | |
| **Use POST instead of GET for Requests and sanitise input variables properly** | | |

## Lab03.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | As 3.2 and also using a backreference to strip all occurances of “<script>” and “</script>”.  FINDING:Reflective XSS |
| **Mitigation** | | |
|  | | |

## Lab04.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

## Lab05.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
| 27/10/2021 | Used:  ?name=<script>al\u0065rt("thisisanale%20rt")</script>test  As an input. | Using unicode characters instead of the bare characters can bypass the filter on the input.  FINDING:XXX |
| **Mitigation** | | |
| **Filter out** | | |

## Lab06.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

## Lab07.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

## Lab08.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

## Lab09.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

## Lab10.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

# Lab04.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

# Lab05.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

# Lab06.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

# Lab08.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

# Lab09.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
|  |  | FINDING:XXX |
| **Mitigation** | | |
|  | | |

# Lab11.php

|  |  |  |
| --- | --- | --- |
| **Date** | **Attack** | **Result** |
| 19/01/21 | With this implementation, once the correct passwrod length is arrived at, with each letter in the password that matches the sytem password the processing time will increase thereby indicating when an attacker is getting closer to the system password. Max iterations is (password length) \* (number of characters) e.g. Alphanumeric set e.g. 8 \* 62 | Time increases while checking the password input against the password stored (once the characters in the password match) in accordance with the length of the input password  FINDING:Side channel Timing Attack |
| **Mitigation** | | |
| This can be mitigated by using a constant time algorithm to check the passwords against each other.  Change the PasswordValid function to:  function PasswordValid($TestForPassowrd, $Password) {  time\_nanosleep(1,0); //ns delay inserted to simulate processing not needed  return hash\_equals($TestForPassowrd, $Password);  } | | |