



security-assessment.com

Web Application Frameworks

Denis Andzakovic – OWASP Day 2012

- **My name is Denis Andžakovič**
 - Pen-Tester @ Security-Assessment.com
- **A sweeping generalization:**
 - Developers should strive to make my life as difficult as possible.

- **The Top Ten**
 - I am going to assume that we are familiar with this list.
- **The recurring theme from previous Web Sec talks has always been ‘Do not roll your own!’**

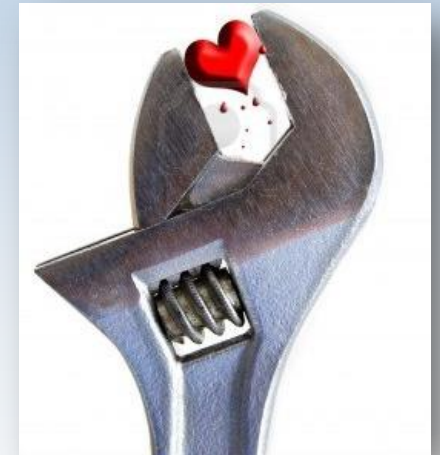
Don't roll your own!



security-assessment.com

■ Frameworks – <3

- They simplify the development process
- There's less code to write
- Code is easily re-used
- Code is robust, often heavily tested and integrated with the rest of your framework
- They make secure implementations easy (*cough*)
- Frameworks make it harder to make mistakes.



Frameworks and Pen-Testers

- **Makin' my life difficult.**
 - Secure, robust core code
 - Often meticulously reviewed and nit-picked
 - Security guidelines offered for the less sec-savvy developer
- **Also makin' my life rather simple :-D**
 - Easier recon
 - Readily available exploit code (on occasion....)
 - Implementation errors
 - Security misconfigurations



Example Framework 1

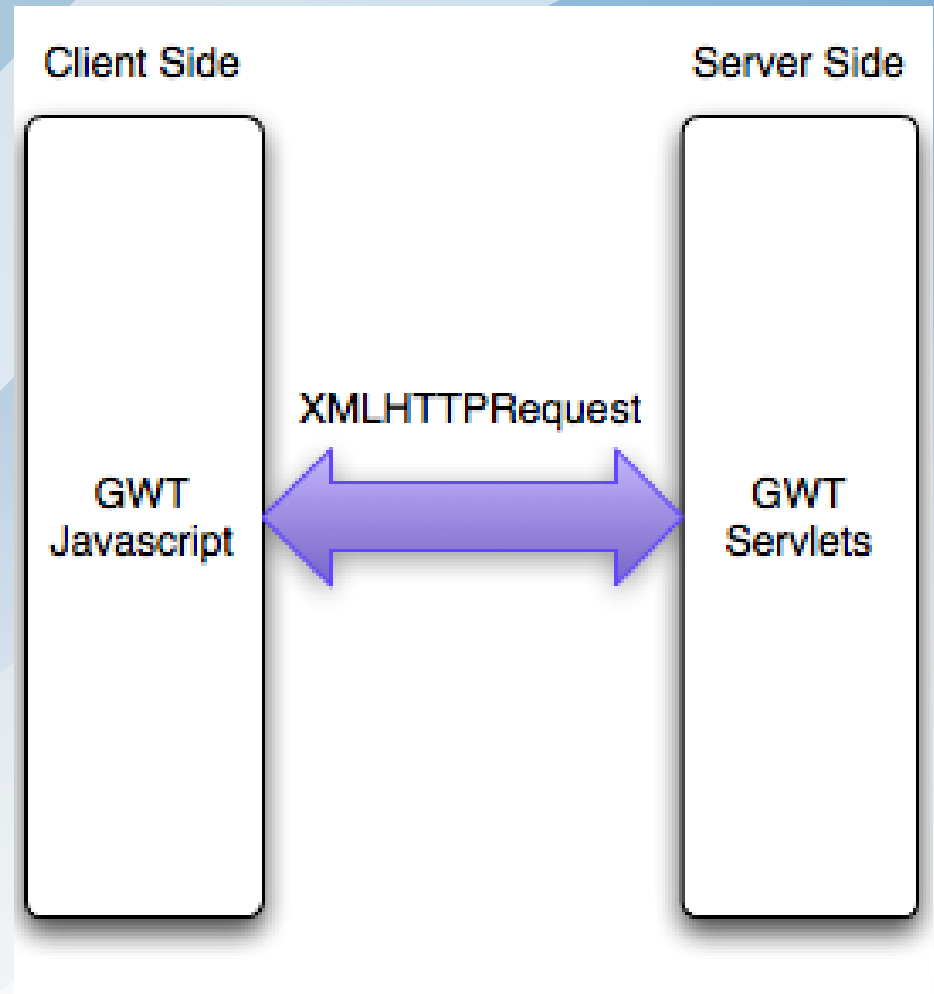
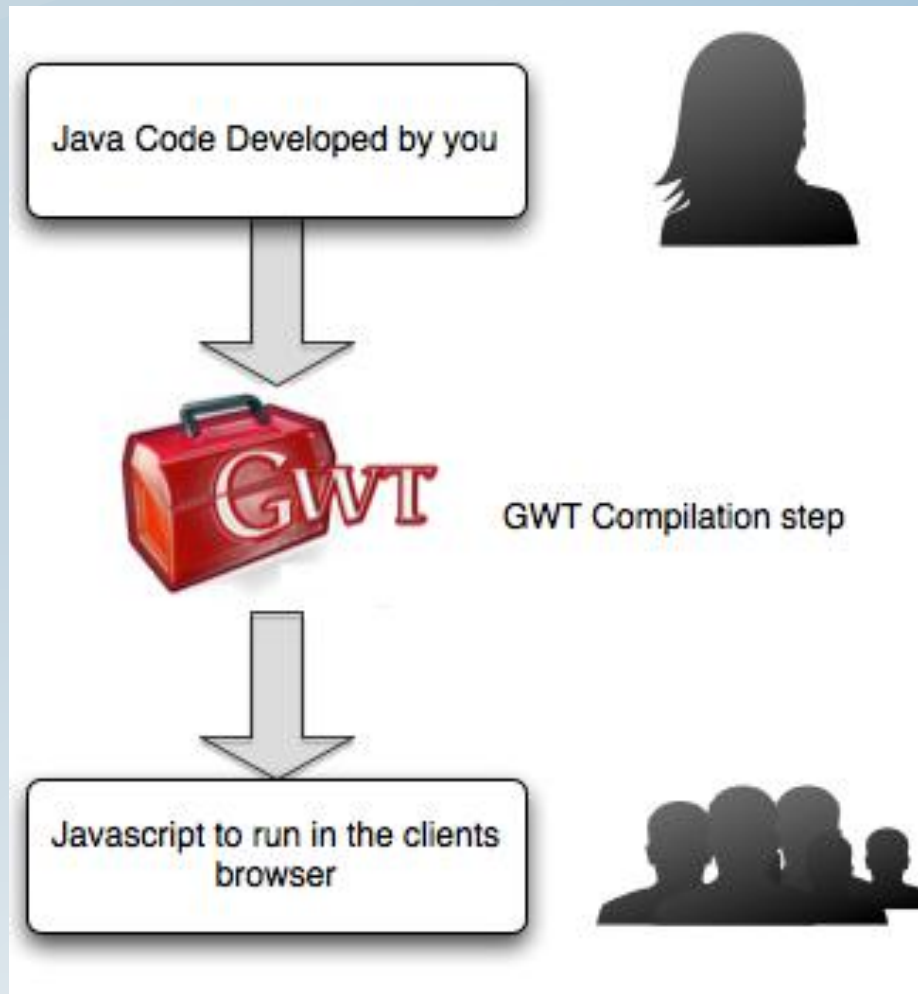
Google Web Toolkit

- Java based
- Compiles Java code into obfuscated JavaScript
- Provides a robust RPC implementation for server <-> client communication



How its strung together...

GWT - Overview



```
function lw(a){jw(a);if(a.j)
{a.b.I.style[PD]=SD;a.b.C!=-1&&iu(a.b,a.b.w,a.b.C);mt((zw(),Dw(null)),a.b)}else{a.d||nt((zw(),Dw(n
)]=BE}
function Se(){Se=nD;Re=new We;Pe=new Ze;Ke=new af;Le=new df;Qe=new gf;Oe=new kf;Me=new nf;Je=new q
{40:1},8,[Re,Pe,Ke,Le,Qe,Oe,Me,Je,Ne]})
function ni(a,b,c){if(!a){throw new bz}if(!c){throw new bz}if(b<0){throw new Ey}this.b=b;this.d=a;
vi(this,c);Nb(this.c,b)}else{this.c=null}}
function rx(a,b){var c,d,e;d</eoc.createElement(HE);c=
(e=$doc.createElement(IE),e['align']=a.b.b,so(e,'verticalAlign',a.c.b),e);Id(d,uw(c));Id(a.d,uw(d)
function Xd(a){if(a.ownerDocument.defaultView.getComputedStyle(a,BD).direction==ND){return (a.scro
((a.scrollWidth||0)-a.clientWidth)}return a.scrollLeft||0}
function Nq(a){var b,c,d,e;b=Yq(a);if(b<0){return fC(a.f,-(b+1))}c=Wq(a,b);if(c==null){return null
(dC(a.f,null),a.f.c),e=I(a.d,a,c),jC(a.f,d-1,e),H(a.d,a,e,c),e}
function ei(a){zc.call(this,'One or more exceptions caught, see full set in UmbrellaException#getC
null:Cj(a.Db(sj(Yn,{40:1,49:1},48,0,0)),49)[0]);this.b=a}
function fz(){fz=nD;ez=tj(Mn,{40:1},-1,
[48,49,50,51,52,53,54,55,56,57,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,11
,121,122])})}
```


Example RPC request

**7|0|7|http://127.0.0.1:8888/owasp_gwt_demo/|9DE0BA7FEFC
7237BEE17C6F7D23512E7|
com.example.owaspdemo.client.GreetingService|greetServ
er|java.lang.String/2004016611|
String1|String2|1|2|3|4|2|5|5|6|7|**

- This implementation helps ward off CSRF attacks and helps us defend against XSS attacks. Awesome.

Common Mistakes

- Unauthenticated access to RPC endpoints.
- UI feature and functionality restriction done on the client side.
- Additional Non-GWT functionality compromising XSS and CSRF protections





security-assessment.com

Unauthenticated access and client side UI restrictions

GWT DEMO

How to avoid this?



security-assessment.com

- **Understand how the specific framework operates (client side versus server side code)**
 - Ron Gutierrez has a very helpful talk titled 'Attacking Google Web Toolkit', which details some common ways to unlock client-side functionality.
- **Implement stringent access controls**
 - Validate, validate and validate some more.
- **Do not rely on Security-Through-Obscurity**
 - GDS have provided a set of tools for RPC endpoint enumeration and de-obfuscation of GWT code. (<http://blog.gdssecurity.com/labs/tag/gwt>)
- **Google's GWT Security Recommendations were followed**
 - <http://developers.google.com/> provide a very useful article titled 'Security for GWT Applications', which includes some easy-to-implement solutions for these issues.

To summarize...

- Client Side. Server Side. These are not the same thing!



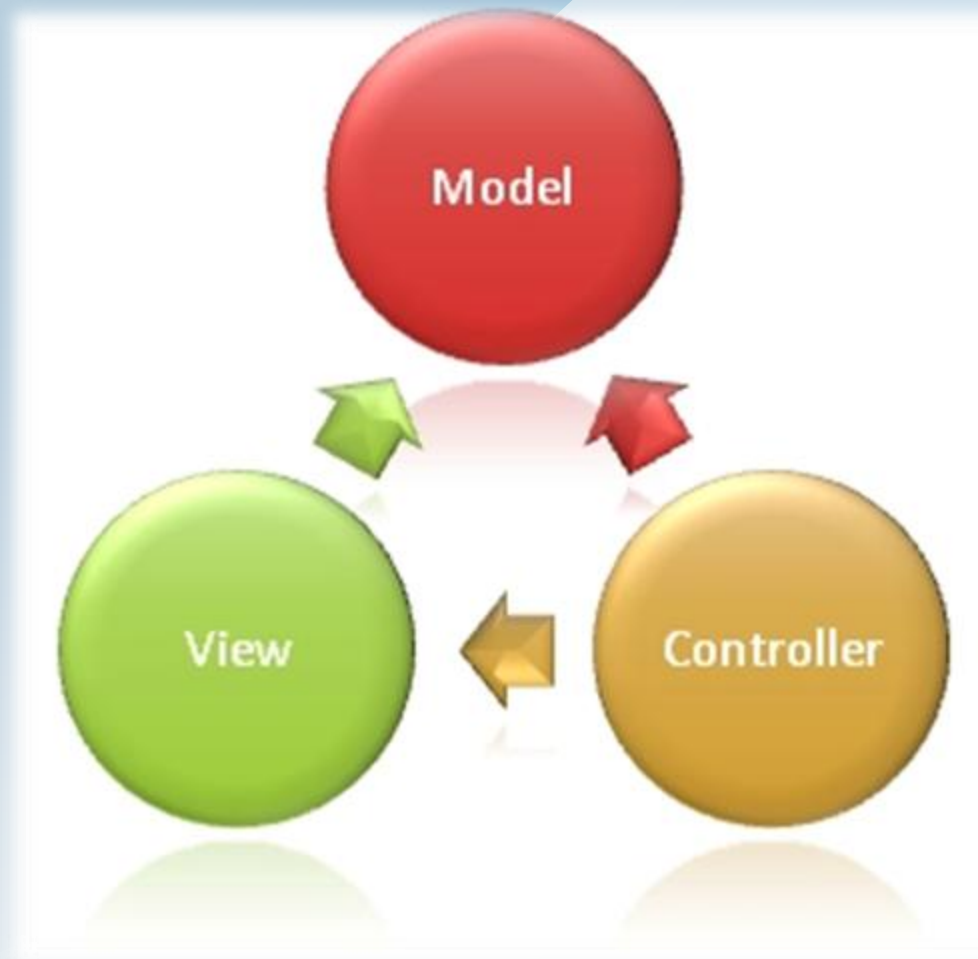
- Users are evil, never trust them. Validate all input.

Zend Framework

- *“A powerful high-quality open-source framework focused on developing modern Web Applications and Web Services”*
- Usually uses a MVC design with a dispatcher
- Without a Dispatcher, every implemented script must embed or implement authentication – Classic approach prone to human error
- Anti-Cross-Site-Scripting Escaping Magic disabled by default
 - This will change in version 2.0, According to Zend Framework project lead Matthew Weier O'Phinney



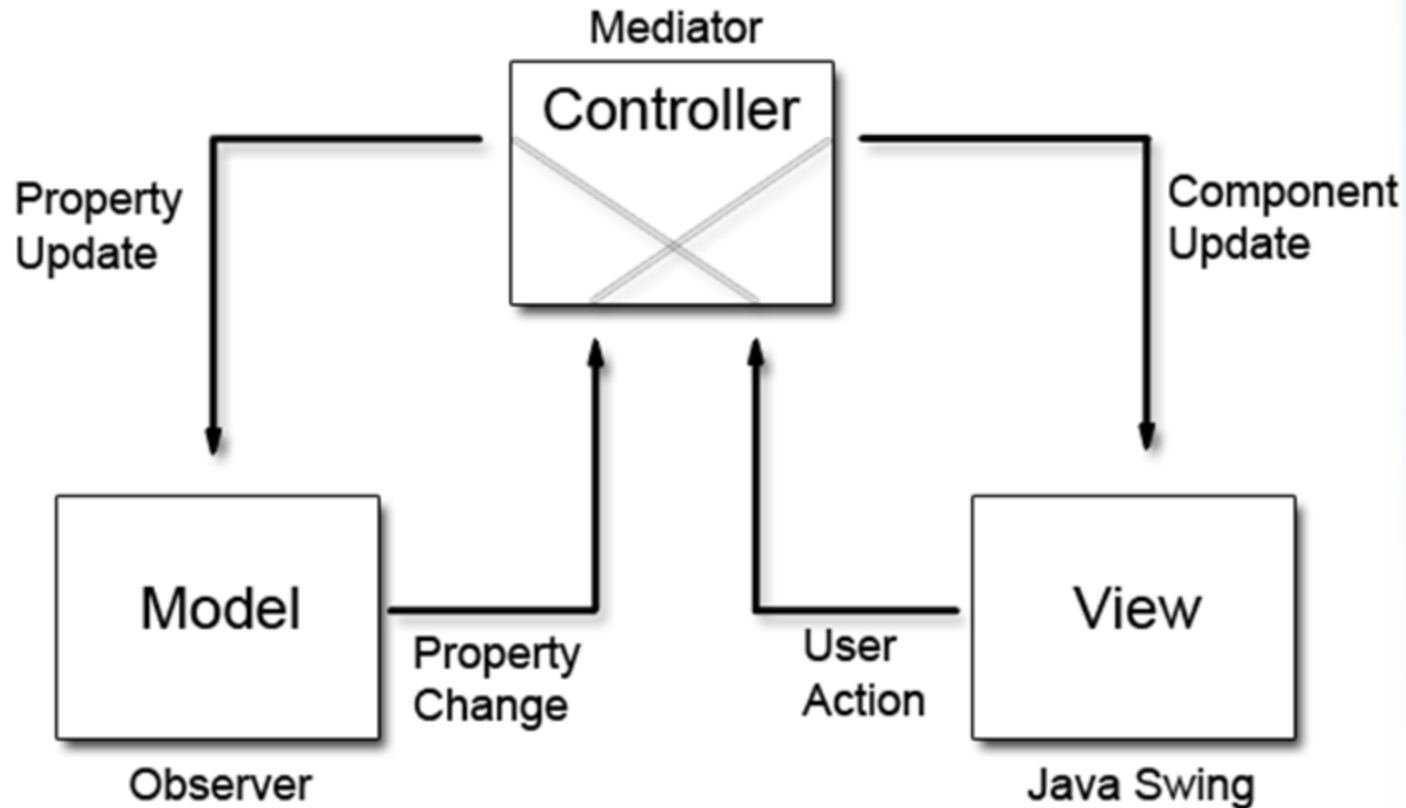
The Model View Controller



More on MVC



security-assessment.com



- **SQL Injection**
 - Zend offers several classes for DB access, yet for some reason no one uses them?
- **Cross Site Scripting issues**
 - Remember how Zend doesn't have auto anti-XSS magic enabled?
- **Framework specific vulnerabilities**
 - Specific versions of Zend are vulnerable to certain bugs in the core framework.
- **Practically the rest of the Top Ten as well...**
 - It's up to the developer to not do something ridiculous.

Who's been pwned?



security-assessment.com

- **XOOPS – Built on Zend... A quick look on exploit DB shows 68 Bugs...**
 - The majority of these are SQLi and XSS bugs...
- **Digitalus CMS – Also built on Zend...**
 - A brief search turned up an arbitrary file upload bug, wonderful.
- **Information disclosure bug in Zend itself**
 - Recently, a vulnerability was discovered in Zends XMLRPC package.

■ SQL Injection

Exploit:

`http://192.168.1.109/xoops-2.5.4/modules/system/admin.php?fct=users&selgroups=[Blind Sqli]`

```
} else {  
    $sql .= " AND l.gperm_groupid=" . $groupid . "";  
}  
$sql .= " AND b.isactive=" . $isactive;  
if (isset($side)) {  
    // get both sides in sidebox? (some themes need this)  
    if ($side == XOOPS_SIDEBLOCK_BOTH) {  
        $side = "(b.side=0 OR b.side=1)";  
    } elseif ($side == XOOPS_CENTERBLOCK_ALL) {  
        $side = "(b.side=3 OR b.side=4 OR b.side=5 OR b.side=7 OR b.side=8 OR b.side=9)";  
    } else {  
        $side = "b.side=" . $side;  
    }  
    $sql .= " AND " . $side;  
}  
if (isset($visible)) {  
    $sql .= " AND b.visible=$visible";  
}  
$sql .= " ORDER BY $orderby";  
$result = $db->query($sql);
```

- XSS – Our POC.

```
<form action='http://[host]/modules/pm/pmlite.php' method="post">
<input type="hidden" name="sendmod" value='1'>
<input type="hidden" name="to_userid" value=''><script>alert(document.cookie);</script>'>
<input type="submit" value="submit" id="btn">
</form>
```

- The culprit code.

```
$GLOBALS['xoopsTpl']->assign('to_username', XoopsUser::getUnameFromId($_POST["to_userid"]));
$pmform->addElement(new XoopsFormHidden('to_userid', $_POST["to_userid"]));
$subject = $myts->htmlSpecialChars($myts->stripSlashesGPC($_POST['subject']));
$message = $myts->htmlSpecialChars($myts->stripSlashesGPC($_POST['message']));
```

- **‘An attacker can exploit this vulnerability via browser by following this link:** `http://<vulnerable site>/scripts/fckeditor/editor/filemanager/connectors/test.html` ’
 - Hold on... FCKEditor?
- **3rd Party Features stuck onto the app... Great...**
 - Exploitable code, probably not even written by you, has gone and compromised the integrity of your entire application.

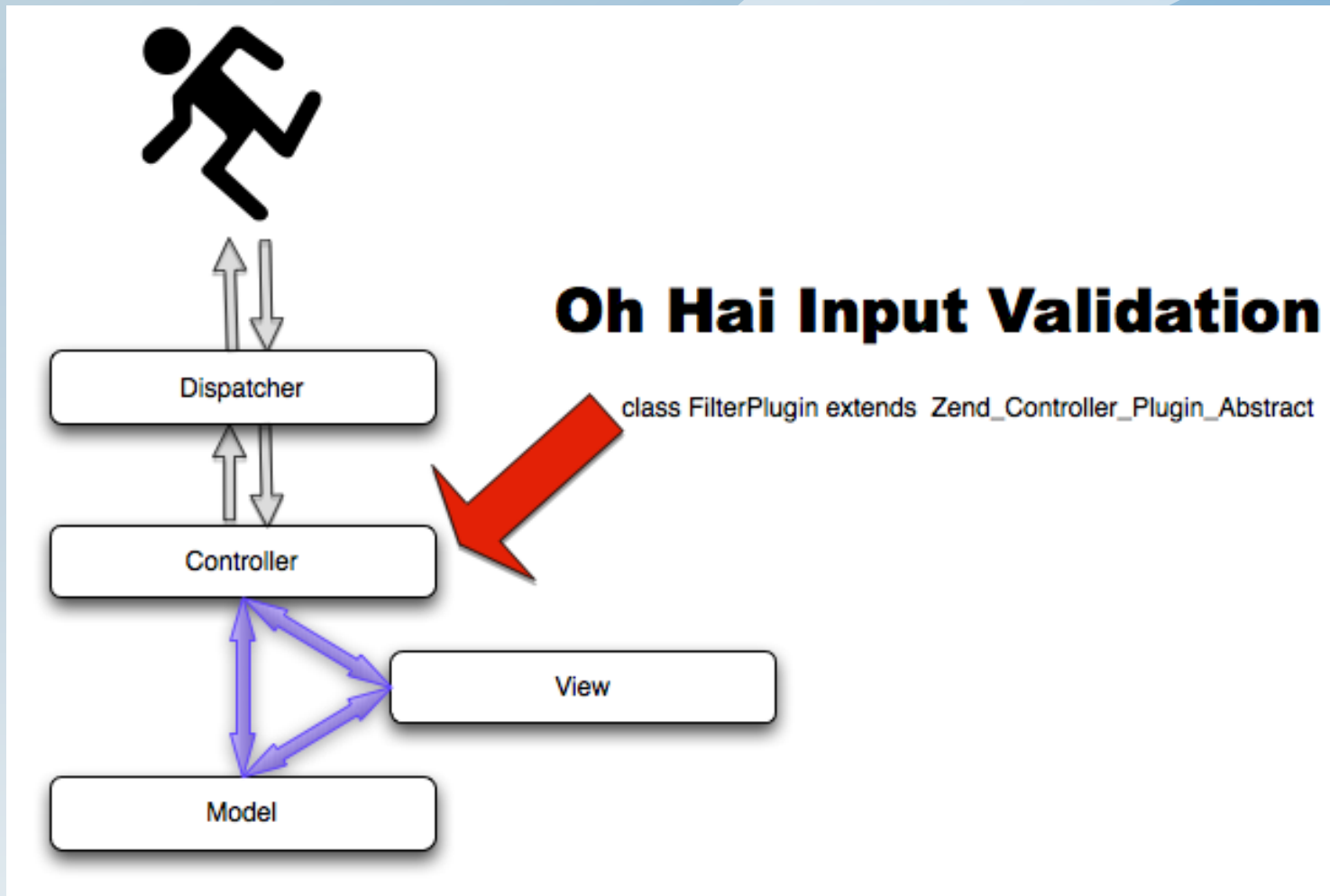
XXE Bug in Zend XMLRPC

```
<member>
  <name>faultString</name>
  <value>
    <string>Method "root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/dev/null
daemon:x:2:2:daemon:/sbin:/dev/null
adm:x:3:4:adm:/var/adm:/dev/null
lp:x:4:7:lp:/var/spool/lpd:/dev/null
sync:x:5:0:sync:/sbin:/dev/null
shutdown:x:6:0:shutdown:/sbin:/dev/null
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/dev/null
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/dev/null
operator:x:11:0:operator:/root:/dev/null
games:x:12:100:games:/usr/games:/dev/null
gopher:x:13:30:gopher:/var/gopher:/dev/null
ftp:x:14:50:FTP User:/var/ftp:/dev/null
nobody:x:99:99:Nobody:./:/dev/null
nscd:x:28:28:NSCD Daemon:./:/dev/null
vcsa:x:69:69:virtual console memory owner:/dev:/dev/null
ntp:x:38:38:./etc/ntp:/dev/null
dbus:x:81:81:System message bus:./:/dev/null
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/dev/null
haldaemon:x:68:68:HAL daemon:./:/dev/null
xfs:x:43:43:X Font Server:/etc/X11/fs:/dev/null
apache:x:48:48:Apache:/var/www:/sbin/nologin
pcap:x:77:77:./var/arpwatch:/sbin/nologin
" does not exist</string>
  </value>
</member>
</fault>
" does not exist</string>
bcsb:x:33:33:./etc/bcsb:/sbin/nologin
bcsb:x:48:48:bcsb:/sbin/nologin
```

What should have been done.

- **Zend comes with classes for database access and escaping.**
 - Zend_Db. Zend_Db_Statement. Zend_Db_Table ect
 - Zend_Db_Select exists to create dynamic SELECT queries, leverages prepared statements internally as often as possible
- **Ye-Oldie XSS scrubbing is not your friend.**
 - Leverage Zend_View_Helper
- **Centralize your validation**
 - Work with the controller
- **Zend provide a useful webinar detailing some common issues and ways to deal with them**
 - <http://static.zend.com/topics/Webinar-Zend-Secure-Application-Development-with-the-Zend-Framework.pdf>

More on Centralised validation



Microsoft .NET Framework

- .NET is basically one giant framework, this thing is huge.
- Many popular sites written in .NET
- First released in 2002
- Suffers the same issue as the previous frameworks...
Devs.



Frameworks built on frameworks



security-assessment.com

- **EG: DotNetNuke and Spring.Net**
- **Yet another layer for error**
 1. Errors with the core framework
 - Padding Oracle attack...
 2. Errors with the framework built on the core framework
 - DNN Arbitrary file upload bug...
 3. Top framework implementing core framework functions incorrectly
 - DNN-2011-9-C Authorization Bypass
 4. Developers implementing Framework itself incorrectly
 - This one is kind of self explanatory...

Vulns :D



security-assessment.com

Title	Date	
DotNetNuke Multiple Vulnerabilities	2012-07-13	
DotNetNuke Arbitrary File Upload Vulnerability	2012-03-09	
DotNetNuke Multiple Vulnerabilities	2012-02-03	
DotNetNuke Editor Script Insertion Vulnerability	2011-11-03	
DotNetNuke Module Permission Check Security Bypass Vulnerability	2011-08-26	
DotNetNuke Module Permission Check Security Bypass Vulnerability	2011-08-26	
DotNetNuke Security Bypass and File Upload Vulnerabilities	2011-07-06	
DotNetNuke Multiple Vulnerabilities	2010-12-07	
DotNetNuke Logging Provider Information Disclosure Weakness	2010-11-22	
DotNetNuke Syndication Handler Denial of Service Vulnerability	2010-08-19	
DotNetNuke Multiple Vulnerabilities	2010-06-18	
DotNetNuke Information Disclosure and Script Insertion	2010-05-20	
DotNetNuke System Messages Information Disclosure Weakness	2010-04-20	
DotNetNuke Cross-Site Scripting Vulnerability	2010-03-18	
DotNetNuke Role Expiration Privilege Escalation Security Issue	2010-02-18	
DotNetNuke Cross-Site Scripting and Information Disclosure	2009-11-27	
DotNetNuke Cross-Site Scripting and Script Insertion Vulnerabilities	2009-09-03	
DotNetNuke ErrorPage.aspx Cross-Site Scripting Vulnerability	2009-05-26	
DotNetNuke PayPal IPN Cross-Site Scripting Vulnerability	2009-04-13	
DotNetNuke Role Membership Security Bypass	2009-01-05	
DotNetNuke Multiple Vulnerabilities	2008-09-12	
DotNetNuke Multiple Vulnerabilities	2008-06-12	
DotNetNuke Cross-Site Scripting Vulnerabilities	2008-06-02	
DotNetNuke Cross-Site Scripting Vulnerability	2008-06-02	
DotNetNuke Multiple Vulnerabilities	2008-03-25	
DotNetNuke Multiple Vulnerabilities	2008-03-25	
DotNetNuke Multiple Vulnerabilities	2008-03-25	
DotNetNuke Multiple Vulnerabilities	2008-03-25	
DotNetNuke Multiple Vulnerabilities	2008-03-25	

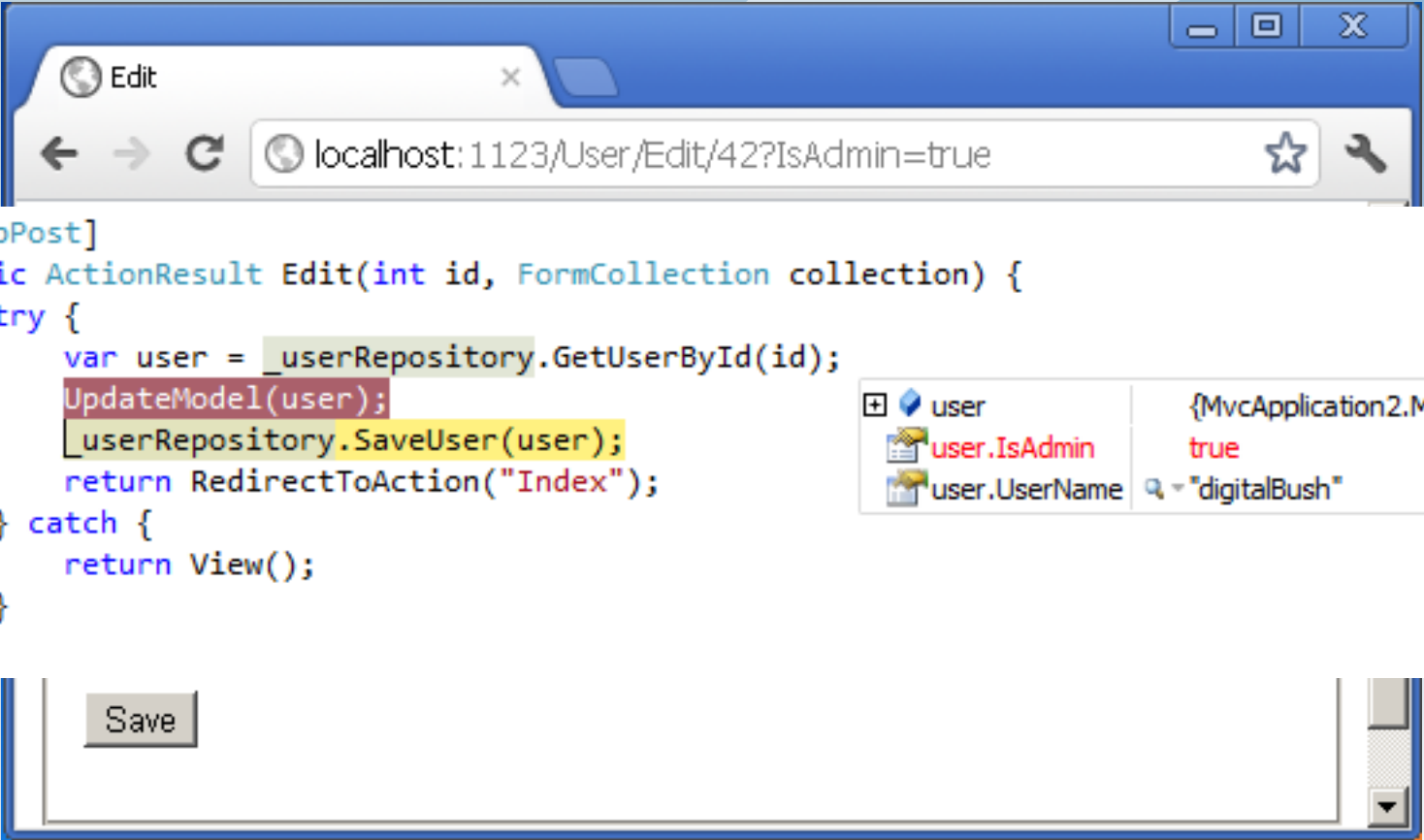
Doing it wrong.



security-assessment.com

- **As you probably know, GitHub was hacked by a miffed Russian gentleman in June...**
 - This was done via a mass assignment bug.
 - Yea, okay, technically that was ruby on rails, but the same concepts apply to .NET MVC.
- **Umbraco (a .NET based CMS) Remote command execution bug (another one from our friends at GDS)**

Mass Assignment



The screenshot shows a web browser window with the address bar displaying `localhost:1123/User/Edit/42?IsAdmin=true`. The page title is "Edit". A code overlay is positioned over the browser content, showing a C# `[HttpPost]` action method. The code attempts to update a user by ID, sets `user.IsAdmin` to `true`, and saves the user. A data table on the right side of the code overlay displays the state of the `user` object after the update.

```
[HttpPost]
public ActionResult Edit(int id, FormCollection collection) {
    try {
        var user = _userRepository.GetUserById(id);
        UpdateModel(user);
        _userRepository.SaveUser(user);
        return RedirectToAction("Index");
    } catch {
        return View();
    }
}
```

user	{MvcApplication2.Models.User}
user.IsAdmin	true
user.UserName	"digitalBush"

Save

- A specially crafted SOAP call results in unauthenticated file upload.
 - Because calls to this guy are not validated...

```
[WebMethod]
public string SaveDLRScript(string fileName, string oldName, string fileContents, bool ignoreDebu

    if (string.IsNullOrEmpty(fileName))
        throw new ArgumentNullException("fileName");

    StreamWriter SW;

    //As Files Can Be Stored In Sub Directories, So We Need To Get The Exeuction Directory Correct
    var lastOccurance = fileName.LastIndexOf('/') + 1;
    var directory = fileName.Substring(0, lastOccurance);
    var fileNameWithExt = fileName.Substring(lastOccurance);
    var tempFileName = IOHelper.MapPath(SystemDirectories.MacroScripts + "/" + directory + DateTi

        //SW = File.CreateText(tempFileName);
        SW = new StreamWriter(tempFileName, false, Encoding.UTF8);
        SW.Write(fileContents);
        SW.Close();
```


Doing it right.

- **Pay special attention to Model interactions**
 - What can a user change?
- **MSDN – use it**
 - Colossal amount of documentation, including a fair few helpful tips and tricks under ‘Writing Secure Code’
 - Following MSDNs ‘Web Service Security’ guidelines could have avoided the Umbraco issue.
 - Webcasts and Whitepapers on secure development offer a wealth of knowledge
 - A good starting point – Security in the .NET Framework
 - <http://msdn.microsoft.com/en-us/library/fkytk30f.aspx>

Vulnerabilities IN the framework

- **So you've written a web app based around a framework...**
 - The code has been peer reviewed
 - The application has been tested by a third party
 - Everything is happy days
- **Now keep an eye on the intertoobz.**
- **Vulnerabilities within the framework itself can compromise the integrity of your application**
- **Example: Zend XXE bug**

Misconfiguration

- Information disclosure is bad for you.
- While it might not be a vulnerability as such...
 - It shows the attacker where to swing the hammer...
 - Remember to lock down your production Implementations!
- Example: Don't forget to turn off debug...



Stack Trace:

```
[Exception: This is not good. Something bad happened.]
ErrorHandling.Controllers.HomeController.About() in D:\Dropbox\My Dropbox\dev\mvc3\ErrorHandling\ErrorHandling\Controllers\HomeController.cs:23
lambda_method(Closure , ControllerBase , Object[] ) +96
System.Web.Mvc.ActionMethodDispatcher.Execute(ControllerBase controller, Object[] parameters) +17
```

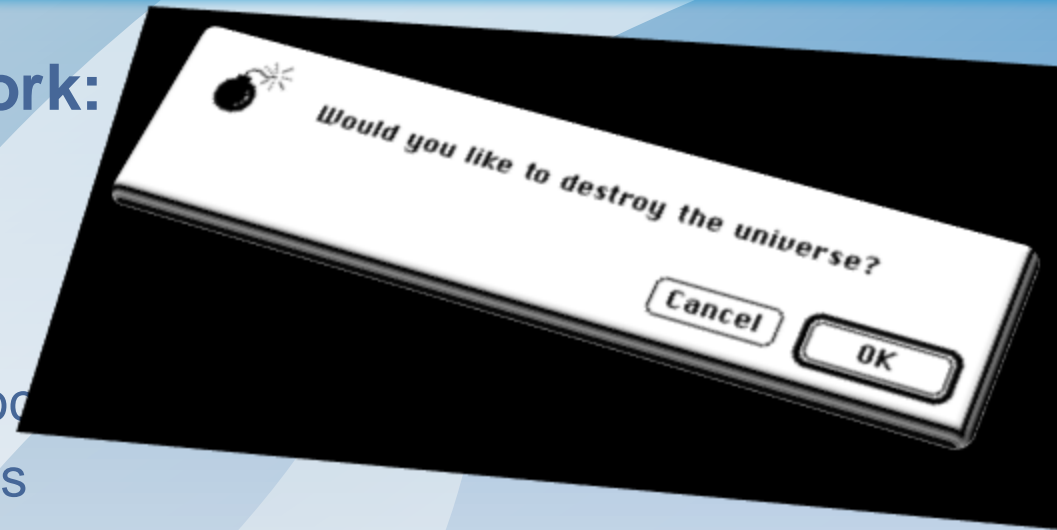
A quick recap – Dos and Don'ts

- **Do think things through, understand what your code and framework of choice is doing.**
- **Embrace your framework**
 - Use the available filtering and security routines where available. OWASP ESAPI is a good choice where said routines are not available, or a different framework entirely...
- **Implement secure coding practices**
- **Do -NOT- include 3rd party code and plugins**
 - Less code, less problems. It's as simple as that.
- **Have your code peer-reviewed**
- **Have your application pen-tested**

Try to avoid horrible software.

What to look for in a framework:

- Is it fit for purpose?
- Security Features
- Good documentation
 - Bonus points for brilliant docs
- Secure development guidelines
- If there were bugs released, how did the vendor respond?
 - Eg – Zend's prompt patching of the XXE bug.



Remain Vigilant and Be Pedantic

To design, deliver and operate a web application securely, it's key to:

- **Be pedantic about your implementations**
- **Double check all configs before going into prod**
 - Probably a good idea to remove README, INSTALL, LICENSE etc. as well...
- **Be vigilant when writing new code**
 - Think 'who could potentially mess with this' and go from there...
 - Kick your rookies until they understand.
- **Feed and Water – you have ops guys for a reason**
 - Keep things up to date.
- **Have a penetration test done by a reputable company**

- **Questions? Comments?**

- Denis Andzakovic – denis.andzakovic@security-assessment.com



Security-Assessment.com are looking for Pen-Testers. Got skills?
Give us a call.