Web Application Frameworks

Denis Andzakovic – OWASP Day 2012
# whoami

- **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!

- **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
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

Java Code Developed by you

GWT Compilation step

Javascript to run in the clients browser

Client Side

GWT Javascript

XMLHttpRequest

Server Side

GWT Servlets
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(null)),"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 qf(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.scrollHeight}((a.scrollHeight||0)-a.clientWidth))return a.scrollTop||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,c),e}
function ei(a){zc.call(this,'One or more exceptions caught, see full set in UmbrellaException#getCnt:0');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,[48,49,50,51,52,53,54,55,56,57,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,121,122])}
Example RPC request

7|0|7|http://127.0.0.1:8888/owasp_gwt_demo/|9DE0BA7FEFC7237BEE17C6F7D23512E7|
com.example.owaspdemo.client.GreetingService|greetServer|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
Unauthenticated access and client side UI restrictions

GWT DEMO
How to avoid this?

- 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](http://blog.gdssecurity.com/labs/tag/gwt))
- Google’s GWT Security Recommendations were followed
  - [http://developers.google.com/](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

![Diagram of the MVC architecture showing the relationships between Model, Observer, Controller, Mediator, View, and specific components like Property Change, User Action, Component Update, and Java Swing.]
Common bugs

- **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?

- **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:


```php
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);
```
X-Oops 2

- 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.
‘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
dae: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
shutd: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
oper: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
nsc: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
cap:x:77:77:/var/arpwatch:/sbin/nologin
" does not exist</string>
  </value>
</member>
```
What should have been done.

- **Zend comes with classes for database access and escaping.**
  - Zend_Db, Zend_Db_Statement, Zend_Db_Table, etc.
  - Zend_Db_Select exists to create dynamic SELECT queries, leverages prepared statements internally as often as possible.

- **Ye-Oldie XXS 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**
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

- 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...
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

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

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

```csharp
[WebMethod]
public string SaveDLRScript(string fileName, string oldName, string fileContents, bool ignoreDebug)
{
    if (string.IsNullOrEmpty(fileName))
        throw new ArgumentNullException("fileName");

    StreamWriter SW;

    // As files can be stored in sub directories, so we need to get the execution directory correctly.
    var lastOccurrence = fileName.LastIndexOf('/') + 1;
    var directory = fileName.Substring(0, lastOccurrence);
    var fileNameWithExt = fileName.Substring(lastOccurrence);
    var tempFileName = IOMapper.MapPath(SystemDirectories.MacroScripts + "/" + directory + DateTime.Now.ToString("yyyyMMddHHmmss") + "." + fileNameWithExt);

    // 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
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...
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 documentation
- Secure development guidelines
- If there were bugs released, how did the vendor respond?
  - Eg – Zend’s prompt patching of the XXE bug.
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**
Fin.

- Questions? Comments?
  - Denis Andzakovic – denis.andzakovic@security-assessment.com