

CAN HAS LOGIN?

Presentation by Wesley Mason (aka 1stvamp)

Insecurities in Web Authentication, Authorisation
and OpenID



Who the hell..?

Who the hell..?

- Wesley Mason

Who the hell..?

- Wesley Mason
- aka Wes

Who the hell..?

- Wesley Mason
- aka Wes
- aka 1stvamp

Who the hell..?

- Wesley Mason
- aka Wes
- aka 1stvamp
- aka "Oi You" to any chavs

Who the hell..?

Who the hell..?

- Web Developer

Who the hell..?

- Web Developer
- Developing web apps for ~10 years

Who the hell..?

- Web Developer
- Developing web apps for ~10 years
- From 'UII (Hull to everyone else)

Who the hell..?

- Web Developer
- Developing web apps for ~10 years
- From 'UII (Hull to everyone else)
- Yes I really am this short..

The Basics

The Basics

- Web Applications

The Basics

- Web Applications
- GET/POST, Forms and Sessions

The Basics

- Web Applications
- GET/POST, Forms and Sessions
- HTTP Auth - Basic and Digest

The Basics

- Web Applications
- GET/POST, Forms and Sessions
- HTTP Auth - Basic and Digest
- HTTPS/SSL

Meant by "Web
Application"..

Meant by "Web Application" ..

- Applications that use web pages as an interface
- Services that push data out over HTTP in a web format ([X]HTML / RSS / Atom / other XML / JSON / XLink etc.)
- Javascript applications that run within userspace within a web browser (sometimes including plug-ins, e.g. Flash, Java or **<snicker>** Silverlight)

Not meant..

Not meant..

- Userspace applications which push any old data out over HTTP
- Web servers (HTTPDs)
- Libraries
- Applets (Java, **<shudder>** ActiveX etc.)
forced to run within a browser but perform no HTTP transactions back to a server (see Desktop Application)

HTTP

HTTP

- GET & POST
- Headers
- [X]HTML Forms
- Sessions..

Sessions – Faking it

Sessions – Faking it

- HTTP is stateless
- Web applications need to be aware of state from moment to the next
- Unique ID known by client and application
- Application keeps data, client knows ID
- ID passed by GET/POST, or by Cookie

HTTP Auth, and HTTPS

HTTP Auth, and HTTPS

- Basic Auth - Via headers, Base-64 encoded
- Digest Auth - Similar to Basic, but uses one way MD5 hashing and a "nonce"
- HTTPS - HTTP over SSL

Common Mistakes & Exploits

Common Mistakes & Exploits

- Access Control

Access Control

Access Control

- Access Control Lists (ACL)

Access Control

- Access Control Lists (ACL)
- Objects – Doing on

Access Control

- Access Control Lists (ACL)
- Objects – Doing on
- Subjects – Who's doing it

Access Control

- Access Control Lists (ACL)
- Objects – Doing on
- Subjects – Who's doing it
- Permissions – What they can do

Access Control

- Access Control Lists (ACL)
- Objects – Doing on
- Subjects – Who's doing it
- Permissions – What they can do
- Whitelists and Blacklists

Access Control

- Access Control Lists (ACL)
- Objects – Doing on
- Subjects – Who's doing it
- Permissions – What they can do
- Whitelists and Blacklists
- This is "business logic" – even if it's dynamic

Common Mistakes & Exploits

- Access Control

Common Mistakes & Exploits

- Access Control
- Exposed Secrets

Exposed Secrets

Exposed Secrets

- Defaulting variables in userspace (e.g. parameters, cookies, embedded in Javascript)

Exposed Secrets

- Defaulting variables in userspace (e.g. parameters, cookies, embedded in Javascript)
- Including user IDs, passkeys etc. in userspace

Exposed Secrets

- Defaulting variables in userspace (e.g. parameters, cookies, embedded in Javascript)
- Including user IDs, passkeys etc. in userspace
- Providing links to functionality to users who don't have permission to use them - if you know it exists, you can start looking at how to get in

Exposed Secrets

- Defaulting variables in userspace (e.g. parameters, cookies, embedded in Javascript)
- Including user IDs, passkeys etc. in userspace
- Providing links to functionality to users who don't have permission to use them - if you know it exists, you can start looking at how to get in
- Providing application errors, debugging info and application logic in user errors

Common Mistakes & Exploits

- Access Control
- Exposed Secrets

Common Mistakes & Exploits

- Access Control
- Exposed Secrets
- SQL Injection, Cross Site Scripting & Forgery

SQL Injection, Cross Site Scripting & Forgery

SQL Injection, Cross Site Scripting & Forgery

- SQL Injection - Lack of validation / stripping allows executing SQL into database backend

SQL Injection, Cross Site Scripting & Forgery

- SQL Injection - Lack of validation / stripping allows executing SQL into database backend
- Cross Site Script (XSS)

SQL Injection, Cross Site Scripting & Forgery

- SQL Injection - Lack of validation / stripping allows executing SQL into database backend
- Cross Site Script (XSS)
 - Type 0 - DOM

SQL Injection, Cross Site Scripting & Forgery

- SQL Injection - Lack of validation / stripping allows executing SQL into database backend
- Cross Site Script (XSS)
 - Type 0 - DOM
 - Type 1 - Non-persistent

SQL Injection, Cross Site Scripting & Forgery

- SQL Injection - Lack of validation / stripping allows executing SQL into database backend
- Cross Site Script (XSS)
 - Type 0 - DOM
 - Type 1 - Non-persistent
 - Type 2 - Persistent

SQL Injection, Cross Site Scripting & Forgery

- SQL Injection - Lack of validation / stripping allows executing SQL into database backend
- Cross Site Script (XSS)
 - Type 0 - DOM
 - Type 1 - Non-persistent
 - Type 2 - Persistent
- Cross Site Request Forgery (CSRF) - Asking users to follow URLs with GET actions

SQL Injection, Cross Site Scripting & Forgery

- SQL Injection - Lack of validation / stripping allows executing SQL into database backend
- Cross Site Script (XSS)
 - Type 0 - DOM
 - Type 1 - Non-persistent
 - Type 2 - Persistent
- Cross Site Request Forgery (CSRF) - Asking users to follow URLs with GET actions
- Cross Zone Scripting - Browser

Common Mistakes & Exploits

- Access Control
- Exposed Secrets
- SQL Injection / Cross Site Scripting & Forgery

Common Mistakes & Exploits

- Access Control
- Exposed Secrets
- SQL Injection / Cross Site Scripting & Forgery
- Session Hijacking and Fixation

Sessions & Cookies

Sessions & Cookies

- Hijacking

Sessions & Cookies

- Hijacking
 - 1. Identify session ID

Sessions & Cookies

- Hijacking
 - 1. Identify session ID
 - 2. Trick user into giving you session ID

Sessions & Cookies

- Hijacking
 - 1. Identify session ID
 - 2. Trick user into giving you session ID
 - 3. ???

Sessions & Cookies

- Hijacking
 - 1. Identify session ID
 - 2. Trick user into giving you session ID
 - 3. ???
 - 4. Profit

Sessions & Cookies

- Hijacking
 - 1. Identify session ID
 - 2. Trick user into giving you session ID
 - 3. ???
 - 4. Profit
- Harvesting details – what data does the user have access to in their session?

Sessions & Cookies

- Hijacking
 - 1. Identify session ID
 - 2. Trick user into giving you session ID
 - 3. ???
 - 4. Profit
- Harvesting details – what data does the user have access to in their session?
- Cookies – More than one way to skin a session?

Common Mistakes & Exploits

- Access Control
- Exposed Secrets
- SQL Injection / Cross Site Scripting & Forgery
- Session Hijacking and Harvesting

Common Mistakes & Exploits

- Access Control
- Exposed Secrets
- SQL Injection / Cross Site Scripting & Forgery
- Session Hijacking and Harvesting
- All the rest - Ajax/Javascript, forgotten details retrieval, and Encryption use

All the rest..

All the rest..

- Ajax/Javascript/DHTML

All the rest..

- Ajax/Javascript/DHTML
 - Does it add value?

All the rest..

- Ajax/Javascript/DHTML
 - Does it add value?
 - Is there a non-Javascript version?

All the rest..

- Ajax/Javascript/DHTML
 - Does it add value?
 - Is there a non-Javascript version?
 - Is data being set inside userspace code?

All the rest..

- Ajax/Javascript/DHTML
 - Does it add value?
 - Is there a non-Javascript version?
 - Is data being set inside userspace code?
 - Json, eval(), requests, Firebug

All the rest..

- Ajax/Javascript/DHTML
 - Does it add value?
 - Is there a non-Javascript version?
 - Is data being set inside userspace code?
 - Json, eval(), requests, Firebug
- Web Frameworks

All the rest..

- Ajax/Javascript/DHTML
 - Does it add value?
 - Is there a non-Javascript version?
 - Is data being set inside userspace code?
 - Json, eval(), requests, Firebug
- Web Frameworks
 - A vulnerability in a framework is a vulnerability in your application

All the rest.. (cont.)

All the rest.. (cont.)

- Retrieval of forgotten details (e.g. passwords)

All the rest.. (cont.)

- Retrieval of forgotten details (e.g. passwords)
 - Adding a "noce" token - usually time based, with hashing

All the rest.. (cont.)

- Retrieval of forgotten details (e.g. passwords)
 - Adding a "noce" token - usually time based, with hashing
- Encryption use - not a holy grail

All the rest.. (cont.)

- Retrieval of forgotten details (e.g. passwords)
 - Adding a "noce" token - usually time based, with hashing
- Encryption use - not a holy grail
 - Used (often in Enterprise) in place of actual well thought out application logic

All the rest.. (cont.)

- Retrieval of forgotten details (e.g. passwords)
 - Adding a "noce" token - usually time based, with hashing
- Encryption use - not a holy grail
 - Used (often in Enterprise) in place of actual well thought out application logic
 - Good for protecting data - but only with correct auth

All the rest.. (cont.)

- Retrieval of forgotten details (e.g. passwords)
 - Adding a "noce" token - usually time based, with hashing
- Encryption use - not a holy grail
 - Used (often in Enterprise) in place of actual well thought out application logic
 - Good for protecting data - but only with correct auth
 - Otherwise it's just Security by Obscurity





- What it solves



- What it solves
 - Many services, many accounts



- What it solves
 - Many services, many accounts
 - Many accounts == lots of replication



- What it solves
 - Many services, many accounts
 - Many accounts == lots of replication
- What it doesn't solve



- What it solves
 - Many services, many accounts
 - Many accounts == lots of replication
- What it doesn't solve
 - Permissions and trust - that's up to you



- What it solves
 - Many services, many accounts
 - Many accounts == lots of replication
- What it doesn't solve
 - Permissions and trust - that's up to you
 - Spam (see above)



- What it solves
 - Many services, many accounts
 - Many accounts == lots of replication
- What it doesn't solve
 - Permissions and trust - that's up to you
 - Spam (see above)
- Misuse & Exploits



OpenID Misuse & Exploits

OpenID Misuse & Exploits

- MD5 hashing is done at the current login site
 - you're trusting them with your login details, before it is sent to your OpenID provider
- The current site can request any details from your OpenID provider, if present, e.g. address details - no per details or site permissions ala LDAP
- Some sites use it to prevent spam and malicious use - however don't provide trusted lists of providers, so you can use your own and bypass any required CAPTCHA or other auth, unless the site uses their own

The logo consists of a white circular arrow pointing right, with an orange vertical bar on its left side.

OpenID Misuse & Exploits (cont.)

OpenID Misuse & Exploits (cont.)

- Code Injections, Forgeries and Spoofing - can hijack an OpenID login just as easily as any other
- Vulnerable to Passive Backdoor Injection - when you login to an OpenID using site, you are telling the web application to connect to an arbitrary third party server, push some POST data and download a response.
 - The response can contain anything - including an injection attack: an OpenID client needs to be as vigilant about parsing any attempted SQL or command injections as if accepting a weblog comment post from a user!





OpenID Extended

- Extending the OpenID specification at the client and server ends, to use a Javascript/Ajax client with client-side hashing

Extending the OpenID spec

Extending the OpenID spec

- Javascript MD5 hashing of tokens at the client side
 - Hashing just login details with login site passing hashed details directory to your OpenID provider
 - OpenID provider could then provide a hashed token based on login details and a shared secret, which the client can verify
 - Off-site login, client open a connection (via new window/tab in browser) to OpenID provider with token from login site, who after login provides verification token to login site via separate connection



OpenID Extended

- Extending the OpenID specification at the client and server ends, to use a Javascript/Ajax client with client-side hashing



OpenID Extended

- Extending the OpenID specification at the client and server ends, to use a Javascript/Ajax client with client-side hashing
- Uses



OpenID Extended

- Extending the OpenID specification at the client and server ends, to use a Javascript/Ajax client with client-side hashing
- Uses
 - Stronger protection against spoofing, as well as protection against providing your details to sites with **absolute** trust



OpenID Extended

- Extending the OpenID specification at the client and server ends, to use a Javascript/Ajax client with client-side hashing
- Uses
 - Stronger protection against spoofing, as well as protection against providing your details to sites with **absolute** trust
 - Can be used in desktop clients to interact with OpenID auth'd web services



OpenID Extended

- Extending the OpenID specification at the client and server ends, to use a Javascript/Ajax client with client-side hashing
- Uses



OpenID Extended

- Extending the OpenID specification at the client and server ends, to use a Javascript/Ajax client with client-side hashing
- Uses
- Caveats



OpenID
Extended
Caveats



OpenID

Extended

Caveats

- Requires trust of the Javascript client code provided by a site



OpenID Extended

Caveats

- Requires trust of the Javascript client code provided by a site
- Can be mitigated by using one trusted external source for this library, leaving user to verify this source is being imported



OpenID Extended

Caveats

- Requires trust of the Javascript client code provided by a site
- Can be mitigated by using one trusted external source for this library, leaving user to verify this source is being imported
 - Or by keeping a trusted copy of the library in userspace, e.g. using Greasemonkey



OpenID
Extended
Caveats (cont.)



Extended

Caveats (cont.)

- Still vulnerable to phishing/spoofing attempts by sites providing modified versions of the library, or fake OpenID login sites

kkthxbye

kkthxbye

- Sources: Wikipedia, Google, phplarchitect, 2600, "Art of Intrusion" – see Adrian Lamo, lots of experience
- Homework: telnet, wget/curl, tcpdump, Nessus, Nitko (and Jitko)
- <http://1stvamp.org/brumcon> – slides, notes and code