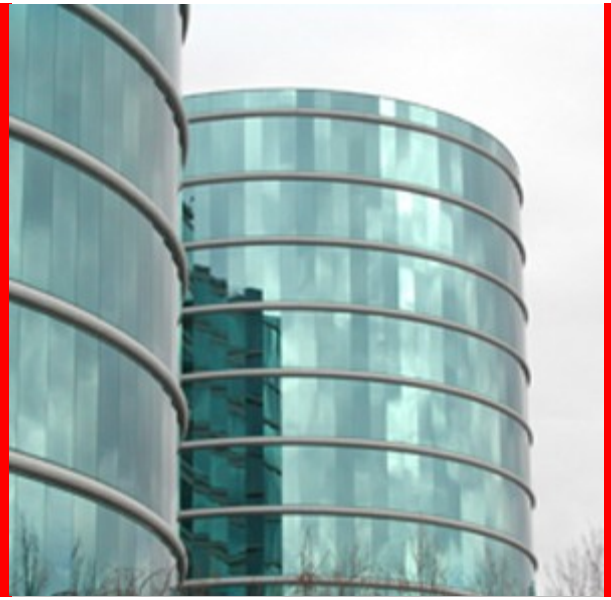

ORACLE®



ORACLE®

Security of Community Developed Wiki Plug-ins

Andy Webber
Ethical Hacker, Global Product Security

Who has:

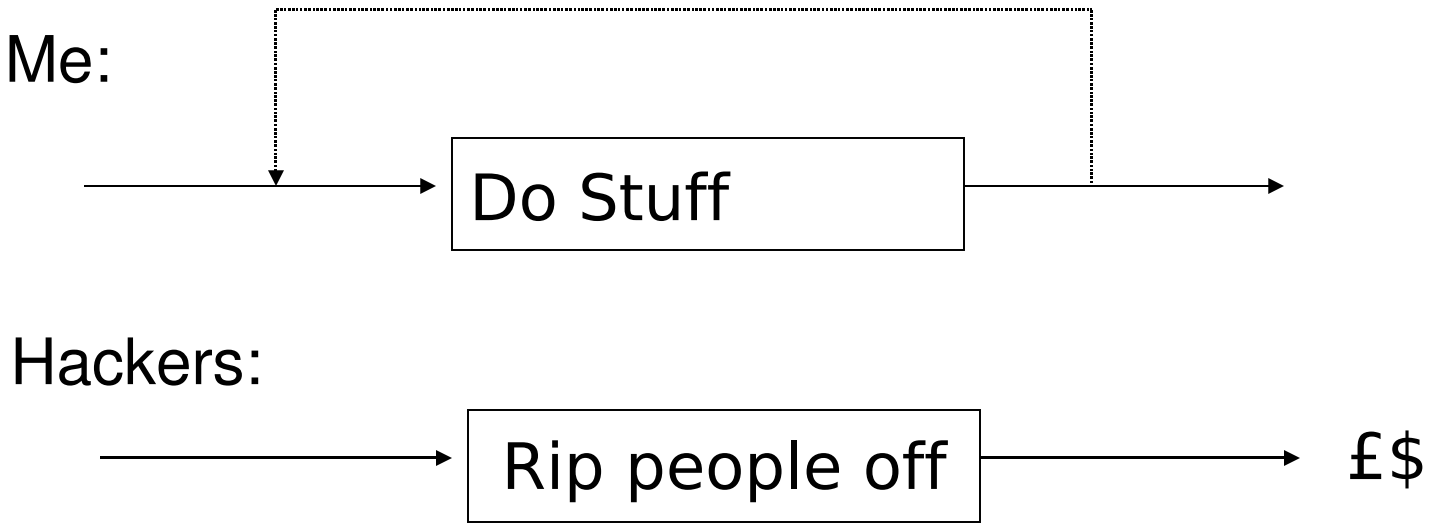
- Used a wiki
- Installed/managed a wiki
- Written a wiki, a plug-in or an extension
- Looked for security bugs in a wiki
- Found security bugs in a wiki
- Fixed security bugs in a wiki

What is Security?

- Need to login
- Access controls
- Accountability/Audit
- Code quality/”lack of [security] bugs”
- Regulatory compliance

Time for a Paradigm Shift

- Marketing:
  - A new ontology for cohesive collaboration in an encyclopaedic concept-driven ecosystem space.



Security

Not every user is your friend

- Security is not absolute
- Wiki software (with plug-ins/extensions) is entrusted with other people's data
  - Value of the data is not known by the developer
  - Can not assess "reasonable" protection
- Cost of security/insecurity
  - Software engineering – repeat what works; learn from mistakes

Plug-ins/Extensions

Not every plug-in developer is a Software Engineer

- Plug-ins are to a Wiki what cgi is to a Web Server
- Plug-ins are [usually] easy to write – even by novices
- Plug-ins extend Wikis into generic web application/content frameworks
- No matter how well designed/coded the Wiki, plug-ins can undermine it
- Plug-in insecurity often compromises the whole Wiki

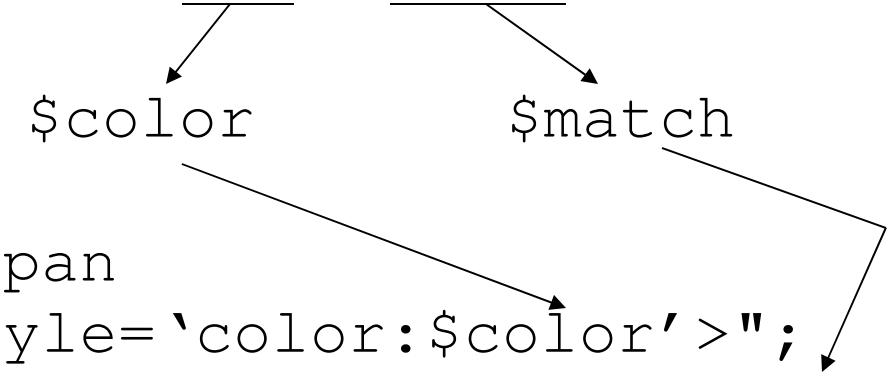
Injection

Not every Software Engineer is a Security Specialist

- DokuWiki “color” plug-in

Regex

```
<color.*?>(?=.*?</color>)
```




```
$renderer->doc.="<span
                 style='color:$color'>";
$renderer->doc.=htmlspecialchars($match);
$renderer->doc.="</span>";
```

```
Wiki: <color red>Redtext</color>
HTML: <span style='color:red'>Redtext</span>
```

Injection Attack

html intrinsic events

```
Wiki
<color red‘ onmouseover=
‘alert(document.cookie)’ alt=‘>XSS</color>
```

```
HTML
<span style=‘color:red‘ onmouseover=
‘alert(document.cookie)’ alt=‘’>XSS</span>
```

Injection

DokuWiki is not alone

- A MediaWiki YouTube extension

```
$wgParser->setHook('youtube',
                   'renderYouTube');
...
function renderYouTube($input, $output) {...
  $output='<object align="'.$argv['align'].'"
                          width="'.
$argv['width'].'"
              ...' ;
return $output; }
```

```
Wiki: <youtube align="&quot onmouseover=...>
HTML: <object align="" onmouseover=...>
```

Injection – Analysis

Keep data as data, code as code and don't mix them up

- Use an API that treats data as data
  - *Eg* bind variables/prepared statement for SQL
- Encode special characters in the output grammar
  - Use the API provided rather than roll your own
- Filter input to avoid special characters in output grammar

Injection – Analysis

Not just HTML/XSS

- Do not trust any input from outside the system – even from databases, LDAP, SOAP etc (2nd order injection)
- SQL injection
- Log file injection
  - *Eg* if taking browser's user agent string, referer, …
- XML/SOAP injection
  - Attack may contain an XML injection to affect SOAP query
  - SOAP response may contain data that needs encoding before use (*eg* a URL returned from SOAP needs encoding before use in …)
- File path injection (path traversal?)
  - Platform dependant filtering
- Shell injection

Wiki vendor actions

What wiki vendors should do

- Have a contact point for reporting security issues – including 3rd party plug-ins/extensions
  - Register it with OSVDB.org
- Define APIs to make it easy to “do it right”
  - MediaWiki has one for xHTML rendering
- Include information about “doing it right” in “how to write a plug-in/extension”
- Consider QA process for plug-ins/extensions
  - *cf* Mozilla Firefox extensions
- When an issue is reported, also examine other similar plug-ins and other plug-ins by the same author.