Vulnerability & Attack Injection for Web Applications

José Fonseca
Marco Vieira
Henrique Madeira

University of Coimbra, Portugal

DSN, Estoril, Portugal, 30/06/2009
Presentation Outline

- Research problem
- Vulnerability Injection
- Attack Injection
- Experiments
- Conclusions
Research Problem

• Are Web Applications secure?
  • 70% of websites at immediate risk of being hacked – Acunetix, 2007
  • 6 vulnerabilities per website – WhiteHat, 2008
  • 55% of the 40,000 vulnerability disclosures are from the web – IBM X-Force, 2008
  • 70 Air Traffic Control web applications have 3,857 vulnerabilities - U.S. Department of Transportation, 2009
  • Sophos software discovers one new infected webpage in 4,5 seconds – 24 hours a day, 365 days a year – Sophos, 2009
  • ...

No, they are not!
Research Problem

• Does it matter?

  • Ad-hoc web applications are highly-profitable and inexpensive target for attackers – IBM X-Force, 2008
  
  • Organized crime responsible for 90% of the 285 million records compromised in 2008 – Verizon
  
  • SQL Injection with 40% and XSS with 28% are the predominant type of attacks – IBM X-Force, 2008
  
  • In less than 20 hours, XSS Samy Worm propagated to over one million users of the MySpace in 2005
  
  • 225 million records compromised in 2008 due to SQLi – Verizon
  
  • The average cost per record is 200 dollars - Ponemon Institute, 2009
  
  • ...

  Yes, it does!
Research Problem

- What can be done?
  - IDS
  - Firewalls
  - Static Code Analyzers
  - Web Application Vulnerability Scanners
  - Security Training
  - ...

- 19% of the organizations that suffered from a data breach attack were PCI compliant – *Verizon, 2009*

**How to assess these security mechanisms in custom deployment scenarios?**
Our Proposal

“Attack is the secret of defence”
Sun Tzu’s Art of War, 500 BCE

Vulnerability & Attack Injection for Web Applications
Field Study on Vulnerabilities

- 6 LAMP web applications
- 655 security vulnerabilities (XSS and SQLi)
Field Study on Vulnerabilities

Missing function call extended – Vulnerabilities caused by a variable that should have been cleaned by a specific function (76% of all the faults found)
Vulnerability Operators

- **Location pattern** – how to locate where to inject a vulnerability

- **Vulnerability code change** – what to change in order to inject a vulnerability
Vulnerability Injection Methodology

1. Analysis of the source code of the web application
2. Search for the locations where a vulnerability may exist
3. Mutation of the code to inject a vulnerability
Vulnerability Injection Example

```php
$blog = intval($_GET['blog']);
...
$sql_text = "delete blogs where author_id = ".$author." and blog_id = ".$blog;
...
$result = mysql_query($sql_text,$conn);
... 
```
**Vulnerability Injection Example**

Vulnerability code change

```php
$blog = intval($_GET['blog']);
...
$sql_text = "delete blogs where author_id = ".$author." and blog_id = ".$blog;
```

Vulnerable SQL query string concatenation

```php
$sql_text = "delete blogs where author_id = ".$author." and blog_id = ".$blog;
```

**Location pattern**

http://[site]/blogs.php?blog=23 or 1=1
Vulnerability Injection

• Some possible scenarios:
  • Train security teams
  • Evaluate security teams
  • Estimation of vulnerabilities in the code
  • Attack Injector
  • …
Attack Injection Methodology

- Methodology to test security mechanisms:
  1. Injection of realistic vulnerabilities
  2. Controlled attack of the vulnerabilities

- Stages:
  1. Preparation
  2. Injection of Vulnerabilities
  3. Attack
Attack Injection: 
1-Preparation

- Deploy HTTP and DB proxies
- Crawl the Web Application
- Monitor HTTP and DB communications
- Synchronize the captured data (web pages, variables and SQL queries) with the matching input interaction
Attack Injection: 2-Vuln. Injection

- Check each possible vulnerable location
- Create copies with one injected vulnerability
- Generate the attack payloads and their matching footprints
**Attack Injection: 3-Attack**

- Deploy HTTP and DB proxies
- Restore database and web application files
- Inject one vulnerability
- Attack the vulnerability using the attack payloads
- Calculate attack success
Experiments

1. Verify the quality of the vulnerabilities and attacks injected

2. Test one IDS for databases

3. Test two top Web Application Vulnerability Scanners (HP WebInspect, IBM Watchfire AppScan)
## Tested Web Applications

<table>
<thead>
<tr>
<th>Web Application</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TikiWiki</td>
<td>SourceForge 2007 finalist</td>
</tr>
<tr>
<td></td>
<td>Widely used wiki</td>
</tr>
<tr>
<td></td>
<td>Official Firefox Support site, KDE wiki, ...</td>
</tr>
<tr>
<td>phpBB</td>
<td>SourceForge 2007 winner</td>
</tr>
<tr>
<td></td>
<td>Most widely used open source forum solution</td>
</tr>
<tr>
<td></td>
<td>Millions of users worldwide</td>
</tr>
<tr>
<td>MyReferences</td>
<td>Manages publications</td>
</tr>
<tr>
<td></td>
<td>Custom made with 13 PHP files</td>
</tr>
</tbody>
</table>
1: Verification of the Injection

Some vulnerabilities could not be attacked:
- Multiple protection of the variable
- Different variables with the same name
The IDS missed five attacks due to a bug in the code of one of its core functions.
Web application vulnerability scanners missed most vulnerabilities:
- They depend on the browser output
- They do not rely on a direct access to the database
Conclusions

- Importance of data associated to the most critical security vulnerabilities
- The Vulnerability Injection and the Attack Injection can be valuable security tools
- Their effectiveness assessing security mechanisms
- Security mechanisms need to be improved
Future Work

• Implement other vulnerabilities, like XSS

• Implement other fault types

• Going from prototype to a stable product that can be used by a large community of users
Questions?

Our objective: Build a real tool

Questions and suggestions are welcomed

http://xkcd.com/538