

# CGI, Perl & PHP Security Briefing

Written & Compiled By Erik Holm [eh@weboventions.com](mailto:eh@weboventions.com)

## Introduction

These are just some of the most common security issues to be aware of when programming web applications using the CGI environment, Perl and PHP.

The CGI environment is the Common Gateway Interface used by Perl for access to an application on the web server through the user agent. Perl is a server-side programming language that uses an interpreter that compiles the script and executes it on the web server. PHP is the server-side preprocessing hypertext language that is embedded in SGML documents and parsed upon the user agent's request for the document, then executed with the results displayed to the web application user.

This briefing includes sections on Programming Security & Web Security Resources. The next briefing will be on counter measures to combat threats to web application security.

## Programming Security

### 1) Variable Declarations

In some cases hacker can create variables that are declared in the global namespace through forms or the directly through the URL. If a web application allows the use of global variables it can possibly be exposed to remote variable declaration. Web applications should not trust variables they have not definitively set.

- Variables in PHP and Perl should be declared locally not globally
- Variables should be interpreted within the context or functions where they are used
- In Perl "use strict" in each programs forces variables to be declared locally
- In PHP global variables can be turned off with register\_globals set to on
- CAPCHA can be used to prevent automated agents from filling out forms
- Variables should be renamed and filtered when handed from the HTML form to the server
- Tainting variables in Perl can be used to mark all user created data
- Filters can be created in PHP and Perl to eliminate malicious data in form variables
- Using intval() and htmlspecialchars() in PHP can initialize vars and protect from exploits
- HTTP environmental variables can be used to check variable data and http calls
- The safe\_mode configurations in PHP can be set to "on" to enable restrictions including:
  - The ability to restrict which commands can be executed
  - The ability to restrict which functions can be used
  - Restricts file access based on ownership of the file
  - File uploading (see below)

### Bad Code

Here the \$auth variable is not predeclared locally:

```
<?php
if ($var == "setauth") {
    $auth = 1;
}
?>
```

If the hacker creates the \$auth variable in the global namespace the script might still think the authentication was successful.

## Exploits

If instead of using a form to call an application ap1.php, a hacker calls the URL directly

```
http://sitename/ap1.php?var=4&auth=1
```

Not only will \$var = 4, but the \$auth variable will contain the number 1.

## Good Code

This function prevents user-supplied text from containing HTML markup

```
<?php
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
?>
```

To register\_globals on shared hosting web servers:

Modify the .htaccess file

```
<FilesMatch "\.(php|html?)$">
php_flag register_globals on
</FilesMatch>
```

intval() can be used to ensure integers are numbers (but watch out for rounding issues). This provides a valid ranges for the month:

```
$month = intval($_REQUEST['month']);
```

```
<?php
$int = 0.99;
echo intval($int); // returns 1
$intvar = 0.99;
echo intval($int); // returns 0
?>
```

The preventive method to avoid automated form agents from access is CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart). Prior to generating a form Text\_CAPTCHA is implemented

```
<?php
require_once 'Text/CAPTCHA.php';
$captcha_test = Text_CAPTCHA::factory('Image');
$captcha_test->init(150, 150);
?>
```

## 2) Unvalidated User Input

Unvalidated or improperly validated input is ripe to be exploited. You can't just use client-side validation. User-provided data simply cannot be trusted, so ensure that the input you receive from the user is what you want.

- Bad input can include obscuring it with null characters or appending it with other data
- Be completely restrictive and only allow characters that are needed
- Watch out for metacharacters like the ; where code additions can be appended, but also &, \$, :, ", and ()
- You can also restrict tag characters like <, >, [, ] to prevent HTML from being added
- File and directory characters are also good to ban like / and :
- SQL case sensitive keywords such as "FROM", "LIKE", OR, and "WHERE" can also be restricted
- Calls to protocols can be barred like http, ftp, etc.
- SQL is susceptible to query injection that will return the username without validating the password
- Escaping characters with single quotes (') or using PHP's addslashes() function can eliminate the problem
- Setting the magic\_quotes\_gpc php configuration to off, will apply addslashes() to all GET, POST or cookies
- The PHP mysql\_real\_escape\_string() can also be used instead of the addslashes()

## Exploits

```
SELECT * FROM users WHERE name='$username' AND pass='$password';  
User submits -> ' OR '1'='1
```

This results in this query being sent to the database as:

```
SELECT * FROM users WHERE name='known_user' AND pass='' OR '1'='1';
```

## Good Code

```
if (get_magic_quotes_gpc()){  
    $_GET = array_map('stripslashes', $_GET);  
    $_POST = array_map('stripslashes', $_POST);  
    $_COOKIE = array_map('stripslashes', $_COOKIE);  
}
```

## 3) File Handling

Files can be opened, written to and saved using server side programming languages like Perl and PHP. This is typical in file sharing programs. These programs are risky because spyware, trojans and exploits can be saved on the system. If users can upload and then access their files, such as Perl files, PHP files or configuration files they can then be exploited by hackers. These types of files must be protected! With file uploads an attack can even be made on other servers such as databases and file servers. If the hacker has the ability to `chmod()` (change the attributes) of a file he then can execute them. Attacks can use the PHP `include()` and `require` commands.

- Don't let just any user upload or FTP files if they can possibly use other scripts to access or run these files
- On the Apache web server the `.htaccess` file can be used to limit file access
- The `chmod()` command can be used to define the file attributes on Linux or Unix systems
- The `chmod()` ability should only be available to the web master, web server's administrator or root user
- In PHP safe mode `open_basedir` can be configured to limit file uploads and stop a remote file attacks
- Never allow just any user to create a file containing PHP or Perl code that can be executed
- Set `allow_url_fopen` to off to completely stop remote files from being opened
- Closing files after opening them in a program is good coding practice

## Exploits

With the following HTML form a user can upload a file to a server directory, but the size restriction is not processed until after the file is saved. This theoretically allows any size file to be uploaded:

```
<FORM method="post" enctype="multipart/form-data">  
  < input type="SUBMIT">  
  <input type="file" name="file1">  
  < input type="file"hidden" name=" size" value="512">  
</FORM>
```

## 4) Error Handling

If a programmer shows program errors to the web user the information displayed can possibly be exploited. Hackers can test web applications for errors to try to gain information about them.

- Don't let programmer errors get displayed to the end user agent
- Error filters in CGI and PHP can redirect errors and give HTML output to the user
- Errors should be logged and can be emailed to alert the site administrator
- Set PHP configuration error handler `display_errors` `php.ini` value to "0"
- Set the `error_log` ini variable to "1" and then check the error logs frequently
- The `set_error_handler()` function can be used as a way to handle generic errors
- In PHP set `display_errors` off, `log_errors` on (after debugging the code)

## 5) Data Transmission Encryption

Sending unencrypted data using HTTP, Email or FTP protocols for transmissions is not recommended for sensitive data. The data packages sent over TCP/IP can be picked up by packet sniffers. Sensitive data can include credit card numbers, customer personal information and usernames and passwords.

- HTTPS, SFTP and other secure protocols or VPN should be used to transmit sensitive materials
- SSL security is then used to transmit the sensitive data to a client application like a browser
- SSH should be used instead of unencrypted Telnet if allowing remote user command line access
- Remote user command line access should not be given root or administrator permissions

## 6) Sensitive Data Storage

Permanent storing of sensitive user data including credit card numbers, user names and passwords, and personal information such as social security numbers, contact information and personal life details without the user permissions should be done on secure servers and not on web servers.

- Physical security to the server is the usually the biggest risk in sensitive data storage
- Sensitive data should be stored permanently on a secure file server offline in a locked box or room
- Transferring the data from the web server and backing it up to another secure location is a good idea

## 7) Access Control Level Flaws

Restricted sections on a website that allow certain users to access data can be exploited if hackers gain the identity of the user name and passwords. This may allow them to get access to sensitive information.

- Access to restricted areas should be granted to register users via their user name and password
- Default user names should be deleted for services running on the web server
- On the Apache web server the .htaccess file can be used to limit access
- The administration directory should always use an .htaccess file with a user name and password
- This .htaccess file should be saved with the .htpasswd file below the site's root directory
- The user's access privileges can be checked with every restricted page to prevent direct addressing
- Access can be limited using the user's IP addresses and host names using CGI or PHP filters
- The PHP "include" function can be used to include files with passwords from a non-accessible directory
- Create a directory structure with libraries, classes and configuration files stored in the includes directory
- Using the .php extension forces the server to parse the file, so don't that extension on a PHP file
- An index.php' can be used in a restricted directory to eliminate direct access

## 8) Session Protection & Password Hashing

Web applications can be used to create sessions with user tracking. In the web's stateless environment users can be tracked using sessions or cookies. Each session has components that use a distinct ID. The most common security issue is a session profile being stored in a database with plain text passwords. Password hashing encrypts a password in PHP or Perl before it is stored in the database. A hash is digital fingerprint used to encrypt any data. A common technique employed to recover the original plain text from a hash is cracking or 'brute force processing' where an attacker generates hashes which are compared with those in the user database. Cookies can be accessed through cross site scripting techniques where embedded scripting commands collect other user's cookies.

- Credit card numbers should always use an SSL secured connection
- If the ID is exposed it can be sniffed and used to hijack the session
- Session hijacking is nearly impossible to stop, but the risks can be mitigated
- A user should be revalidated when performing security based actions,
- Don't allow a session-validated user to enter a new password without also reentering the old one
- Always avoid giving out sensitive data to a user who has only been validated by session ID
- In PHP the session\_regenerate\_id function is used to regenerate the ID
- To prevent it escape the characters with filters or pass the data through PHP's htmlspecialchars function
- When storing a password in a session variable hash it using the sha1, sha-256 sha-512 or md5() function

## Good Code

To start a session with a hashed password:

```
if ($_SESSION['sha256password'] == sha256($userpass)) {
    // perform session function
}

<?php

/* Store user details */
$passwordHash = sha1($_POST['password']);
$sql = 'INSERT INTO user (username,passwordHash) VALUES (?,?)';
$result = $db->query($sql, array($_POST['username'], $passwordHash));

?>
```

A salt string can be added for extra security. It is limited to a maximum length is 32 characters. A salt generator will overcome this limit.

```
<?php
define('SALT_LENGTH', 9);

function generateHash($plainText, $salt = null)
{
    if ($salt === null)
    {
        $salt = substr(md5(uniqid(rand(), true)), 0, SALT_LENGTH);
    }
    else
    {
        $salt = substr($salt, 0, SALT_LENGTH);
    }

    return $salt . sha1($salt . $plainText);
}

?>
```

## Bad Code

Automatically setting the variable `$session_auth` in later scripts will modified the value. without insuring the variable using user input can be risky:

```
<?php
    session_destroy(); // Kill any data currently in the session
    $session_auth = "dave";
    session_register("session_auth"); // Register a session variable

?>
```

## 9) Dangerous Functions

PHP has the ability to configure which functions are not allowed to run with `disable_functions`. It can be used to eliminate functions when the web server is started. The following is a list of some of the commonly used PHP and Perl code that can be exploited.

PHP:

`exec` - executes a specified command and returns the last line.

`require()` and `include()` - Both these functions read a specified file and interpret it.

`eval()` - Interprets a defined string as PHP.

`preg_replace()` - When used with the `/e` modifier this function interprets a replacement string as PHP.

`fopen()` - Opens a file and associates it with a PHP descriptor.



### [CSIS](#)

<http://www.isse.gmu.edu/~csis/publication.html>

Publications by Center for Secure Information Systems (CSIS) members.

### [SRI International](#)

<http://www.sri.com/>

SRI International contains highly technical security information for security network experts.

### [Purdue University's COAST Security Archive](#)

<http://www.cs.purdue.edu//coast/archive>

Purdue University COAST Archive Leading resource for tools and documents on security.

### [General Accounting Office's report](#)

[http://www.epic.org/security/GAO\\_OMB\\_security.html](http://www.epic.org/security/GAO_OMB_security.html)

General Accounting Office's report on the information security risks at Department of Defense.

### [Criminal Justice Studies](#)

<http://www.leeds.ac.uk/law/pgs/yaman/cryptog.htm>

Criminal Justice Studies of the Law Faculty of University of Leeds, The United Kingdom.

### [EPL](#)

<http://www.radium.ncsc.mil/tpep/epl/epl-by-class.html>

The Evaluated Products List (EPL) List of products evaluated for security ratings.

### [Glossary of Computer Security Terms](#)

<http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-004.txt>

Glossary of Computer Security Terms (Teal Green Book).

### [SANS Institute Resources](#)

[http://www.sans.org/newlook/resources/IDFAQ/solar\\_sunrise.htm](http://www.sans.org/newlook/resources/IDFAQ/solar_sunrise.htm)

SANS Institute Resources: Intrusion Detection FAQ.

### [CIAC Bulletin](#)

<http://ciac.llnl.gov/ciac/bulletins/d-15.shtml>

CIAC Bulletin Vulnerability in Cisco Routers Used as Firewalls.

### [Secure Socket Layer](#)

<http://home.netscape.com/eng/ssl3/ssl-toc.html>

The Secure Socket Layer Protocol from Netscape.

### [Global Technology Research](#)

[http://www.aracnet.com/~kea/Papers/threat\\_white\\_paper.shtml](http://www.aracnet.com/~kea/Papers/threat_white_paper.shtml)

Global Technology Research: Intelligence-Based Threat Assessments for Information Networks and Infrastructures.

### [US Naval Institute](#)

<http://www.usni.org/Proceedings/Articles98/PROcebrowski.htm>

US Naval Institute: Network-Centric Warfare: Its Origin and Future.

### [Technical Defense](#)

<http://www.devost.net/mgd/documents/montreal.asp>

Technical Defense: Political Aspects of Class III Information Warfare: Global Conflict and Terrorism.

### [United States National Security and Computers](#)

<http://www.devost.net/mgd/documents/digitalthreat.asp>

Technical Defense: The Digital Threat: United States National Security and Computers.

### [FIRST](#)

<http://www.first.org/>

Forum of Incident Response and Security Teams (FIRST).

### [General Accounting Office: Information Security](#)

[http://www.epic.org/security/GAO\\_OMB\\_security.html](http://www.epic.org/security/GAO_OMB_security.html)

General Accounting Office: Information Security: Computer Attacks at Department of Defense Pose Increasing Risks.

### [Information Warfare](#)

<http://www.fas.org/irp/wwwinfo.html>

Information Warfare and Information Security on the Web.

### [CGIsecurity.net/](#)

<http://cgisecurity.net>

A site dealing with web application security.