Damn Vulnerable Web Application (DVWA)

Documentation

Revision 1.2

Date published: 20/06/2010

Contents

Introduction

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

Damn Vulnerable Web Application (DVWA) is a RandomStorm OpenSource project. For further details about the services and products RandomStorm offer please visit; http://www.randomstorm.com.

The DVWA project started in December 2008 and has steadily grown in popularity. It is now used by thousands of security professionals, students and teachers world wide. DVWA is now included in popular penetration testing Linux distributions such as Samurai Web Testing Framework and many others.

License

Warning

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any working web server as it will be compromised. We recommend downloading and installing XAMPP onto a local machine inside your LAN which is used solely for testing.

We do not take responsibility for the way in which any one uses Damn Vulnerable Web App (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s that uploaded and installed it.

Installation

DVWA is a web application coded in PHP that uses a MySQL back-end database. DVWA needs a web server, PHP and MySQL installed in order to run. The easiest way to install DVWA is to download and install 'XAMPP' if you do not already have a web server setup.

XAMPP is a very easy to install Apache Distribution for Linux, Solaris, Windows and Mac OS X. The package includes the Apache web server, MySQL, PHP, Perl, a FTP server and phpMyAdmin.

XAMPP can be downloaded from:

http://www.apachefriends.org/en/xampp.html

DVWA default username = admin

DVWA default password = password

Windows

Once you have downloaded and installed XAMPP place the uncompressed DVWA folder in your Apache htdocs folder. Normally located at 'C:\XAMPP\htdocs'. DVWA should now be accessible from your browser at http://127.0.0.1/dvwa.

Linux

Once you have downloaded and installed XAMPP place the uncompressed DVWA folder in your Apache htdocs folder. Normally located at '/opt/lampp/htdocs'. Start Apache with the following command; 'sudo /opt/lampp/lamp start'. DVWA should now be accessible from your browser at http://127.0.0.1/dvwa.

Vulnerabilities

DVWA as the name suggests is vulnerable to the most common types of web application vulnerabilities. DVWA incorporates most of the Open Web Application Security Project's (OWASP) top 10 web application security risks for 2010 as reported in the OWASP TOP 10 document. http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf

The OWASP Top 10 Web Application Security Risks for 2010 are:

- A1: Injection
- A2: Cross-Site Scripting (XSS)
- A3: Broken Authentication and Session Management
- A4: Insecure Direct Object References
- A5: Cross-Site Request Forgery (CSRF)
- A6: Security Misconfiguration
- A7: Insecure Cryptographic Storage
- A8: Failure to Restrict URL Access
- A9: Insufficient Transport Layer Protection

- A10: Unvalidated Redirects and Forwards

Some of the web application vulnerabilities which DVWA contains;

- **Brute Force:** HTTP Form Brute Force login page; used to test password brute force tools and show the insecurity of weak passwords.
- **Command Execution:** Executes commands on the underlying operating system.
- **Cross Site Request Forgery (CSRF):** Enables an 'attacker' to change the applications admin password.
- **File Inclusion:** Allows an 'attacker' to include remote/local files into the web application.
- **SQL Injection:** Enables an 'attacker' to inject SQL statements into an HTTP form input box. DVWA includes Blind and Error based SQL injection.
- **Insecure File Upload:** Allows an 'attacker' to upload malicious files on to the web server.
- **Cross Site Scripting (XSS):** An 'attacker' can inject their own scripts into the web application/database. DVWA includes Reflected and Stored XSS.
- **Easter eggs:** Full path Disclosure, Authentication bypass and some others. (find them!)

Where they are

Low security

Brute Force/Weak Passwords;

http://127.0.0.1/dvwa/login.php

http://127.0.0.1/dvwa/vulnerabilities/brute/

Command Execution;

http://127.0.0.1/dvwa/vulnerabilities/exec/

Cross Site Request Forgery (CSRF);

http://127.0.0.1/dvwa/vulnerabilities/csrf/

File Inclusion;

http://127.0.0.1/dvwa/vulnerabilities/fi/?page=include.php

SQL Injection;

http://127.0.0.1/dvwa/vulnerabilities/sqli/

http://127.0.0.1/dvwa/vulnerabilities/brute/

Insecure File Upload;

http://127.0.0.1/dvwa/vulnerabilities/upload/

Reflected Cross Site Scripting;

http://127.0.0.1/dvwa/vulnerabilities/xss_r/

Stored Cross Site Scripting;

http://127.0.0.1/dvwa/vulnerabilities/xss_s/

Full path Disclosure;

Site wide. Set PHPSESSID to NULL. (Null Session Cookie)
http://www.owasp.org/index.php/Full_Path_Disclosure

Authentication bypass;

If the admin changes the default password (password) and the 'attacker' knows what the default password is. The 'attacker' may access http://127.0.0.1/dvwa/setup.php to reset the database including the default password.

DVWA Security

As well as being vulnerable, DVWA has some other features which aid in the teaching or learning of web application security. DVWAs Security features can be divided into two parts, one is the security levels and the other is PHP-IDS.

The security levels are named low, medium and high. Each level changes the vulnerability state of DVWA throughout the application. By default when DVWA is loaded the security level is set to High. Below is an explanation of each security level and its purpose.

- **High** – This level is to give an example to the user of good coding practises. This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code.

- **Medium** – This security level is mainly to give an example to the user of bad security practices, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.

- **Low** - This security level is completely vulnerable and has no security at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.

Every vulnerability page with in DVWA has a 'view source' button, this button is used to view and compare the source code of each vulnerability in respect to its security level. This allows the user easy access to the source code for comparison of secure and insecure coding practices.

PHP-IDS is a popular PHP Intrusion Detection System (IDS) also known as a Web Application Firewall (WAF). PHP-IDS works by filtering any user supplied input against a blacklist of potentially malicious code. PHP-IDS is used in DVWA to serve as a live example of how WAFs can help improve security in web applications and in some cases how WAFs can be circumvented. PHP-IDS can be enabled or disabled at the click of a button. DVWA has explicit written permission from the owner of PHP-IDS Mario Heiderich for it to be included and distributed within DVWA as long as the licensing is left intact. For further information on PHP-IDS please visit; http://www.php-ids.org

User security

DVWA does not emulate web application vulnerabilities, the vulnerabilities within DVWA are real and therefore great care should be taken on where it is installed. DVWA takes a proactive approach in protecting its users wherever possible. This is done by bold written warnings at the download of the application and within the application itself. DVWA can only be accessed from the localhost and not from remote machines, this is done by setting certain rules within the .htaccess file which is part of the application.

The warning messages state that DVWA should not be installed on live web servers or production machines. Instead it should be installed within a LAN on a machine that is solely used for testing purposes. DVWA at no point should ever be uploaded to an internet facing web server.

DVWA also contains a robots.txt file, if the application was ever uploaded to an internet facing web server this file ensures that the application will not be indexed by search engines.

On each page that contains a vulnerability there are external links to resources which contain further information regarding that particular vulnerability. When external links are clicked it is possible for the remote server to gather information such as the 'Referer' HTTP header. This information contains the URL of where the application is installed, the server administrators could potentially view this information and compromise the sever on which DVWA is installed. For that reason all of DVWAs external links are passed through a trusted third party proxy which clears any sensitive information from the HTTP headers.

User security is of up most importance to the DVWA project. If users do not disable any of these features and follow the advice given, installing and using DVWA will not compromise the security of the machine it is installed on.

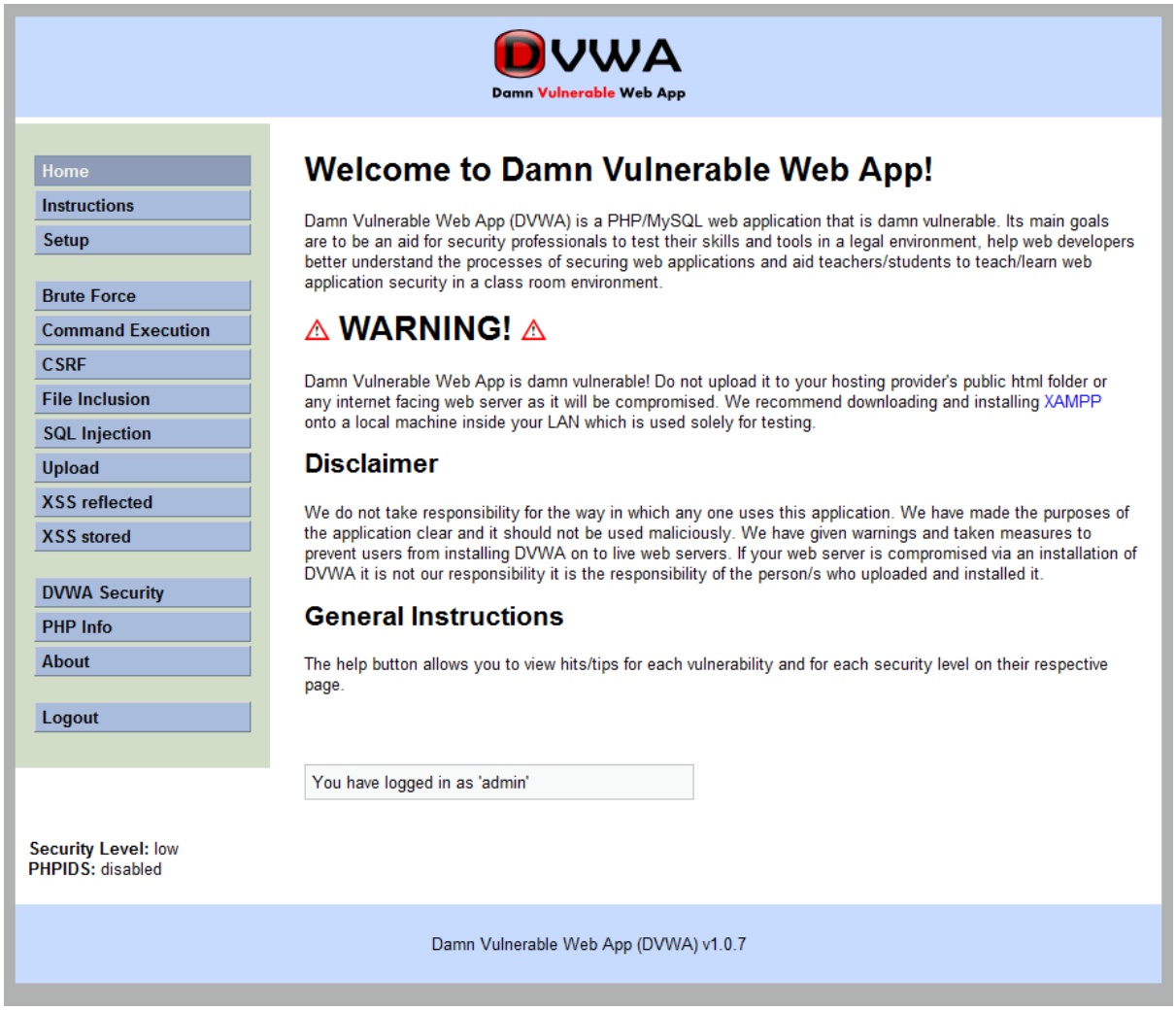
Usage

In this part of the documentation we will give examples of how DVWA can be used to teach and learn web application security in a legal environment.

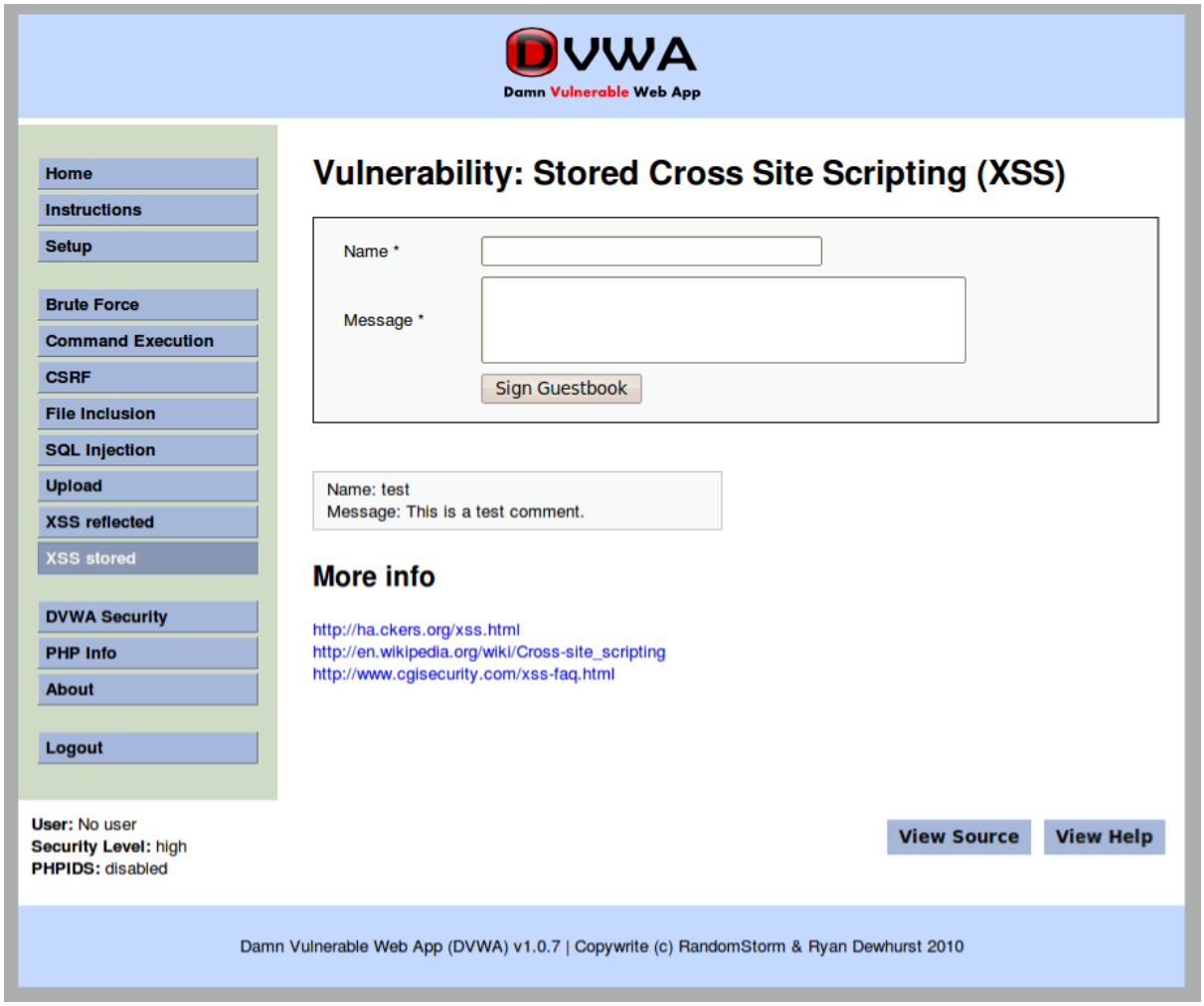
DVWA can be used in a variety of ways. It can be used to teach web application security by showing practical examples and setting challenges for the students. It can be used as just a learning aid, DVWA is designed as such to be as easy as possible to set up and use. There is plenty of information within DVWA to help the beginner get started. DVWA can also be used as a reference to secure coding, if a developer is not quite sure if they have protected their application against XSS for example, they can view DVWAs source code as a reference. After all the DVWA source code has been peer reviewed by thousands of security professionals and students.

Once the user has set up a web server and the MySQL database, to begin they will need to point their browser to the 'localhost' web server. They will be greeted with instructions guiding them through the simple two button click installation process.

DVWA main screen:



As an example we will show how a user might exploit the Stored (type-2) XSS low security level vulnerability.



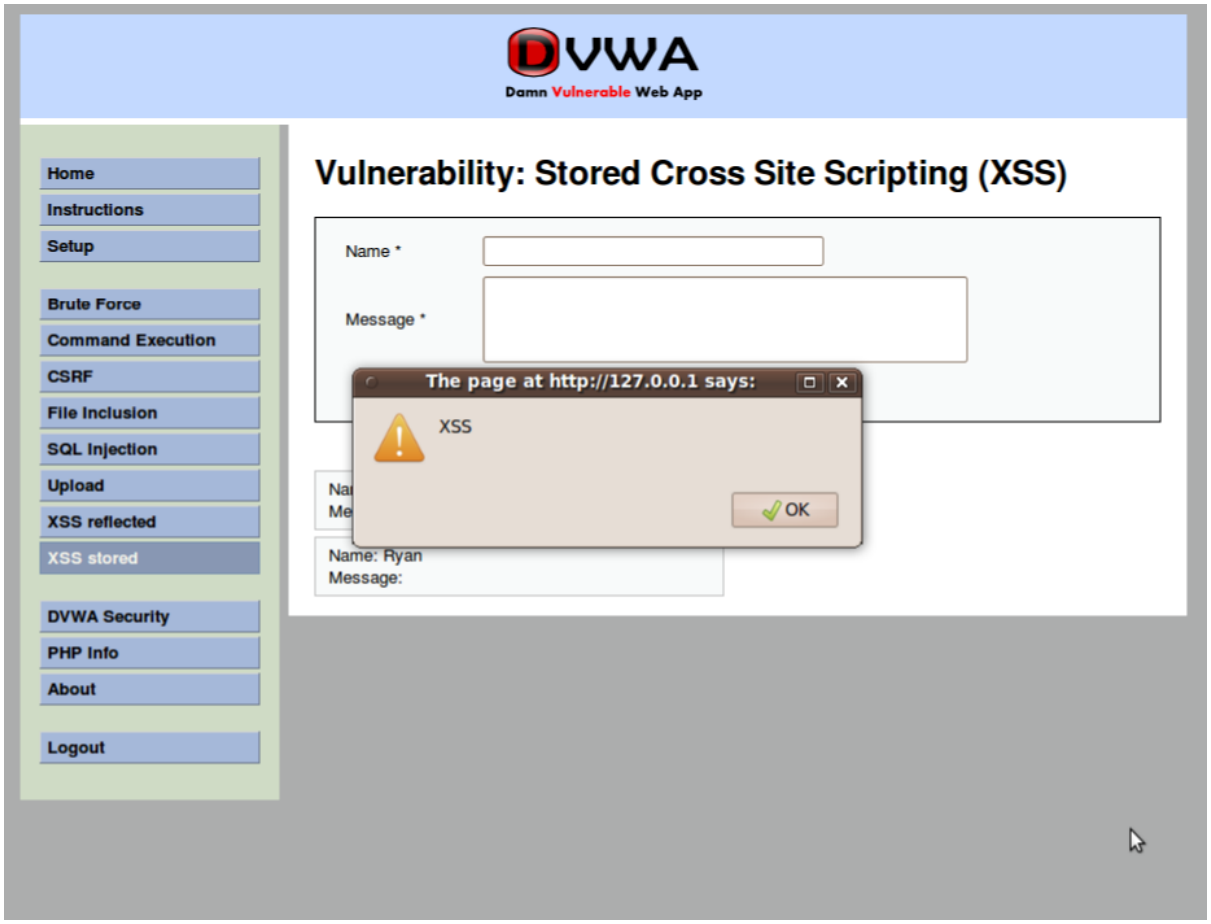
This particular vulnerability has been placed in a guestbook type function. The idea is that a legitimate user leaves comments on a web page that includes their name. As the user supplied input is stored permanently in the backend database, if there were no input sanitisation a malicious user could permanently store their payload within it. Output validation could work here to stop the malicious payload from being executed however as we will see this particular guestbook does not sanitise input nor output properly.

Let's take a closer look at the source code behind the guestbook; we do this by pressing the 'View Source' button on the bottom right hand corner. The source code is coloured within DVWA to help with its readability.

<?php

```
if(isset($_POST['btnSign']))

{

        $message = trim($_POST['mtxMessage']);

        $name= trim($_POST['txtName']);


        //Sanitize message input

        $message = stripslashes($message);

        $message = mysql_real_escape_string($message);


        //Sanitize name input

        $name = mysql_real_escape_string($name);


        $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";


        $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>' );


}


?>
```

We have two variables passed from the form which contains user supplied input, these are $name and $message. The first thing we do is use the trim() PHP function to remove any white space from the beginning or end of the strings. The $message variable is passed through the stripslashes() PHP function to remove any slashes and then also passed through the mysql_real_escape_string() PHP function to escape any special characters; this prevents SQL Injection. The $name variable is only passed through the mysql_real_escape_string() function before being placed in the final query ($query). So as you can see there has been some input sanitisation, but is it enough?



As you can see from the above screen shot we have successfully injected a XSS payload into the database. In this example we used the '<script>alert('XSS');</script>' payload within the $message variable. If we take a look at the high security level source code for the same vulnerability it should give us some clues as to why the low security level is insecure.

```
<?php


if(isset($_POST['btnSign']))

{


        $message = trim($_POST['mtxMessage']);

        $name = trim($_POST['txtName']);
```

```
        //Sanitize message input
        $message = stripslashes($message);
        $message = mysql_real_escape_string($message);
        $message = htmlspecialchars($message);

        //Sanitize name input
        $name = stripslashes($name);
        $name = mysql_real_escape_string($name);
        $name = htmlspecialchars($name);

        $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";

        $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');
}

?>
```

If you compare the low security level source code to the high security level one you will notice that the high security level source code has some extra input sanitisation. Both the $name and $message variables are passed through the htmlspecialchars() PHP function. The htmlspecialchars() function converts special characters to HTML entities, therefore the user input is HTML encoded meaning that it is just displayed as normal HTML rather than being executed.

In this example we used a very simple JavaScript alert box to show the vulnerability existed. It could have easily been a complex AJAX script stored on a remote web server that stole your session cookies, installed malware onto your computer or tricked you into supplying your bank accounts username and password.

Troubleshooting

Q. SQL Injection won't work on PHP version 5.2.6.

A. If you are using PHP version 5.2.6 you will need to do the following in order for SQL injection and other vulnerabilities to work.

In the file .htaccess:

Replace:

<IfModule mod_php5.c>

php_flag magic_quotes_gpc off

#php_flag allow_url_fopen on

#php_flag allow_url_include on

</IfModule>

With:

<IfModule mod_php5.c>

magic_quotes_gpc = Off

allow_url_fopen = On

allow_url_include = On

</IfModule>

Q. Command execution won't work.

A. Apache may not have high enough privileges to run commands on the web server. If you are running DVWA under Linux make sure you are logged in as root in Windows log in as Administrator.

Q. My XSS payload won't run in IE.

A. If you're running IE8 or above IE actively filters any XSS. To disable the filter you can do so by setting the HTTP header 'X-XSS-Protection: 0' or disable it from internet options. There may also be ways to bypass the filter.

Further information

Contact: dvwa@dvwa.co.uk

Website: http://www.dvwa.co.uk

Download: http://sourceforge.net/projects/dvwa/

SVN: http://dvwa.svn.sourceforge.net/svnroot/dvwa

Credits

Craig - www.youreadmyblog.info

Jamesr - www.creativenucleus.com / www.designnewcastle.co.uk

Ryan Dewhurst - www.ethicalhack3r.co.uk

Tedi Heriyanto - http://tedi.heriyanto.net

Tom Mackenzie - www.tmacuk.co.uk

For a complete list of contributors please see the about page within DVWA as it is updated more regularly than the documentation.