Securing websites

Executive Summary:

This paper discusses some of the common ways that web servers are attacked and details various techniques in which they – and by extension the websites they host – can be protected.

Author: Chris Mitchell,
SophosLabs Australia
chris.mitchell@sophos.com

Table of Contents

1. Introduction

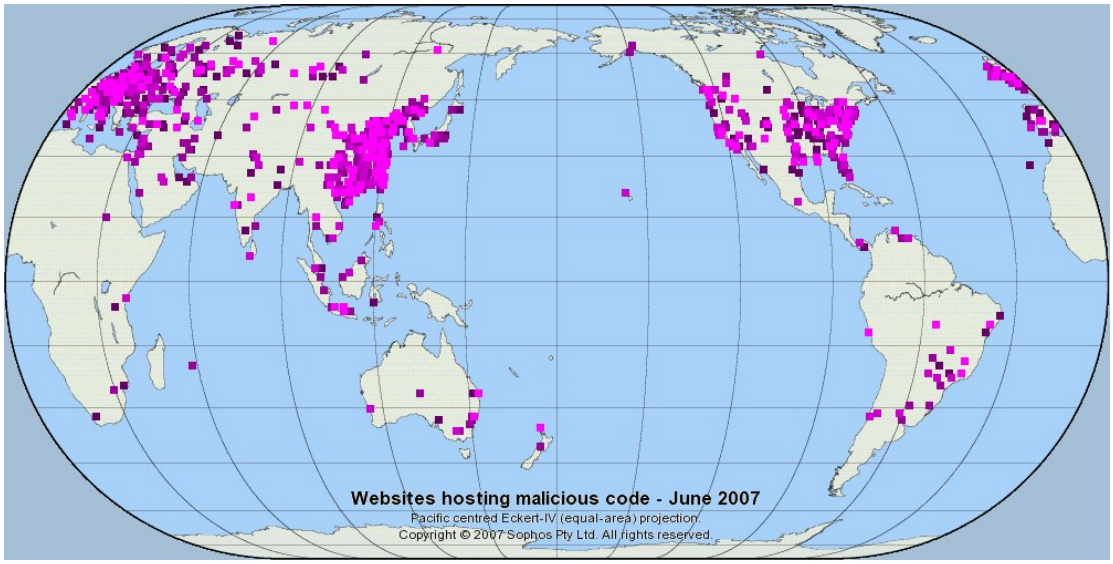
For systems like servers that are designed to be 'always on', security is an important issue. Web servers are the backbone of the internet. They provide the core services and functionalities of the billions of websites around the world and, as a result, act as a repository for the personal data of everyone who visits them. Ensuring that servers are secure from outside attack is a prime concern for any organisation who rely on them.

In the last few years attacks against web servers have increased substantially. As the map below shows, it is immaterial where in the world you base a web server: malicious code respects no boundaries. The threat is not only international, but now comes from organised criminal gangs looking to harvest passwords, financial details and other information, rather than teenage hackers looking to cause mischief. In most cases an attack occurs unobtrusively, with servers and websites corrupted with malware designed to infect as many users as possible.

Web servers are particularly vulnerable as they are 'open' by nature, with users encouraged to send and receive information to them. The HTTPD (HTTP server daemon), database software and code behind a website can each be re-written by a criminal and their original function altered.

However, that is not to say that web servers cannot be protected. They can, but it requires an integrated approach from website administrators, programmers and designers alike, with areas such as anti-virus software, operating systems (OS) and access permissions requiring constant review.

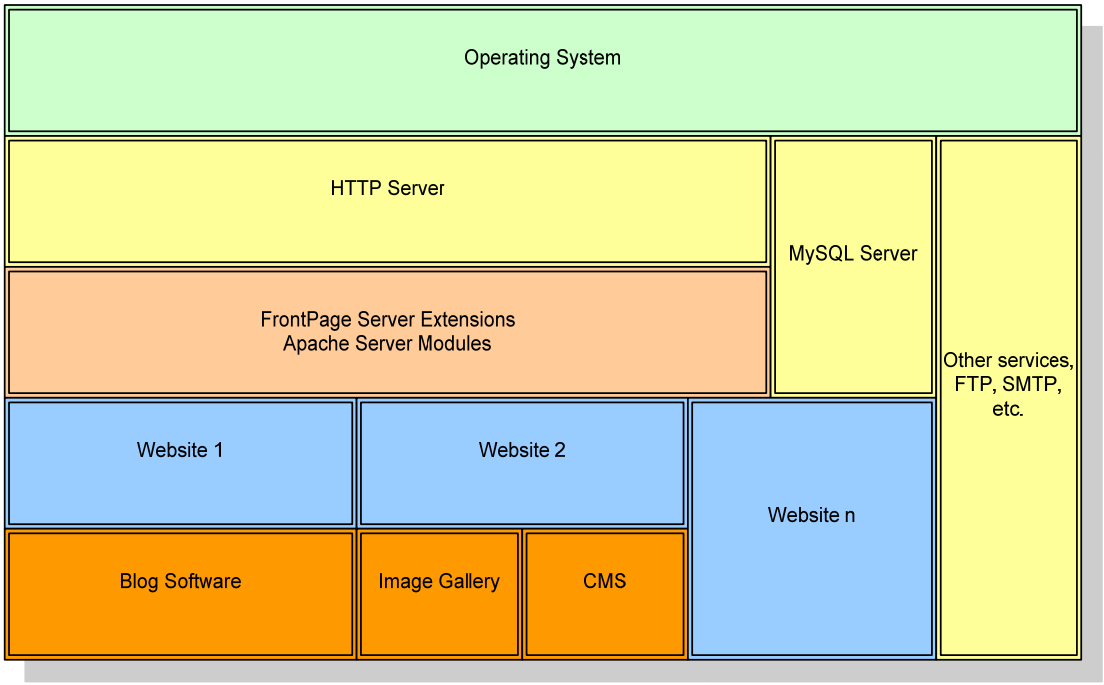
This paper will explore many of the common areas that lead to a compromised web server and the ways of preventing them.



2. Secure foundations

The first step in designing, building or operating a secure website is ensuring that the server that hosts it is as safe as possible.

A web server is made up of layers that provide multiple avenues of attack, as the diagram below shows. Remember, each block is a possible target.



The foundation of any server is the OS and the secret to ensuring that it remains secure is simple: keep it updated with the latest security patches. Doing so could not be easier, with Microsoft [1], together with many flavours of Linux, allowing organisations to apply the patches automatically or launching them with a simple mouse click.

However, remember that hackers too automate their own attempts with malware designed to jump from server to server until it finds one which is unpatched. This is why it is important to ensure that your patches are up-to-date and installed properly, as any server running old patches will become a victim.

You also need to remember to update any software components that run on a web server. Anything that is non-essential, such as DNS servers and remote administration tools like VNC or Remote Desktop, should be disabled or removed. If remote administration tools are essential, however, then avoid using default passwords or anything that can be easily guessed [14]. This is not only applicable for remote access tools, but user accounts, switches and routers as well.

The next area to be addressed is anti-virus software. This is a must for any web server – whether it running Windows or Unix – and, combined with a flexible firewall, is one of the strongest forms of defence against security breaches. When a web server is targeted the attack will attempt to upload hacking tools or malware immediately, so as to take advantage of the security breach before it is fixed. Without a good anti-virus package, a breach in security can go unnoticed for a significant amount of time.

When it comes to defence, a multi-layered approach is best. In the frontline are the firewall and the OS, while in the trenches is the anti-virus, ready to rush in and fill any gaps that present themselves.

In summary:

- Do not install software components you do not need. Every component is a risk, the more there are, the greater the risk
- Keep your OS and applications patched with the latest security updates.
- Use anti-virus, enable automatic updates and regularly check that these are installed correctly.

Some of these tasks might appear onerous, but do not forget that just a single security hole is enough for an attacker. The potential risks include stolen data and bandwidth, server IP blacklisting, the negative impact on an organisation's reputation and the possibility that your website could become unstable.

The next most important piece of software is the HTTPD itself, with the two most popular alternatives being ISS and Apache.

2.1 Internet Information Services (IIS)

ISS is part of Microsoft Windows and is a popular and commonly used web server, as it requires very little configuration.

When implementing it, however, it is worth remembering the following:

- Disable default services such as FTP and SMTP unless you need them. Disable the directory browsing function unless it is required as it allows visitors to see which files are running on your system.
- Disable any FrontPage Server Extensions that are not being used.

You should also keep ISS fully updated, which can be done by simply enabling the Auto Update function that is found in the Windows Control Panel.

2.2 Apache HTTP Server

Apache is a highly configurable and well-maintained open source web server. It requires a more detailed configuration to deploy successfully, but provides greater control over a web server. Most Apache servers run on Linux/BSD, but it can also run on Windows.

Because configuring Apache is complex, there is not space in this paper to detail the entire procedure. However, the following tips [2,3,4] are worth bearing in mind:

- Deny resource access by default and only allow resource functionality as desired.
- Log all web requests as they help identify suspicious activity.
- Subscribe to the Apache Server Announcement mailing list which can send updates, patches and security fixes.

Websites that require a more complicated functionality sometimes augment their HTTPD with a server-side interpreter via CGI (Common Gate Interface). The two most popular are PHP and ASP.

2.3 PHP and MySQL

PHP is one of the most common server-side scripting languages. It has a very large functional code base, simple syntax, adaptable code and, most importantly, interacts with a large number of database formats. MySQL is one of the most popular database choices to use in conjunction with PHP as it is fast, feature-rich, easy to configure and use.

PHP has often been accused of being security-lax as over the years many exploitable bugs have been found within it. However, it has matured steadily and most of the bugs tend to be avoidable by either configuring the installation correctly and/or writing the code securely.

Here are some configuration tips (writing secure code is covered in a later section) that relate to the variables in the “php.ini” file:

- Set ‘register_globals’ off
- Set ‘safe_mode’ on
- Set ‘open_basedir’ to the base directory of the website
- Set ‘display_errors’ off
- Set ‘log_errors’ on

- Set ‘allow_url_fopen’ off

For more information on these configuration directives and why they are important, please see [6,7,10].

When MySQL is installed it creates a default ‘test’ database and an open ‘root’ account that is password-free. The root account is then automatically given free access to every other database on the server which is why it is important to:

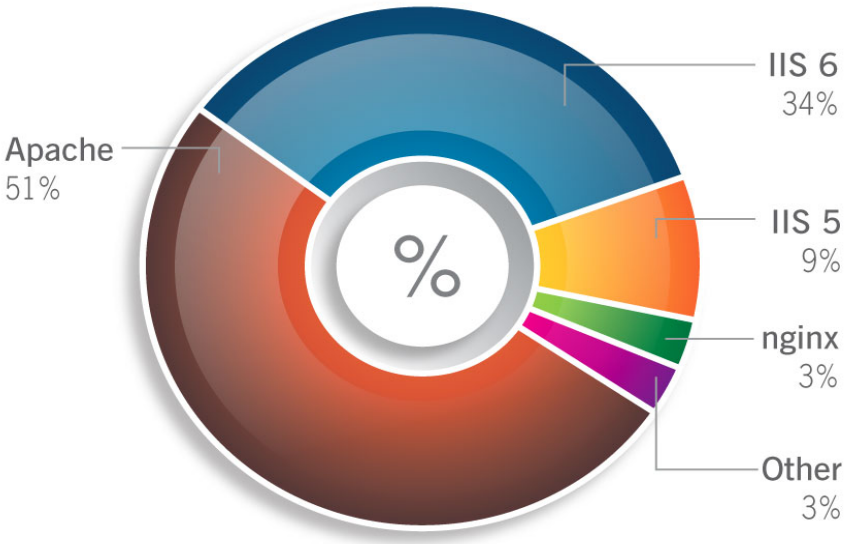
- Change the root password immediately.
- Create a new MySQL user and give it the bare minimum privileges.
- Remove the test database and test users.

2.4 Active Server Pages (ASP)

ASP is a Microsoft add-on that is supported by IIS, though there is also an Apache implementation. ASP is integrated in IIS and so usually requires little or no configuration.

2.5 Security

Anti-virus is generally the final line of defence against an attack which is why web servers, particularly those dealing with dynamically generated content, should have on-access scanning enabled at all times. As the chart below shows, no web server is safe from malware. No matter how secure you think your web server is, there is always a chance that it will get hacked. On-access scanning significantly reduces the chance of malicious code running on the system as it can scan in both ‘on read’ and ‘on write’ modes, and can then deliver an immediate notification as soon as any piece of malware tries to store itself on the server.



While on-access scanning can affect the throughput of the server slightly, but the added security benefits far outweigh any possible performance issues. There are also areas of the system, such as the HTTPD log folder, that can be excluded from the scan, which further reduces any impact on the system.

Attacks against web servers can be generally categorised into two main types: local and global.

- Local attacks usually attempt to steal information or take control of a specific web server.
- Global attacks are generally targeted towards multiple websites and aim to infect anyone visiting them.

Although Linux and BSD are regarded in some quarters as more secure than Windows, they are certainly not exempt from organised crime. They can – and should – have anti-virus software installed. Even if malware cannot execute on the host server because it is protected with anti-virus software, it can still be served up as valid content to website users as some hackers upload it in PHP or ASP, thus rending the OS of the web server redundant.

It is also possible for servers to become infected across a local network. The Fujacks family of worms, for example, infect HTML, PHP and ASP files over shared drives and network shares.

3. External Web Hosting

Most organisations do not have the hardware or stability of bandwidth to host their own web server and as such use external providers. There are three alternatives that are suitable for small and large organisations:

- Shared dedicated hosting.
- Virtual dedicated hosting.
- Dedicated hosting.

3.1 Shared dedicated hosting

This is possibly the most used and abused of all forms of web hosting and involves a dedicated server hosting multiple websites. It is one of the cheapest forms of hosting and consequently one of the most dangerous, as it can take just one infected user to infect everybody else using the server.

An excellent real-life example of the problems inherent with shared hosting can be found in the following SophosLabs Blog posting:

http://www.sophos.com/security/blog/2007/06/172.html

3.2 Virtual dedicated hosting

Virtual dedicated servers – sometimes referred to as elastic servers – are created by using virtualization software to run a number of separate, self-contained virtual servers on just one machine. This is appropriate for any growing organisation as each user have access to their own OS and server software.

3.3 Dedicated hosting

Dedicated servers are exclusively reserved for one user. There are typically two forms available: managed and unmanaged.

- Managed servers have staff to take care of duties such as managing local security issues and troubleshooting.
- Unmanaged servers are unmonitored and slightly cheaper to operate, as any assistance would have to be bought in.

Of the three options presented here, virtual dedicated hosting seems to be the most efficient, being generally cheaper than dedicated hosting but retaining the latter's flexibility and security.

4. Design yourself safer

No matter what you do and no matter how small your website, it will be attacked. Design is intrinsic to security as it can reduce the damage caused by viruses, spyware and other malware.

Try putting yourself in the attacker's shoes and use common sense to plug glaring holes. Some website mistakes are made so commonly – by beginners and old hands alike – that it is worth going over them here.

4.1 Cookies

One of the main problems encountered when designing a web application is that every request for a new page is dealt with independently from the previous request. Asking a web application to 'remember me' is therefore more difficult than it is in normal applications.

There are two methods that web applications use to remember visitors and which are supported by most browsers: cookies and session cookies.

- A cookie is a small file that is created by the browser and stored on the user's computer. It can contain virtually anything, but is usually a name, an expiry date and an arbitrary amount of data like: "Count = 100" or "Member = false".
- A session cookie is similar to a regular cookie, except it allows web applications to store the data in memory.

The difference between the two is that a cookie is stored directly onto the user's computer and stays resident unless manually deleted. A session cookie, meanwhile, is only saved as long as a computer is switched on, and so is lost automatically as soon as the browser is closed. They do have something in common: they can both be tampered with.

Developers often trust the data they retrieve from cookies simply because they developed the code and so it must be good, right? Wrong. Hackers can easily modify a cookie (and in some cases live session data) to fool a website into giving them access to a restricted page.

When designing your system never trust user input, whether it comes directly from visitors, or indirectly through cookies. Try and limit the amount of data that is stored in cookies, especially if it is data that should not be made available to the public. A good rule is to treat any data that is stored on an end-user machine as suspect.

MySpace.com was targeted by a Trojan (JS/SpaceStalk-A) early this year, which stole information stored in cookies and transmitted it to a remote server. This information could theoretically contain confidential information such as login names, internet preferences and passwords.

4.2 Authentication

If your website contains areas that are only intended for certain customers or registered users, you need a way for visitors to identify themselves before they gain access[8].

There are a number of ways to authenticate users: basic authentication, digest authentication and HTTPS.

- Basic authentication allows a username/password combination to be visible inside the web request. Even if the restricted content is not especially secret this is best avoided, since a user might use the same password on many sites. A Sophos poll showed that 41% of users use the same password for all online activity, whether it is a banking site or a local community forum [15]. Try to protect your users against this mistake by using a more secure authentication method.

- Digest authentication – which all popular servers and browsers support – encrypt the username and password securely inside the request. It keeps user names and passwords secure, which creates a better impression on the user and reduces the chance of your server being abused.

- HTTPS encrypts all data transferred between the browser and the server, not just the username and password. You should use HTTPS (which relies on a security system called Secure Sockets Layer, or SSL) whenever you are asking users to provide private or personal data such as their address, credit card or banking details.

When choosing an authentication system, it is good practice to choose the best available. Anything less will worry security-conscious customers and possibly expose them to unnecessary risks.

4.3 Components, libraries and add-ons

Many web developers do not have time to reinvent the wheel. When asked to add a feature that is common elsewhere the simplest approach is to source a package that already contains the necessary component and customise it. Such outsourcing occurs primarily with complex, feature-rich micro-applications such as blogs, forums and content management systems (CMS).

The reason for using pre-built and customisable systems are obvious: they save time and money.

Like all pieces of software, however, add-ons can contain flaws and so it is wise to keep an eye on any packages that are in use and update them regularly. The popularity of some of these packages can sometimes instil a misleading sense of trust among the public and many of the popular products have been found to be exploitable, even when apparently installed and configured correctly.

Popular server-side applications that have had problems in the past with critical, exploitable bugs include:

- Wordpress (blogging software).
- phpBB (forum software).
- CMS Made Simple (CMS Software).
- PHPNuke (CMS Software).
- bBlog (blogging software).

Many of the above (and similar) add-ons are widely used, which makes them very attractive targets for hackers as they greatly increase the number of possible victims. Since most OS and HTTPD software can be automatically updated many developers 'set and forget' certain features, but neglect to update the various add-ons: a dangerous mistake.

Again, the golden rule here is as before, if you do not need it, get rid of it! If your hosting provider supplies such features by default, turn them off. If you are unable to disable them, then you should think about finding a new provider.

4.4 Log Files

Server logs are a very important commodity when managing a website. Most HTTP servers can be configured to save access logs as well as error logs, and this should be enabled at all times as it can be important when conducting a review.

They should also be reviewed regularly as they can provide a better understanding on the threats that websites face. Log files provide an insight into any potential breach by recording, in great detail, every single successful or attempted access to a site.

5. Breaking the code

Writing secure code is not always as easy as it sounds. It not only takes a skilled programmer, but also one that is knowledgeable about specific security issues [9]. There are whole books dedicated to writing secure code so I will only cover the basics here[13].

- Always enable global variables as they can be purposely initialised by a fake GET or POST request.

- Turn off error reporting and ensure that you log-to-file instead, as such information can help attackers provoke a similar problem and then manipulate it to expose further vulnerabilities.

- Do not trust any user data and always use filter functions to strip out special SQL characters and escape sequences.

5.2 SQL injection

SQL injection can be used to attack websites that interact with databases. It occurs when unfiltered input designated by the user is used in an SQL query.

SQL queries can be used to query a database, insert data into a database or modify/delete data from a database. A lot of modern websites use scripting and SQL to generate page content dynamically. User input is frequently used in SQL queries and this can be dangerous as hackers can try to embed invalid SQL code within the input data. Without careful attention, this malicious SQL may be executed successfully on the server.

Take the following PHP code:

```
$firstname = $_POST[“firstname”];
mysql_query(“SELECT * FROM users WHERE
first_name=’$firstname’”);
```

After submitting your first name to the web form, the SQL query will return a list of users that have your first name. If I put my name “`Chris`” in the form, the SQL query would be:

```
“SELECT * FROM users WHERE first_name=’Chris’”
```

This is a valid statement and will work as you would expect, but what would happen if instead of my first name, I put in something like “`’; DROP TABLE; #`”? The statement would then read:

```
"SELECT * FROM users WHERE first_name=''; DROP TABLE
users; #'"
```

The semi-colon allows multiple commands to be run, one after the other. Suddenly the simple statement is now a complex three part statement:

```
SELECT * FROM users WHERE first_name='';
DROP TABLE users;
#'
```

The original statement is now useless, and can be ignored. The second statement instructs the database to drop (delete) the entire table and the third uses the '#' character which tells MySQL to ignore the rest of the line.

The above is particularly dangerous and can be used to display sensitive data, update fields or delete/remove information. Some database servers can even be used to execute system commands via SQL.

Fortunately this type of vulnerability is easily avoided by validating user input. In PHP there is a special function for stripping out potential SQL injection code called 'mysql_real_escape_string'. This function should be used to filter any data that is passed to an SQL statement.

5.3 XSS (cross-site scripting)

This type of attack focuses on websites that display user-supplied data. Rather than attempting to control the database with malicious input, the attacker attempts to attack the website code itself with malicious output.

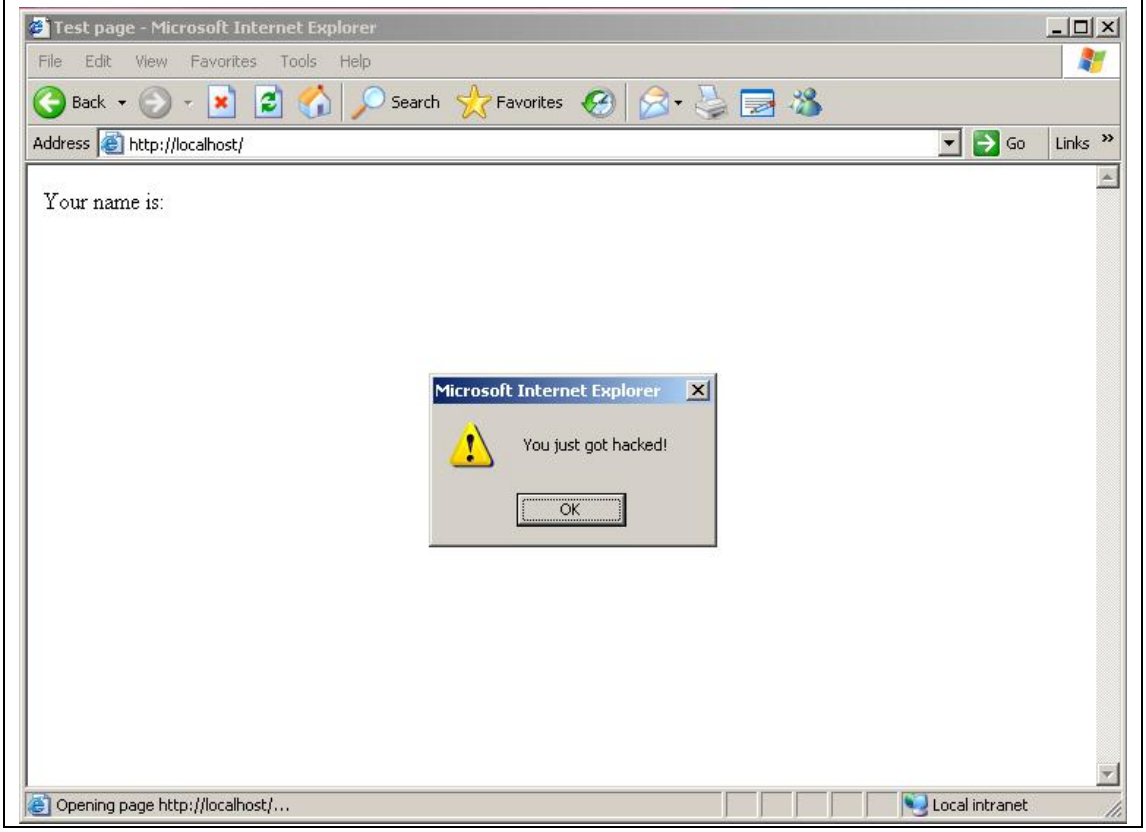
Many sites store the usernames of every visitor in a database so that they can display a specific name when that user logs in. For an attacker it is a simple thing to create a false account, but place malicious code into the username field instead of a name. Such attacks are typically achieved with malicious Javascript scripts that then load content from another website. The database stores what it thinks is the username, but is in fact malicious code. Subsequently, when the website attempts to display the username at the top of the page, the malicious code is unwittingly executed. Since the code could, depending on the circumstances, do just about anything, this is a very real concern and often overlooked by developers. In recent history many high-profile websites have been the victim of XSS attacks, including MySpace, Facebook and Google Mail.

Take the following PHP code:

```
$firstname = $_POST["firstname"];
echo "Your name is: $firstname";
```

After submitting your first name to the web form, the website will display the message on the page. If I put my name "`Chris`" in the form, the message would say: "`Your name is: Chris`".

What if I decided to use "`<script>alert("You just got hacked!");</script>`" instead of my name?



Unfortunately, XSS attacks can sometimes be hard to defend against as they rely on the correct filtering of input and output and then the validation of every single field that can be modified by a user. This includes data retrieved from GET and POST requests, as well as queries that have been returned from the database.

If you use PHP there are a number of packages that can help you filter output easily, an example being CodeIgniter[5]. Alternatively, there is a native PHP function called 'htmlspecialchars' that can be used to filter output.

6. A study of how easy it is

While researching this paper I decided to see how easy it would be to find examples of data leakage and so searched Google for the default log filename for a common FTP client. I found thousands of websites that were publicly displaying (and unknowingly indexing) this seemingly unimportant FTP log file. Each one was a brilliant example of data leakage.

Here is one such (censored) log:

```
99.07.16 08:34 A x:\xxxxxxxx\xxxxxx\xxxxxx\WS_FTP.LOG <--
<Site name> /export/home/<username>/xxxxxx/xxxxxx
WS_FTP.LOG
99.07.16 08:53 A x:\xxxxxxxx\xxxxxx\xxxxxx\home.html -->
<hostname>  /xx/www/xxxxxx-xxx/xxxxhome.html
```

From this I learned a number of interesting things:

- The <site name> gave me the name of the website.
- The <user name> provided the login name on the Linux/BSD style server.
- The <host name> supplied the server's hostname.

This tells me the following about the host:

- The name and IP of the web server.
- The remote path it was copied into.
- The local path it was copied from.

This kind of information is gold dust to any criminal, as by knowing the hostname and username he or she can attempt to gain administrator access. They could also simply discover the web hosting company's phone number or email address and attempt to gain the password via social engineering.

The latter is often easier than attacking the server itself as many web hosting companies undertake minimal security checks before handing out security credentials. This may be because they are often contacted by individual web contractors who are building a site on behalf of a third-party, and so are quite used to getting calls asking for account credentials or password resets.

I myself have done this several times – legitimately of course – and only one of the four different companies I asked required the original business to provide express permission.

Yes, it really is as easy as that.

References & Further Reading

1. How to keep your Windows computer up-to-date, http://support.microsoft.com/kb/311047
2. Apache Security Tips, http://httpd.apache.org/docs/2.2/misc/security_tips.html
3. Securing Apache 2: Step-by-Step, http://www.securityfocus.com/infocus/1786
4. 20 ways to Secure your Apache Configuration, http://www.petefreitag.com/item/505.cfm
5. The CodeIgniter PHP Framework, http://www.codeigniter.com
6. Writing Secure PHP Code, http://solidox.org/index.php?w=module:article,action:view,id:11
7. Ten Security Checks for PHP, Part 1, http://www.onlamp.com/pub/a/php/2003/03/20/php_security.html
8. Creating a Secure PHP Login Script, http://www.devshed.com/c/a/PHP/Creating-a-Secure-PHP-Login-Script/
9. Exploiting Common Vulnerabilities in PHP Applications, http://www.securereality.com.au/studyinscarlet.txt
10. Securing PHP: Step-by-Step, http://www.securityfocus.com/infocus/1706
11. Securing MySQL: Step-By-Step, http://www.securityfocus.com/infocus/1726
12. Apache Attack Samples, http://www.ossec.net/wiki/index.php/Apache_attack_samples
13. The Art of Software Security Assessment, http://www.amazon.com/Art-Software-Security-Assessment-Vulnerabilities/dp/0321444426/ref=pd_bbs_sr_1/105-5809794-4886861?ie=UTF8&s=books&qid=1194490294&sr=8-1
14. Alex, Richard, Paul and Michael - Time to Change Your SSH Passwords, http://www.sophos.com/security/blog/2007/11/661.html
15. Employee password choices put business data at risk, Sophos poll reveals, http://www.sophos.com/pressoffice/news/articles/2006/04/passpoll06.html