Scan Results	
Hostname	webshine.co.uk
Scan date	2015-07-07
Scan Status	34.0% - Vulnerability Testing
Vulnerability Score	**0.68 (F)**
Vulnerability Summary	
High	14 Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code Vulnerabilities in Custom Web Code
Medium	0
Low	5 HTTP Packet Inspection Sitemap.xml File and Directory Enumeration ICMP Echo Request Mailman Detection HTTP Server Detection
Total	19

Vulnerability by Risk Level	Vulnerability by Service	Vulnerability Count
	(Displays High and Medium risk vulnerabilities)	

Security Tests Performed

Type	Tests	Failed	Passed
Infrastructure Tests	11862	8	11854
Blind SQL Injection	1610	14	1596
SQL Injection	1955	14	1941
Cross Site Scripting	3335	0	3335
Source Disclosure	1955	0	1955
PHP Code Injection	920	0	920
Windows Command Execution	1380	0	1380
UNIX Command Execution	1495	0	1495
UNIX File Disclosure	920	0	920
Windows File Disclosure	3105	0	3105
Directory Disclosure	1955	0	1955
Remote File Inclusion	115	0	115
HTTP Header Injection	1035	0	1035

High risk vulnerabilities results for: webshine.co.uk

1. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/?s=**
Affected Parameter: **s**
Vector Used: **VALUE' AND SLEEP(24)='**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/?s=' AND SLEEP(24)='**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:	
Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).	
More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

2. **Vulnerabilities in Custom Web Code** (**High**)

Port:	http (80/tcp)
Summary:	

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_orderby=date&add-to-cart=11244**
Affected Parameter: **product_orderby**
Vector Used: **VALUE' AND SLEEP(24)='**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_orderby=date' AND SLEEP(24)='&add-to-cart=11244**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More infor	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://

mation:	www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

3. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_count=12&paged=1**
Affected Parameter: **product_count**
Vector Used: **VALUE);SELECT pg_sleep(24);--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_count=12);SELECT pg_sleep(24);--&paged=1**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:

Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user. * Source Disclosure: Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them. * Non-SSL login: All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords * Sensitive information sent over non-encrypted page: Make sure all sensitive information is sent over SSL-protected pages.	
Impact:	
Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).	
More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

4. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_count=12&paged=1**
Affected Parameter: **paged**
Vector Used: **VALUE;SELECT pg_sleep(24);--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_count=12&paged=1;SELECT pg_sleep(24);--**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information: See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

For IIS source and path disclosure issues, see:
* IIS 5.0 and below: http://support.microsoft.com/kb/302570
* IIS 6.0: http://support.microsoft.com/kb/814869/en-us

Test ID: 2062

5. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_orderby=name&add-to-cart=11250&product_order=desc**
Affected Parameter: **product_orderby**
Vector Used: **VALUE));WAITFOR DELAY '00:00:24';--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_orderby=name));WAITFOR DELAY '00:00:24';-- &add-to-cart=11250 &product_order=desc**

Recommended Solution:	
* SQL Injection: Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as ' * Cross Site Scripting: Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user. * Source Disclosure: Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them. * Non-SSL login: All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords * Sensitive information sent over non-encrypted page: Make sure all sensitive information is sent over SSL-protected pages.	
Impact:	
Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).	
More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

6. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_orderby=name&add-to-cart=11250&product_order=desc**
Affected Parameter: **add-to-cart**
Vector Used: **VALUE;SELECT pg_sleep(24);--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_orderby=name &add-to-cart=11250;SELECT pg_sleep(24);-- &product_order=desc**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

7. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_order=desc**
Affected Parameter: **product_order**

Vector Used: **VALUE';WAITFOR DELAY '00:00:24';--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_order=desc';WAITFOR DELAY '00:00:24';--**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us

Test ID: 2062

8. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/product/winter-tyre/**
Affected Parameter: **quantity**
Vector Used: **VALUE'));SELECT pg_sleep(24);--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/product/winter-tyre [quantity=1'));SELECT pg_sleep(24);--&add-to-cart=11247]**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

9. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/contact/**
Affected Parameter: **email**
Vector Used: **VALUE));SELECT pg_sleep(24);--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/contact [contact_name= &email=));SELECT pg_sleep(24);-- &url= &msg= &submit=Submit Form]**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

10. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/contact/**
Affected Parameter: **url**
Vector Used: **VALUE' AND SLEEP(24)='**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/contact [contact_name=&email=&url=' AND SLEEP(24)='&msg=&submit=Submit Form]**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

11. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/contact/**
Affected Parameter: **msg**
Vector Used: **VALUE);WAITFOR DELAY '00:00:24';--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/contact [contact_name= &email= &url= &msg=);WAITFOR DELAY '00:00:24';-- &submit=Submit Form]**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:	
Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).	
More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

12. **Vulnerabilities in Custom Web Code** (**High**)

Port:	http (80/tcp)
Summary:	

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/contact/**
Affected Parameter: **submit**
Vector Used: **VALUE;WAITFOR DELAY '00:00:24';--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/contact [contact_name= &email= &url= &msg= &submit=Submit Form;WAITFOR**

DELAY '00:00:24';--]

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us

Test ID:	2062

13. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_orderby=name&add-to-cart=11250&product_count=12&paged=1**
Affected Parameter: **product_count**
Vector Used: **VALUE));WAITFOR DELAY '00:00:24';--**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_orderby=name &add-to-cart=11250 &product_count=12));WAITFOR DELAY '00:00:24';-- &paged=1**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information: See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

For IIS source and path disclosure issues, see:
* IIS 5.0 and below: http://support.microsoft.com/kb/302570
* IIS 6.0: http://support.microsoft.com/kb/814869/en-us

Test ID: 2062

14. **Vulnerabilities in Custom Web Code** (**High**)

Port: http (80/tcp)

Summary:

We discovered vulnerabilities in the scripts listed below. Next to each script, there is a description of the type of attack that is possible, and the way to recreate the attack. If the attack is a simple HTTP GET request, you can usually paste it into your browser to see how it works. If it's a POST attack, the parameters for the POST request will be listed in square parenthesis.

Blind SQL Injection
URL: **http://webshine.co.uk/shop/?product_orderby=name&add-to-cart=11250&product_count=12&paged=1**
Affected Parameter: **paged**
Vector Used: **VALUE' AND SLEEP(24)='**
Pattern found: **Timing test**
Complete Attack: **http://webshine.co.uk/shop?product_orderby=name &add-to-cart=11250 &product_count=12 &paged=1' AND SLEEP(24)='**

Recommended Solution:

* SQL Injection:
Use stored procedures to prevent attackers from altering the queries, and filter user input to discard invalid characters such as '

* Cross Site Scripting:
Filter user input to discard characters such as < and >. Make sure your server does not display error messages that contain input received from the user.

* Source Disclosure:
Make sure all debugging information is turned off from production servers. Scripts should be configured to be executables only, with no ability for a user to view them.

* Non-SSL login:
All login pages should be SSL protected (e.g. have an https:// link). When using non-SSL protected pages eavesdroppers might be able to capture usernames and passwords

* Sensitive information sent over non-encrypted page:
Make sure all sensitive information is sent over SSL-protected pages.

Impact:

Attackers can take control over your database, and in some cases over the operating system (using master..xp_cmdshell, CREATE LIBRARY, etc).

More information:	See http://www.securiteam.com/securityreviews/5DP0N1P76E.html,http://www.securiteam.com/securityreviews/5UP010A6AA.html,http://www.securiteam.com/securityreviews/5IP030K8AA.html,http://www.securiteam.com/securityreviews/5GP0E2K7FO.html,http://www.owasp.org/index.php/Guide_to_SQL_Injection, andhttp://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet For IIS source and path disclosure issues, see: * IIS 5.0 and below: http://support.microsoft.com/kb/302570 * IIS 6.0: http://support.microsoft.com/kb/814869/en-us
Test ID:	2062

Low risk vulnerabilities results for: webshine.co.uk

1. **HTTP Packet Inspection** (**Low**)

Port:	http (80/tcp)

Summary:

This test gives some information about the remote HTTP protocol - the version used, whether HTTP Keep-Alive and HTTP pipelining are enabled, etc.

```
Protocol version: HTTP/1.1
SSL: no
Pipelining: yes
Keep-Alive: no
Options allowed: (Not implemented)
Headers:
Date: Tue, 07 Jul 2015 15:23:10 GMT
Content-Type: text/html, charset=UTF-8
Transfer-Encoding: chunked
```

```
Connection: keep-alive
Set-Cookie: __cfduid=d812e2e6876837266b78ced285b62ea601436282590, expires=Wed,... Vary: Accept-Encoding,Cookie
Last-Modified: Tue, 07 Jul 2015 15:00:14 GMT
ETag: W/"6dfc-51a4a4795b3d1"
Cache-Control: max-age=0, public, must-revalidate, proxy-revalidate
Expires: Tue, 07 Jul 2015 15:23:10 GMT
X-Powered-By: W3 Total Cache/0.9.4.1
Pragma: public
Server: cloudflare-nginx
CF-RAY: 20248c8cff9c230c-LAX
```

Test ID:	10209

2. **Sitemap.xml File and Directory Enumeration** (Low)

Port:	http (80/tcp)

Summary:

The Sitemap file informs search engines about the available pages on your websites. In its simplest form, a Sitemap is an XML file that lists URLs for a site.

This is usually not a security vulnerability, but it does help a potential attacker when gathering intelligence. You should go over the list below and make sure all the pages listed are 'public' pages that are not supposed to be hidden or confidential.

```
/sitemap.xml
<loc>http://webshine.co.uk/sitemap-misc.xml</loc...</loc>
```

Recommended Solution:

Site owners should review the contents of there sitemap.xml file for sensitive material.

Impact:	
None, only an intelligence gathering method	
More information:	https://www.google.com/webmasters/sitemaps/docs/en/protocol.html
Test ID:	10025
3. **ICMP Echo Request** (**Low**)	
Port:	general/icmp
Summary:	
The remote host answers an ICMP echo request (ping).	
Recommended Solution:	
Filter out the ICMP echo requests (8)	
Impact:	
The remote host answers ping, an attacker can use this to determine the host is running.	
Test ID:	9507
4. **Mailman Detection** (**Low**)	

Port:	http (80/tcp)

Summary:

Mailman is a Python-based mailing list management package from the GNU Project. This test detects whether the remote host is running Mailman and extracts version numbers and locations of any instances found.

```
The following instance of Mailman was detected on the remote host:
Installed version: 2.1.18-1
URL: http://webshine.co.uk/mailman/listinfo/
```

Test ID:	7098

5. **HTTP Server Detection** (**Low**)

Port:	http (80/tcp)

Summary:

We were able to detect your web server type and version.

Recommended Solution:

Configure your server to use an alternate name like:
'Wintendo httpd with Dotmatrix display'. See the URL below for more information.

For Apache, add the lines:

ServerSignature Off ServerTokens Prod in httpd.conf For IIS, you can use URLScan to hide the IIS version number.	
Impact:	
Attackers can gain critical information about the host.	
More information:	http://www.securiteam.com/securitynews/5RP0L1540K.html
Test ID:	1035

None risk vulnerabilities results for: webshine.co.uk	
1. **Scan Information** (None)	
Port:	general/tcp
Summary:	
Scanner IP: 67.207.202.9 Target IP: 104.28.10.52 Target Hostname: webshine.co.uk	
Test ID:	9162

2. **Open Port** (None)	
Port:	https (443/tcp)
Summary:	
Test ID:	719
3. **Open Port** (None)	
Port:	http (80/tcp)
Summary:	
Test ID:	719

Scan Results	
Hostname	webshine.co.uk
Scan date	2015-07-08
Scan Status	Done
Vulnerability Score	**100.00 (A+)**

Vulnerability Summary

High	0	
Medium	0	
Low	4	HTTP Packet Inspection HTTP Packet Inspection Sitemap.xml File and Directory Enumeration Mailman Detection
Total	4	

Vulnerability by Risk Level	Vulnerability by Service	Vulnerability Count
	(Displays High and Medium risk vulnerabilities)	

Security Tests Performed

Type	Tests	Failed	Passed
Infrastructure Tests	11862	4	11858

Blind SQL Injection	1610	0	1610
SQL Injection	1955	0	1955
Cross Site Scripting	3335	0	3335
Source Disclosure	1955	0	1955
PHP Code Injection	920	0	920
Windows Command Execution	1380	0	1380
UNIX Command Execution	1495	0	1495
UNIX File Disclosure	920	0	920
Windows File Disclosure	3105	0	3105
Directory Disclosure	1955	0	1955
Remote File Inclusion	115	0	115
HTTP Header Injection	1035	0	1035

Low risk vulnerabilities results for: webshine.co.uk

1. **HTTP Packet Inspection** (Low)

Port:	https (443/tcp)
Summary:	

This test gives some information about the remote HTTP protocol - the version used, whether HTTP Keep-Alive and HTTP pipelining are enabled, etc.

```
Protocol version: HTTP/1.1
SSL: no
Pipelining: no
Keep-Alive: no
Options allowed: (Not implemented)
Headers:
Server: cloudflare-nginx
Date: Wed, 08 Jul 2015 23:31:41 GMT
Content-Type: text/html
Content-Length: 275
Connection: close
```

Test ID:	10209

2. **HTTP Packet Inspection** (Low)

Port:	http (80/tcp)
Summary:	

This test gives some information about the remote HTTP protocol - the version used, whether HTTP Keep-Alive and HTTP pipelining are enabled, etc.

```
Protocol version: HTTP/1.1
SSL: no
Pipelining: yes
Keep-Alive: no
Options allowed: (Not implemented)
Headers:
Date: Wed, 08 Jul 2015 23:31:41 GMT
Content-Type: text/html, charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=d05d10c33eb61f8ab25ffdbb8a56043351436398301, expires=Thu,... Vary: Accept-Encoding,Cookie
Last-Modified: Wed, 08 Jul 2015 23:14:41 GMT
ETag: W/"6ced-51a654db20af4"
Cache-Control: max-age=0, public, must-revalidate, proxy-revalidate
Expires: Wed, 08 Jul 2015 23:31:41 GMT
X-Powered-By: W3 Total Cache/0.9.4.1
Pragma: public
Server: cloudflare-nginx
CF-RAY: 202f958890a02450-IAD
```

Test ID:	10209

3. **Sitemap.xml File and Directory Enumeration** (**Low**)

Port:	http (80/tcp)
Summary:	

The Sitemap file informs search engines about the available pages on your websites. In its simplest form, a Sitemap is an XML file that lists URLs for a site.

This is usually not a security vulnerability, but it does help a potential attacker when gathering intelligence. You should go over the list below and make sure all the pages listed are 'public' pages that are not supposed to be hidden or confidential.

```
/sitemap.xml
<loc>http://webshine.co.uk/sitemap-misc.xml</loc...</loc>
```

Recommended Solution:

Site owners should review the contents of there sitemap.xml file for sensitive material.

Impact:

None, only an intelligence gathering method

More information:	https://www.google.com/webmasters/sitemaps/docs/en/protocol.html
Test ID:	10025

4. **Mailman Detection** (**Low**)

Port:	http (80/tcp)

Summary:

Mailman is a Python-based mailing list management package from the GNU Project. This test detects whether the remote host is running Mailman and extracts version numbers and locations of any instances found.

The following instance of Mailman was detected on the remote host: Installed version: 2.1.18-1 URL: http://webshine.co.uk/mailman/listinfo/	
Test ID:	7098