Manual

Revision 1.2

Thank you for choosing SecureRequest.

The Problem with SQL inject is one of the toughest to solve and the "word on the street" is usually that "there are no shortcuts to protect yourself from it". But now that shortcut is here.

Legal notice

By installing/using SecureRequest module and the SecureRequest Configuration you agree to not hold CadCoordinator AB responsible for any damages caused on your website, server or other appliances. You may only use the module in the way it was intended and as described in this document. An unlicensed copy of SecureRequest must be uninstalled after the evaluation time expires (30 days after installation).

SecureRequest itself will not protect you from SQL inject attacks, but it provides a simple way to find and stop SQL inject attacks. To do this, SecureRequest must be configured with rules to know what to look for. It's up to you to ensure that SecureRequest is configured correctly, but we will provide you with templates that makes this configuration easier. However, by installing/using SecureRequest you also agree not to hold CadCoordinator AB responsible for any attacks not caught by SecureRequest.

About the Reporting Feature

As of version 1.0.0.660 we've implemented a reporting feature that will send a small amount of data whenever SecureRequest is triggered. The reports is only used for statistics and to be able to provide good Security Templates to keep your website safe.

Please note that we, CadCoordinator AB, fully respects your privacy. We will never share this information with anyone else! We will never use any of the data in any other way than stated above.

If you want to opt-out of the reporting, use the DontSendReports setting and no reports will be sent.

The data that is sent contains the following:
The **URL** of the page that caused SecureRequest to react
The **IP address of the visitor**
The **USER_AGENT** value of the visitor(browser version, etc)
The **request method** (POST or GET)
The **rule name** that was triggered (for instance "bannedKey")
The **rule value** that was found (for instance "having 1=1")
The **collection** that the value was found in (Querystring, Form, Cookies, ServerVariables)
The **action** that was taken (Banned, replaced, distorted)

The report is encrypted before it's sent to our server.

Revision history of this manual

1.2 - Added reference information about new settings and updated the layout.

1.1 - Added information about the new Setup Tool

1.0 - Initial release

Index

Thank you for choosing SecureRequest. 2
Legal notice 2
About the Reporting Feature 2
Revision history of this manual 3
Index 3
Installation 5
Using the setup tool 5
Manual Installation 6
Preparing a website 6
Installing the SecureRequest module 7
Configuring and registration 7
Manual Configuration/SecureRequest reference 9
Application settings. 9
DontSendReports *(boolean)* 9
licenseKey *(String)* 9
bannedMessage *(String)* 10
urlToErrorPage *(String)* 10
bannedPatterns *(String Array)* 10
bannedKeys *(String Array)* 11
logFilePath *(String)* 11
keys_ToReplace *(String Array)* keys_ReplaceWith *(String Array)* 11
Disable filtering of specific collections 12
serverKeysToFilter *(String Array)* 12
keysToDistort *(String Array)* 13
Pass-through settings 13
secureIP *(String Array)* 13

securePaths *(String Array)* 14
secureSessionVariables*(String Array)* secureSessionValues *(String Array)* 14
SecureRequest Object 15
Collections 15
Native methods 15
SecureRequest-specific methods 15
DisableFilter *Public sub* 15
EnableFilter() *Public sub* 16
override setting *(String)*, value *(Variant)*, [doClean = True] *(Boolean)* 16
Troubleshooting 17

Installation

Using the setup tool

Since of module-version 1.0.0.614 we include a setup tool with the download package making the setup instructions below more or less unnecessary. We have however chosen to include it in case something forces you to do it manually.

The configurator is still used so step 45 to 52 does still apply no matter if you run the setup tool or not.

The setup tool should be located in the same directory as this manual.

Extract all of the files and run Setup Tool.exe on the web server (where IIS is installed) and then follow the instructions provided in the wizard.

Note! In order for it to work you need to have .net framework 2.0 (or later) installed. You can download it from Microsoft's homepage:

http://www.microsoft.com/downloads/details.aspx?FamilyID=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=en

Manual Installation

This instruction aims to be extremely detailed. Basically, it leads you through creating a new website that will be initially used for testing and configuring, and then turned into the production website. Then it leads you though installing the dll, then the configuration of the rule base and registration of the component.

Preparing a website

1. Open the **Internet Information Services (IIS) Manager** (Located in Control Panel -> Administrative Tools)
2. In the IIS manager; Expand the **<computername> (Local Computer)** node.
3. Expand the **Web Sites** node
4. **Rightclick** the website that you want to secure and choose **All Tasks** -> **Save Configuration to a file**
5. In the dialog that opens (Save Configuration to a File); Type a name in the **file name** textfield.
6. Press the **Browse** button (next to the Path textfield) and navigate to your Desktop.
7. Press OK in the folder browse dialog
8. Press OK in the **Save Configuration to a File** dialog.
9. In the IIS manager; **Rightclick** the **Web Sites** node and choose **New** -> **Web Site (from file)**
10. in the dialog that opens (Import Configuration); Press the **Browse** button (next to the file textfield).
11. Navigate to your desktop and choose the file that you just created (step 5 to 8).
12. Press **open** in the file browse dialog.
13. In the Import Configuration Dialog; Press the **Read File** button and select the webpage in the list.
14. Press the OK button in the **Import Configuration** dialog.
15. In the dialog that opens (IIS manager); Ensure that "Create a new Site is selected" and press the OK button.
16. You will now have two almost identical websites listed in the **Web Sites** node - but one will have the text "(Stopped)" after it.
17. Rightclick the one that has the name "(Stopped)" in it and choose Rename
18. Rename it to **Secured_<websitename>** (replace <websitename> with the original name of the website).
19. Right click the renamed website and choose **Open**
20. In the file explorer that opens; Go to the parent directory of the folder that is displayed (eg. press the "Up" button or the Backspace key)
21. Rightclick the folder that contains the website and choose **Copy** (or select it and press Ctrl+c)
22. Rightclick any clear/white area and choose **Paste** (Or press Ctrl + v)
23. Rename the new copy to **Secured_<original folder name>**
24. Close the explorer window.
25. In IIS manager; right click the **Secured_<websitename> (Stopped)** website and choose **Properties**
26. in the dialog that opens (**Secured_<websitename> (Stopped) Properties**); Select the tab **Home Directory**
27. Press the **Browse** button (next to the Local Path text field) and select the recently created **Secured_<original folder name>** folder.

28. Press OK in the **Browse For Folder** dialog.
29. In the Secured_<websitename> (Stopped) Properties dialog; Select the **Web Site** tab and push the **Advanced** button.
30. Focus the a row in the **Multiple Identities for this Web site** field.
31. Push the **Edit** button.
32. Enter 8080 in the TCP port field instead of 80 (or whatever the original value was) and push ok.
33. Repeat 30 to 32 for each row in the **Multiple Identities for this Web site** field until they are all TCP port 8080.
34. All URLs should now be associated with 8080 rather than the original port.
35. Press ok to close the Secured_<websitename> (Stopped) Properties dialog
36. (If a the dialog **Inheritance Overrides** opens - Press Cancel)
37. Right click the **Secured_<websitename> (Stopped)** website and choose **start**
38. Now, open up Internet Explorer and type the URL of the website that you are working with but at the end of the address add ":8080". for instance, if your webpage is located at *www.mydomain.com*, type: **http://www.mydomain.com:8080**
39. Verify that the site is working as usual.
40. Close the Internet Explorer window.

Installing the SecureRequest module

41. Extract all the files inside the downloaded zip-file into **c:\SecureRequest**
42. Open the start-menu and choose run.
43. type the following text in the text field:
    regsvr32 "c:\SecureRequest\SecureRequest.dll"
44. A dialog should appear saying that the registration was successful then press ok to close the dialog.

Configuring and registration

45. Open the **c:\SecureRequest\Configurator** folder and run the **SecureRequest Configurator.exe**
46. In the SecureRequest Configurator; Press the Browse button and navigate to the **Secured_<original folder name>** folder and press OK.
47. (**If a dialog claims that "The file "global.asa" was not found" - Press Yes and then Yes again in the dialog that asks you if you want it to be created.**)
48. If you have purchased a license key - Press the **+** icon and enter the domain-name and the corresponding license-key - then press the Save button to the left.
49. Select the **Filter Settings** tab and verify that it contains rules. (If it's empty you can either add your own rules or download template-files from www.securerequest.net)
50. Select the **ASP pages** tab and press the **Select all (not includefiles)** button.
51. Press the "Save Changes" button.
52. Verify that all the rows in the list of asp-files has the "**is secure**" status set to **True**
53. Open up a new Internet Explorer window and type in the address of the website you previously entered (step 33).
54. Verify that the website is working.

55. Go to http://www.securerequest.net and use the analyzer to verify that the site is secured (don't forget to add :8080 in the analyzer as well!)
56. Close the Internet Explorer window
57. If everything is OK - open up the **Internet Information Services (IIS) Manager** (as in step 1)
58. Right-click the original website (**not** the one with Secured_ in the name) and choose Stop
59. Right-click the **Secured_<websitename>** and choose Properties.
60. in the dialog that opens (**Secured_<websitename> Properties**); Change the **TCP Port** value from 8080 to **80**.
61. Press OK to close the Dialog
62. Open up Internet Explorer and enter the original webpage address (this time **without** ":8080")
63. Verify that everything is working.
64. Setup complete.

Manual Configuration/SecureRequest reference

Application settings.

The SecureRequest component reads it's settings from the Application Object and therefore we recommend you to specify all the settings inside the Application_OnStart() method located in global.asa - This is to ensure that SecureRequest always has the information it needs to successfully protect you.

DontSendReports *(boolean)*

Description:
If you want to opt-out of the reporting set this DontSendReports to **True** and no reports will be sent. Please see the foreword for more information about the reporting feature.

Default value:
False

Example:

```
Application("DontSendReports") = True
```

licenseKey *(String)*

Description:
If you've purchased a SecureRequest license, this is where you should but it. If you have multiple licenses for the same site you need to delimit them with a comma (,)

Default value:
EVALUATION

Example:

```
Application("licenseKey") = "XXXX-XXXX-XXXX,EVALUATION"
```

bannedMessage *(String)*

Description:
If you prefer to use a custom message that gets displayed when someone sends banned data (keys or patterns) you can specify it here

Default value:
"The data that the server received included banned values. If this is the first time you've seen this error, please try again. If this error reoccur, please contact the webmaster"

Example:

```
Application("bannedMessage") = "Your request contained
suspicious content and was therefore blocked"
```

urlToErrorPage *(String)*

Description:
if you want the users to get redirected to a custom page when the server receives banned data you can specify the URL here.

Default value:
Empty

Example:

```
Application("urlToErrorPage") = "/blocked.htm"
```

bannedPatterns *(String Array)*

Description:
An array containing strings with RegularExpression patterns. If a pattern matches any of the data in the request object it will be banned.

Default value:
Empty

Example:

```
Application("bannedPatterns") = Array(";[\d\D]*--
","having[\b]?")
```

bannedKeys *(String Array)*

Description:
An Array containing strings with banned keywords.

Default value:
Empty

Example:

```
Application("bannedKeys") = Array("DECLARE","EXEC")
```

logFilePath *(String)*

Description:
A physical file path to a text file where SecureRequest will write log entries whenever it found something suspicious. No log entries will be written if this isn't set.

Default value:
Empty

Example:

```
Application("logFilePath") = "c:\securerequest\log.txt"
```

keys_ToReplace *(String Array)*
keys_ReplaceWith *(String Array)*

Description:
keys_ToReplace: An Array containing strings that you want to be replaced. Used in conjuction with keys_ReplaceWith

keys_ReplaceWith: An Array containing strings with the values that will replace the keys found in keys_ToReplace. Used in conjunction with keys_ToReplace

Default value:
Empty

Example:

```
Application("keys_ToReplace") = Array(";","'")
Application("keys_ReplaceWith") = Array(":","''")
```

Disable filtering of specific collections

Description:
If you want to disable the filtering of specific collections in the request object you can set this with the following application variables:

```
Application("DontFilterCookies")  = true
Application("DontFilterServerVariables")  = true
Application("DontFilterQuerystring")  = true
Application("DontFilterForm")  = true
```

This will cause those collections to use the default (unprotected) collections in the request object instead of the filtered ones.

Default value:
false

serverKeysToFilter *(String Array)*

Description:
An Array containing keywords in the Request.ServerVariables collection that you want to search through. Since the ServerVariables collection includes data that uses semicolon - only some of the keywords are included by default. If you want to override this setting you can set your own array with keywords.

Default value:
Array("HTTP_HOST", "URL", "PATH_TRANSLATED", "SERVER_NAME", "HTTP_REFERER", "HTTP_ACCEPT_LANGUAGE", "AUTH_USER", "AUTH_PASSWORD", "REQUEST_METHOD", "HTTP_COOKIE", "HTTP_ACCEPT_LANGUAGE")

Example:

```
Application("serverKeysToFilter") = Array("HTTP_REFERER",
"URL")
```

keysToDistort *(String Array)*

Description:
An array containing keywords that will be html encoded. This is very useful for dynamic websites where the database includes documents that are presented on the website. The html-encoded words will be displayed correctly when displayed in a webpage but the SQL server will not consider them as a command in case of an SQL injection attack. For instance; If the word **SELECT** is distorted it will look like this in the database: **SELECT** but when displayed on a webpage it will be translated into the original **SELECT**

Default value:
Empty

Example:

```
Application("keysToDistort") =
Array("SELECT","DROP","EXEC","DECLARE")
```

Pass-through settings

These features was introduced in version 1.0.0.660 to be able to minimize "false positives". Please not that users that matches the settings you set below will NOT be filtered.

secureIP *(String Array)*

Description:
An array with IP numbers or IP ranges that will NOT be filtered by SecureRequest. By setting this to your intranets IP span you can be sure that you'll not get any false positives. Just be sure that no computer on your network is infected with any viruses that might try to hack your website. Can be used with the **override method**

Note: IP ranges must be delaminated with a hyphen-character (-)

Default value:
Empty

Example:

```
Application("SecureIP") = Array("127.0.0.1","192.168.0.0-
192.168.0.255","77.72.97.128-77.72.97.255")
```

securePaths *(String Array)*

Description:
An array containing strings that if it matches the URL of the current page, it will not filter the result. This can be used to unsecure a part of your website with restricted access (e.g. you'll get a windows dialog where you enter your credentials. This may also be used with the **override method** to use different settings on different parts of your website.

It will only match if the URL starts with the value in the array so the URL /admin/editpage.asp will match the setting /admin/ as well as /admin/editpage.asp **but not** /editpage.asp

Default value:
Empty

Example:

```
Application("SecurePaths") = Array("/admin/","/guestbook/")
```

secureSessionVariables *(String Array)*
secureSessionValues *(String Array)*

Description:
Used to disable SecureRequest if the users session variable matches a specific value. Let's say that you set a session variable called "isAdmin" to "yes" when an admin logs in to the page, you can then tell SecureRequest not to filter any of his requests. Can be used with the **override method** to set different rules for different session variables.

secureSessionVariables: An array containing variable names that SecureRequest will look for.

secureSessionValues: An Array containing values that the session variable must match.

Default value:
Empty

Example:

```
Application("secureSessionVariables") = Array("isAdmin")
Application("secureSessionValues") = Array("yes")
```

SecureRequest Object

The SecureRequest object is designed to fully replace the native Request Object. But it also has some extended features. The following methods should be run on the Request object on each page (not used in the Application object nor in global.asa)

Collections

All of the following collections should mimic the native request collections indifferently.
Note: the SecureRequest is unable to replace values in the Cookie collection and therefore Cookies are only protected by rules specified in the **bannedPatterns** and the **bannedKeys**.

- Querystring (Dictionary)
- Form (Dictionary)
- ServerVariables (Dictionary)
- Cookies (CookieCollection)
- ClientCertificate As Variant

Native methods

The following methods are the same as the native request object. For further information about them - please see the ASP 3.0 Reference provided by Microsoft.

- *Public Property Get* **TotalBytes**() As Long
- *Public Function* **BinaryRead**(pvarCountToRead)

SecureRequest-specific methods

DisableFilter *Public sub*

Description:
This will disable secureRequest temporarily in case you need to do so in order to resolve compatibility issues. Put this command before the code you want to access the unsecured request object and then put EnableFilter() when you're done.

Example:

```
str = Request.form("name") ' Secured data
Request.disableFilter() ' Disable SecureRequest
unsecureStr = Request.form("name") ' Unsecure data
Request.enableFilter() ' Enable SecureRequest again
secureStr = Request.form("name") ' Secured data
```

EnableFilter() *Public sub*

Description:
This will enable SecureRequest after it's been disabled.

Example:
See the example for ***DisableFilter()***

override *setting (String), value (Variant), [doClean = True] (Boolean))*

Description:
Will override the application settings. Will only work on replace rules or if any of the override settings has disabled SecureRequest.

Syntax:

```
Request.override setting, value [, doClean]
```

Setting *(String)* - the name of the setting that you want to override. This is the same name as you use when you set the settings in the Application object

value *(variant)* **-** the value that you want to override the specific setting with

doClean (*Boolean)* **-** optional value that tells the method if you want to apply the new settings. the default value is True . Always set the last override call to True or leave it blank for SecureRequest to respond to your new settings.

Example:

```
Request.Override("bannedKeys", Array(), False)
Request.Override("bannedPatterns", Array(), False)
Request.Override("keysToDistort ",
Array("SELECT","DROP","EXEC","DECLARE"),True)
```

Troubleshooting

I receive a server error when I visit my website after running the configurator for the first time.	This might be caused by SecureRequest not being registered successfully. Verify that you've completed steps 41-44 in the installation guide.
When I try to start the cloned website (step 37) I receive a message that it cannot be started.	This is probably because another website or another software is using port 8080 or that not all of the http-headers has been set to 8080 and is still trying to use port 80. Please verify that you've completed step 33. If that fails, try another port instead (for instance 8090).
When I try to enter www.mydomain.com:8080 I receive an error message in Internet Explorer: The webpage cannot be displayed	Since port 8080 is not the default http-port you've got to manually type **http://** in front of your URL. If you still receive an error, please check so that your firewall settings isn't blocking your request and that IIS and the website is up and running.
I'm having a problem not listed here.	Please visit the support section at http://www.securerequest.net