

Summary: Using an IIS custom module to intercept an incoming request and perform logic on it before allowing the request to proceed onto the website logic. This leverages the IIS 7 pipeline architecture to inject custom code into the process. *This project will show to write a custom module to leverage the pipeline architecture and how to deploy and configure your custom module.*

Problem Statement: In our customer's environment, we need to adhere to the PCI standards and were at risk of failing an audit because we were exposed to a brute Force attack or a hack called Sweet32. This attack makes use of a feature in the webserver called KeepAlive where a single session can be opened and kept open for an indefinite period. A lot of web servers such as Apache allow additional configuration where you can limit either or both the number of requests through a single session, or the time the session may be kept open. In IIS you can either turn on KeepAlive or turn it off. Turning it off has a nominal performance impact because now a new session with the web browser needs to be established for each call from the browser.

To solve this, we wrote this custom module. We use the OnEndRequest event and add a record of incoming request to a dictionary if it is not found. If it is found, we increment a counter and forcibly the session, causing a new session to be created. We also update the last accessed time for each incoming request. Periodically we clear out all the stale sessions.

A few comments on how to deploy the app.

1. The app needs to be deployed to the GAC so it needs to be signed. In this example the password to the file signing the DLL is "password".
2. To get the file into the GAC open a command prompt as admin and run the following.

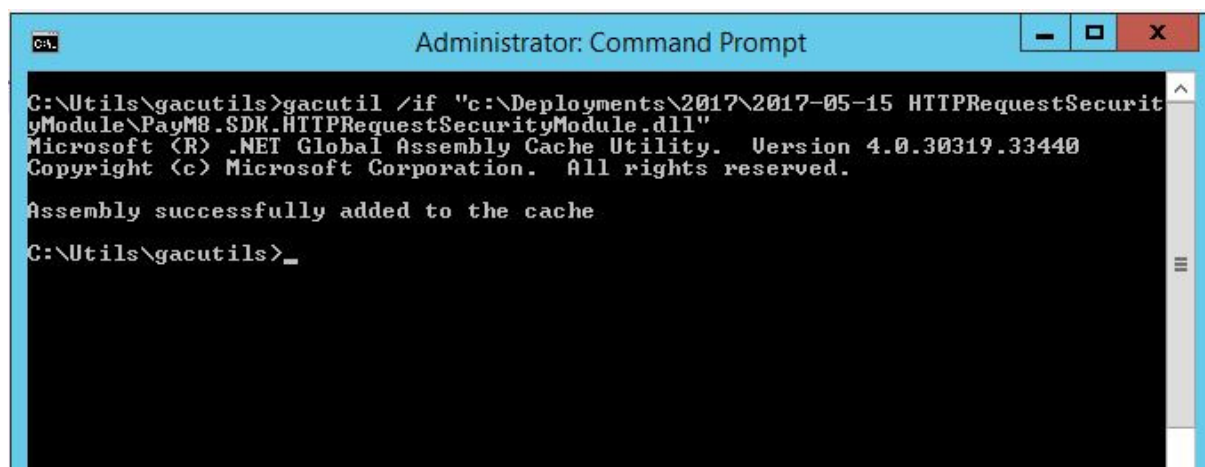
(Note if the .NET SDK is not installed on the server you are deploying to then copy the following three files from your dev environment to the server. I created the folder "[c:\utils\gacutils\](#)"

[gacutil.exe.config](#)

[gacutilrc.dll](#)

[gacutil.exe](#)

You will find these in your dev env "[c:\Program Files \(x86\)\Microsoft SDKs\Windows\v8.1A\bin\NETFX 4.5.1 Tools\](#)" and in the subfolder "[\1003](#)"



```
Administrator: Command Prompt
C:\Utils\gacutils>gacutil /if "c:\Deployments\2017\2017-05-15 HTTPRequestSecurityModule\PayM8.SDK.HTTPRequestSecurityModule.dll"
Microsoft (R) .NET Global Assembly Cache Utility. Version 4.0.30319.33440
Copyright (c) Microsoft Corporation. All rights reserved.

Assembly successfully added to the cache
C:\Utils\gacutils>_
```

3. To check it is installed run...

```

Administrator: Command Prompt
C:\Utils\gacutils>
C:\Utils\gacutils>gacutil /I > c:\temp\ga2c.txt
C:\Utils\gacutils>_

```

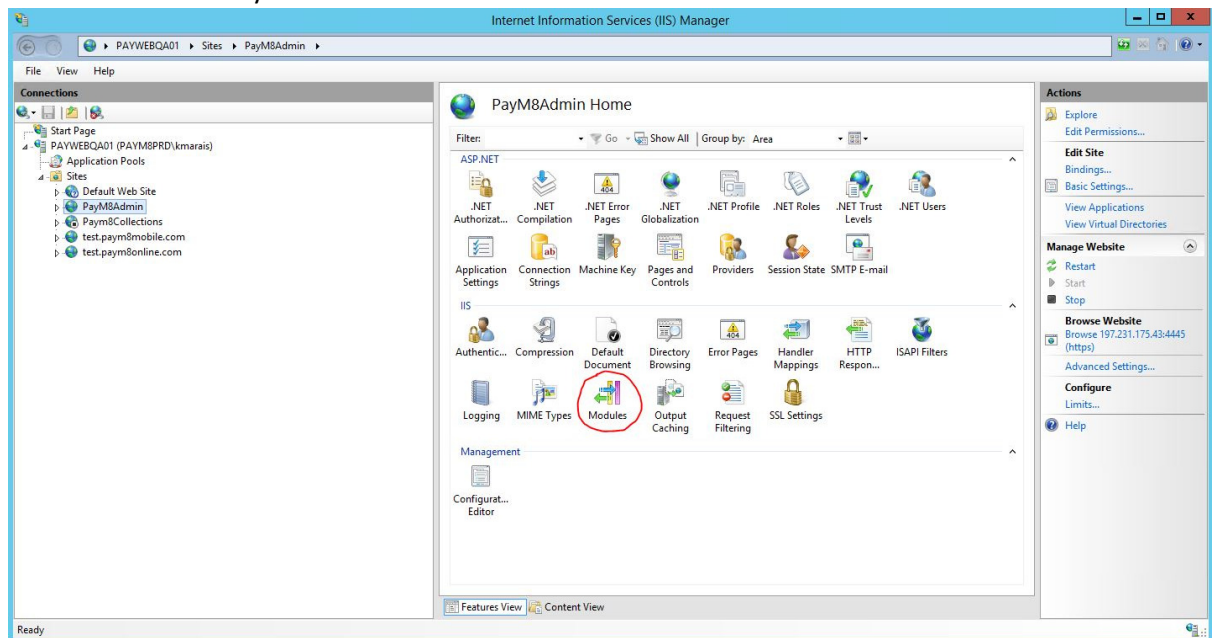
And check the resulting text file...

```

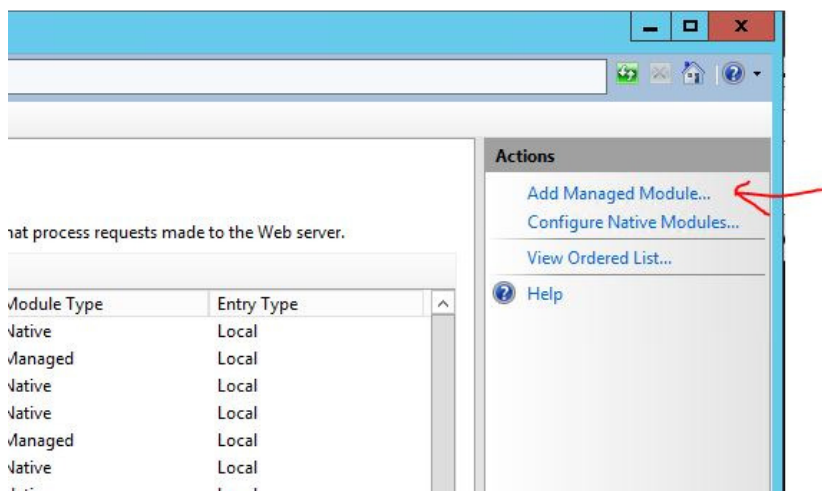
502 ##### Version=6.3.0.0, Culture=en, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL
503 ##### Version=6.3.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL
504 ##### Version=6.3.0.0, Culture=en, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL
505 ##### Version=6.3.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL
506 ##### Version=6.3.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL
507 ##### Version=6.3.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL

```

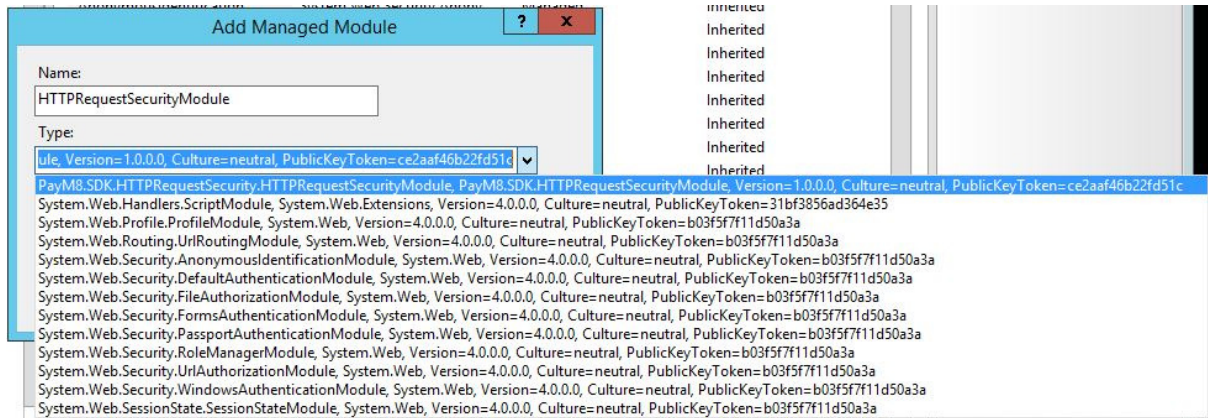
4. In IIS click on one of your websites and show the features. Double click on *Modules*.



5. Click on Add managed module in the upper right hand corner.



- Type in the Name you gave your module and select your module from the Type list. (Note: Not sure why but to get the module to show in the list, even though it was in the GAC, I had to copy the DLL into the BIN folder.) Once it's added you can remove it from the BIN folder.



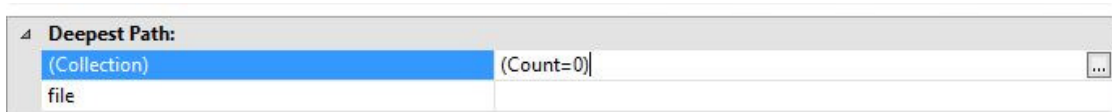
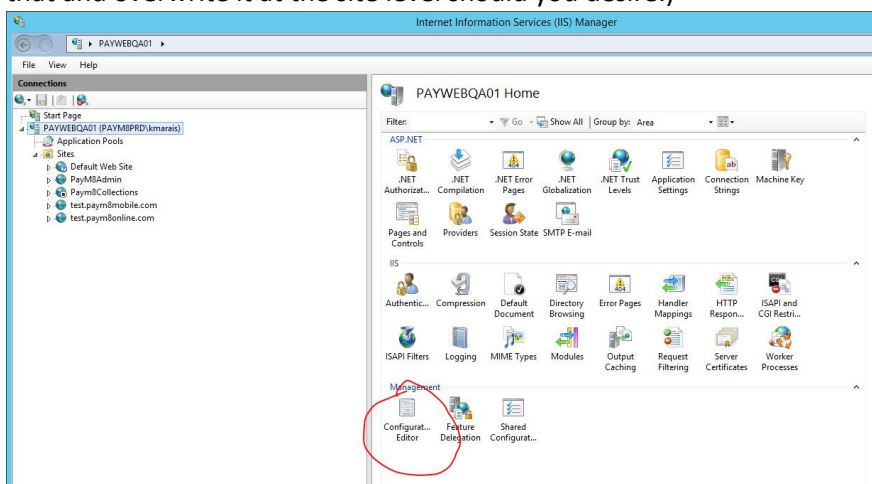
This will add the following to the web config file

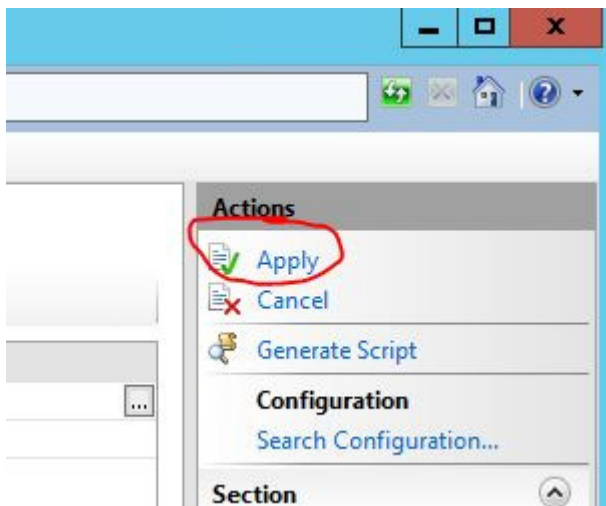
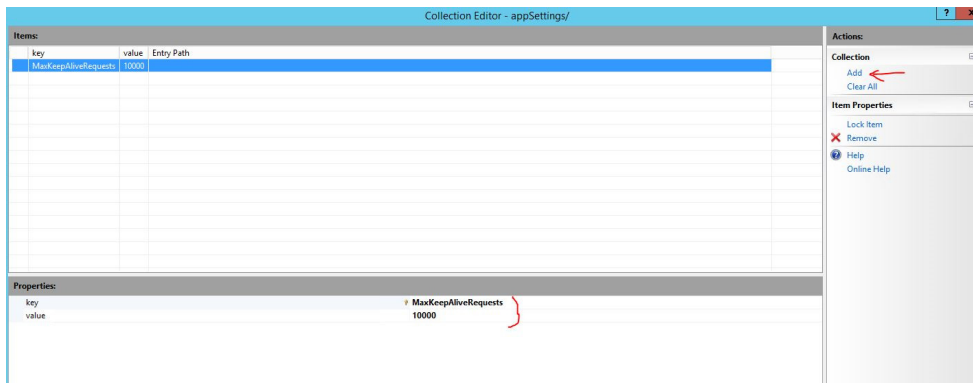
```
<system.webServer>

  <modules>
    <add name="HttpRequestSecurityModule"
        type = "PayM8.SDK.HttpRequestSecurity.HttpRequestSecurityModule,
PayM8.SDK.HttpRequestSecurityModule, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=cfa0f608de62f06e" />
  </modules>
</system.webServer>
```

To add the module to other websites you only need to copy this into their web config files.

- The config you can can either add to the website's web config or if you are using the module across many websites you can centralize this in the Server level config file. (You can also do that and overwrite it at the site level should you desire.)





8. Our module uses nLog to do logging. Where it logs to is in the site's web config file. If you're deploying to a site that does not use nLog just remember to copy the nLog dll to the site's Bin folder.

The output looks as such...

```

2017/05/15 12:24:26.673 Info [7] Current request count = 4
2017/05/15 12:24:26.814 Info [7] OnEndRequest key='197.231.175.43:60297'
2017/05/15 12:24:26.814 Info [7] Found '197.231.175.43:60297' in dictionary.
2017/05/15 12:24:26.814 Info [7] Current request count = 5
2017/05/15 12:24:26.892 Info [7] OnEndRequest key='197.231.175.43:60297'
2017/05/15 12:24:26.892 Info [7] Found '197.231.175.43:60297' in dictionary.
2017/05/15 12:24:26.892 Info [7] Current request count = 6

```

Conclusion: The pipeline architecture is simple to use and quite powerful. I can think of several applications. This application will even work across a load balanced web farm, with or without sticky (always routing to the same web server) sessions. Although in the event of non-sticky sessions it does become redundant. That would essentially be the same as just turning off KeepAlives.