Security is an essential part of a web application and should be taken into consideration from the first stage of the development process. A website couldn’t ever be secure enough unless you would undertake necessary security initiatives to protect the web server from all breaches, because hackers can easily penetrate a web mechanism by exploiting the loopholes of a web server, even if the corresponding source code of a website is duly locked. Even if the software itself is entirely bug-free, with no vulnerability at all, it could be compromised because of the way it interacts with other system, or because of poor operational practices.

Certainly, there are a number of products in the market that claim to make your server more secure, but attacks could be in any form and unpredictable. There is a common misconception that the security of a website could only be supervised while source coding. Whereas indeed, the ASP.NET website can also be appropriately protected at the IIS web server end from vicious intention. Therefore, this article is designed to unveil the unseen security aspects of an IIS web server, in order to shield the website from breaches.

**ESSENTIALS**

In order to harden the Internet information server, the security researcher should have a comprehensive proficiency in .NET technology, ASP.NET in particular, because the .NET centric websites are typically operated via IIS. Apart from that, understanding of web server framework life cycle with essential coding skills wherever it requires, in context of ASP.NET website configuration, will definitely add an advantage.

**IIS BREACHES**

There is almost an inexhaustible number of possible potential attacks against your website, ranging from database loopholes and vulnerability in source code to common viruses, even more through disgruntled employees and social engineering. However, a complete classification of all attacks against the IIS server is beyond the scope of this paper. But generally, the attacker targets the authentication, authorization, and auditing mechanism of a web system. The intruder tries to prove or identify himself as a legitimate user by cracking the corresponding pass keys. Subsequently after being authenticated, the malicious hacker endeavors to gain access to the resources for which he is not allowed indeed, by revoking the access control list maintained on the server, which determines the action of a user. Finally, he erases the traces, since every action of a user is typically recorded on the server during a comprehensive auditing process.

Moreover, some hackers perform ‘passive’ attacks which are much harder to distinguish. Here the hacker doesn’t try to modify anything on your existing system. Instead, passive hackers are only observing existing activity happening during the round-trip communication. Hence, detection, prevention, and combating these classes of sophisticated confrontations can be difficult in an environment that doesn’t abide by virtuous administration and development procedures.

**IIS SECURITY**

IIS is responsible for processing requests received on specific ports. For this motive, WWW services run on the machine, which handle and process requests received on various TCP/IP ports, where port 80 is normally assigned to HTTP. Before the ASP.NET runtime gets in contact with the incoming request, the IIS first verifies the security according to its own configuration. In fact, IIS is the first gatekeeper in the security pipeline of your web application. Hence, IIS provides you with a couple of essential security mechanisms, like authentication, authorization, and confidentiality, which act as a gatekeeper before ASP.NET starts with the request processing. Therefore, the IIS interface offers various security options, as given in Figure 1.1, that can be used to restrict access or deny types of requests.

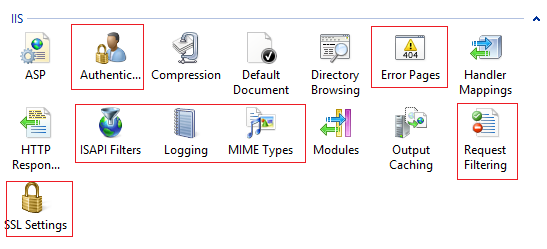


Figure 1.1 Security Parameters

**IP Restriction**

IP restriction enables you to selectively allow or deny access to the files, folder, website, and web server. Overall, the administrator has the autonomy to determine which remote computer can connect to IIS. Custom rules can be built in context of a particular client, or DNS lookup to provision their restriction. A vulnerability of ASP.NET would allow a hacker to potentially bypass any restriction code into your application before your code has a change to run. When a client who is not permitted access requests a resource, a ‘Forbidden: IP address of the client has been rejected (403.6)’ or ‘DNS name of the client is rejected (403.8)’ HTTP status will be reflected and logged. Moreover, there are two terms introduced by IIS in this scenario: IP and Domain Restriction.

Typically, it is advisable to configure a connection restriction at the lowest level in the OSI model as possible. This prevents a misconfiguration in a lower-level component from allowing a hacker to bypass a higher-level restriction.

To configure the default policy on DNS lookup, which permits all access except those clients specially rejected, click the ‘Edit Feature Settings’. Subsequently, the IP and domain dialog box will appear.

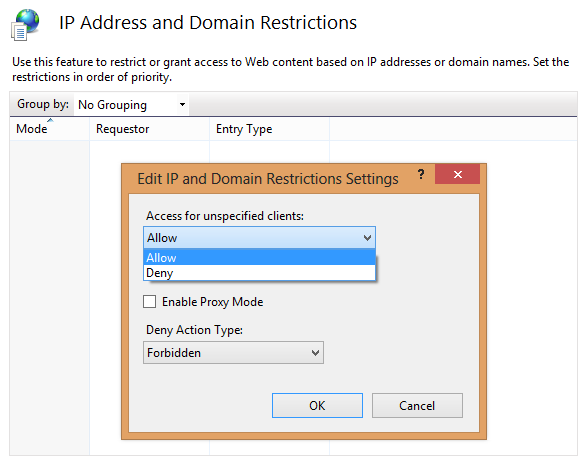


Figure 1.2 Domain restriction

Further, you can create rules for specific remote hosts or subnets. To create rules that allow or deny access, click on ‘Allow Entry’ where you can specify the blacklisted IP address or subnet range as following.

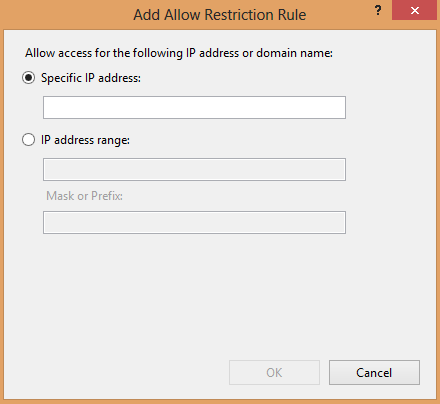


Figure 1.3 Host Restrictions

**MIME-Type Configuration**

MIME prevents undefined file types from being hosted by IIS, protecting the web server by shielding vicious attackers from downloading sensitive files. Although configuration files and data files would not typically be stored within the web root folder, when a hacker tries to download a file that does not have a defined type, a 404.3 HTTP status is issued by IIS, as well as recorded in the log file. If you have a custom file extension that needs to be downloadable by clients, you will need to add a new MIME type as follows.

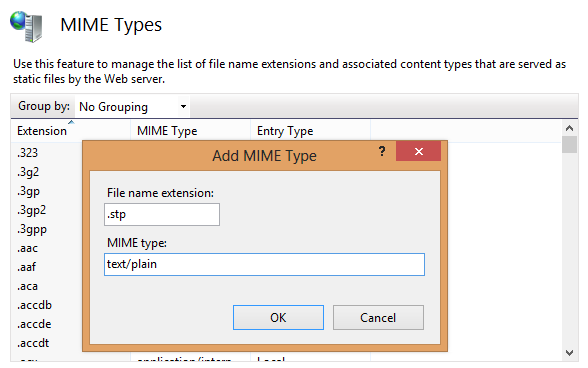


Figure 1.4 Adding new MIME type

**Request Filtering**

Request filtering enables a configurable set of rules that allows you to determine which type of request should be allowed or denied for the web site and web server. It implements functionality that allows for requests for a specific namespace and is tightly integrated with IIS logging functionality. It also offers request filtering by HTTP, file extension, request limit, and URL sequence.

In HTTP filtering, the verbs section allows or disallows requests that use specific HTTP verbs like GET, POST, etc. You can use the following XML tags within the configuration file.

1. **<security>**
2. **<requestFiltering>**
3. **<verbs** allowUnlisted**=**”false” **>**
4. **<add** verb**=**”GET” allowed**=**”true” **/>**
5. **</verbs>**
6. **</requestFiltering>**
7. </security >

Filtering on behalf of a file that is not matched by specific rules can be set to allow or reject. If request is rejected, a 404.7 HTTP status is logged. You can use the following XML tags within the configuration file.

1. **<security>**
2. **<requestFiltering>**
3. **<fileExtensions** allowUnlisted**=**”false” **>**
4. **<add** fileExtension**=**”.asp” allowed**=**”true” **/>**
5. **</fileExtensions>**
6. </ requestFiltering>
7. </security >

The request limit option enables you to restrict the size of requests made by clients, which prevents malicious requests. This setting could be achieved as:

1. **<security>**
2. **<requestFiltering>**
3. **<requestLimits**
4. maxAllowedContentLength**=**”200000000”
5. maxUrl**=**”30” **/>**
6. </requestLimits >
7. </ requestFiltering>
8. </security >

Finally, the filtering by URL sequence enables to reject requesst that contain some malicious substring like Directory traversal (“. .” ) in the requested URL. You can explicitly configure the sequence to be rejected, as shown below.

1. **<security>**
2. **<requestFiltering>**
3. **<denyUrlSequence>**
4. **<add** sequence**=**”. .” **/>**
5. </denyUrlSequence >
6. </ requestFiltering>
7. </security >

**Application Pool Configuration**

Application pool configuration is quite useful when a similar administrator needs to accept content from external parties, because it is difficult to isolate web application pools from each other. In other words, if multiple application pools run as a same identity, then code running inside one pool will clash with other application pool file systems. Therefore, IIS prevent this conflict to some extent by introducing application pool configuration. Each application pool has its own configuration file stored in the %systemdriver\inetpub\temp\appPools folder, as well as having an additional security identifier (SID) which is injected into the relevant w3wp.exe process. Later on, each process has its own SID and each pool’s configuration file is ACLed to a different SID.

The multiple application pools allow complete isolation between them to ensure that a malicious site cannot harm the other site in a server environment where more than one site is being deployed. If one website is compromised, the hacker can’t affect the other site on the server.

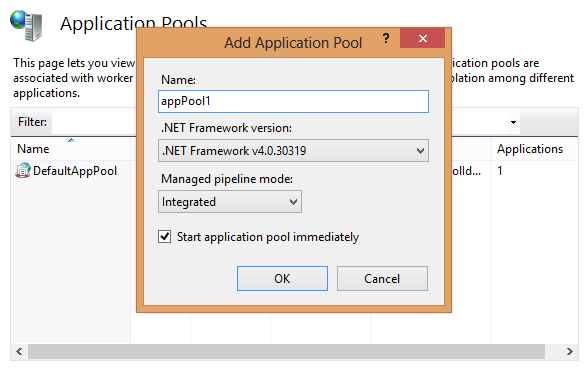


Figure 1.5 Adding app pool

**ISAPI Configuration**

The ISAPI extension provides additional functionality when a particular type of file is requested. Developers often write custom API extensions to extend the functionality of a server. From a knowledge point of view, PHP websites can be hosted in IIS via custom PHP ISAPI extension. Hence, ASP.NET websites can host in IIS because .aspx files are by default mapped to ASP.NET ISAPI extension. When a file with .aspx extension is requested by a client, processing is immediately handed over to the IIS ISAPI extension, which determines what additional work should be done.

IIS introduces the server side administration to configure what ISAPI extension is permitted to run on the server. IIS allows each ISAPI extension to be set to an allowed or disallowed state. Further, it logs and generates a 404.2 HTTP status for a disallowed malicious extension, because hackers can implant malicious viruses into the server remote file intrusion attack. Configuring additional ISAPI extensions requires taking further steps to ensure the ongoing security of your server. To add a new ISAPI extension, enter the DLL path and custom name as follows.

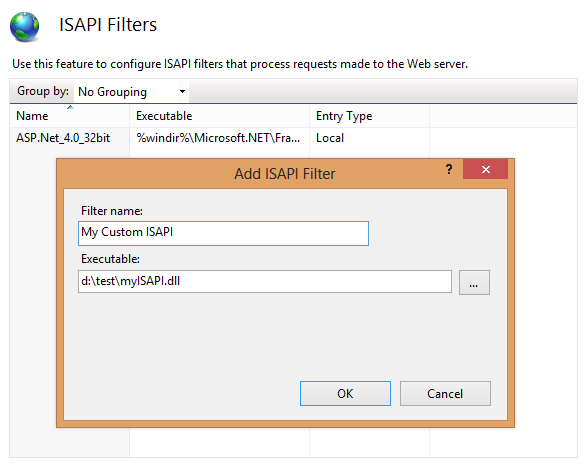


Figure 1.6 Adding ISAPI

**Logging Configuration**

Hackers often perform denial of service attacks to flood the server with massive requests which could result in the log files growing very large. Logs offer the proof of breach and aid to identify the approach by which the hacker gained illegitimate access. Administrators therefore perform periodical logging and auditing operations to detect possible traces of malicious activity over the web server. However, the Windows operating system has also the in-built Windows Events Logs feature where important information is logged, including logging time, password guessing attempts, etc., which indicate a possible sign of attack. Logging can be configured on a per-site basis with W3C, which writes log entries using a text-customizable ASCII format.

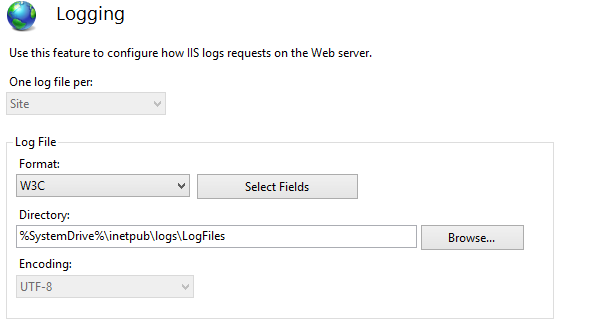


Figure 1.7 W3C logging

Hence, a large number of failed logon events can delineate an attempt to brute-force the password for an account. So, IIS introduces the provision of logging and auditing for both successful and unsuccessful alert attempts to the administrator to detect abnormal activity.

**Errors Pages Configuration**

Malicious hackers can obtain sensitive information about the web server and application while encountering an error. If the website is unable to handle an unexpected fatal error, it is more prone to exploit, because the default HTTP status displays along with other information which enchants the hackers*.*For example, take a look at the generated error page thrown by the server. Here one can easily obtain a bunch of critical information like user name, password, server IP, database structure, etc.

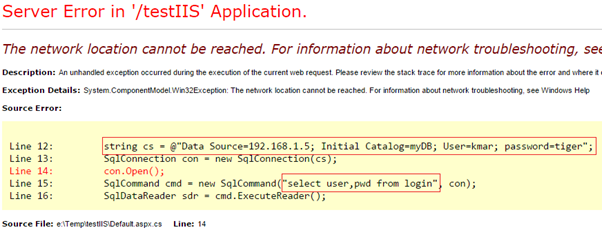


Figure 1.8 HTTP Error

So, it is a bad programming practice not to handle errors in the form of HTTP status code. Instead, there should be custom error pages in case of any error occurrence, so that sensitive information could be hidden. Custom error pages can be set for each HTTP status code. Therefore, IIS has a convenient method of displaying a different error to end-users than to administrators and developers, as follows.

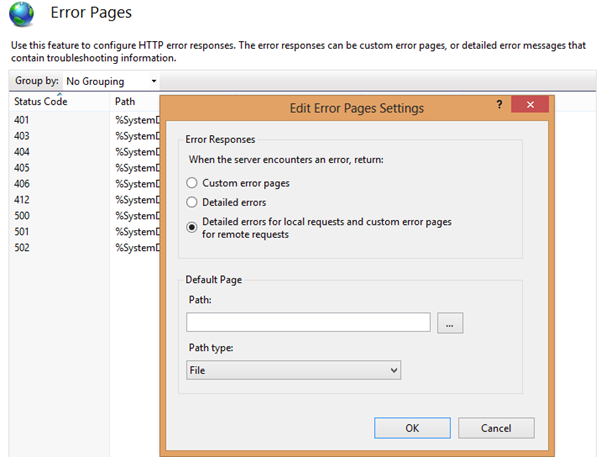


Figure 1.9 Error Page configuration

**FINAL WORDS**

Even after duly securing the working environment and IIS itself, it is important not to neglect application security. Recently, more attention has been devoted to breaking applications rather server software itself. A wide array of vulnerabilities can exploit application software, for example, SQL injection, cross site scripting, session replay, RFI, and many more attacks are prevalent. Attackers are becoming more sophisticated day by day, having advanced tools and tactics at their disposal. Hence, the perimeter must be secure from all fronts.

In this article, we have examined the requirements and mechanisms for securing ASP.NET websites by configuring essential parameters of IIS including HTTPS, authenticating end-users, client certificates, IP address restriction, and many more. Overall, this article serves as a primer to IIS security and various initiatives that administrators and security officers can refer to when enabling or restricting functionality on an IIS. Therefore, you should now have a comprehensive understanding of diverse security options, as well as the security-related mechanisms that concern IIS.