构建一个定制的IIS 7.0 服务

<http://www.iis.net/learn/get-started/planning-for-security/build-a-custom-iis-server>

介绍

IIS6.0及之前的版本，在服务本身的内部实现了许多广为使用的功能。与之相反，IIS 7.0 以及高版本的WEB服务引擎提供了一个模块结构，基于此，所有的服务特性都作为外挂组件提供。

* 能够有选择性地起用必要的服务特性。关闭不必要的特性以减少服务器的被攻击面/内存印迹（MemoryPrint）。
* 能够用第三方或自定义的实现来覆盖每一个特性。
* 能够基于其在服务拓扑中的角色，详细定义一个服务。
* 能够对服务的功能集进行良好的粒度及应用程序授权方面的高级控制。

这些服务组件，又称为服务模块，在应用程序池工作进程初始化时被加载，并且提供对请求进行处理的服务。每一个IIS7.0及以上版本的应用，都是由该应用中可用的模块提供的服务和这些服务所关联的内容所构成的组合。这些服务提供由模块扮演的两种主要角色。

* 提供请求服务。比如授权以及输出缓存（类似于IIS6中的ISAPI过滤器）。
* 提供请求处理句柄，比如静态文件处理句柄、CGI、ASP.NET页面处理。（类似于IIS6中的ISAPI扩展）

通过激活模块，服务可以被配置为提供服务器上的应用所想要的服务。

本文所要描述的任务有：

* 审视服务配置，以及一系列服务默认加载的模块。
* 移除所有模块以使服务达到其最小化的配置，然后在footprint上查看其作用。
* 通过增加模块建立自己的服务，以支持特定的场景。

本文包含

* 审视服务配置
* 检查服务FootPrint
* 剥光服务
* 定制服务
* 总结

检查默认的模块配置

主要的服务配置包含在applicationHost.config文件中，存放在IIS配置文件夹下（%windir%\system32\inetsrv\config\）。我们看一下包含在<system.webServer>节点组中的配置。

<globalModules>节点，它包含了服务级别的模块，由服务工作进程加载，并且与实现它们功能的本地DLL关联。

<modules>节点，它包含应用级别的模块，由特定的应用使用。这个节点用来选择一个应用应当加载的模块的子集，以及加载其它应用级别的模块。

<handlers>节点，它包含URL级别的句柄映射，用来引导请求到特定的模块去处理。这类似于IIS6的scriptmaps，或者ASP.NET，并且提供一个联合的请求映射到本地或托管的处理模块。

IIS模块的完整描述：<http://www.iis.net/default.aspx?tabid=2&subtabid=25&i=930>

创建一个配置文件的备份

首先，我们备份配置文件，以便在需要时恢复它。以管理员身份运行下述命令行命令：

%windir%\system32\inetsrv\appcmd add backup initial

我们可以以管理员身份运行下述命令行命令来恢复服务配置到初始状态。

%windir%\system32\inetsrv\appcmd restore backup initial

检查默认的模块列表

进入<system.webServer>/<globalModules>节点。这个节点，只能在服务级别下才能配置，包含由每一个服务工作进程加载的模块。每一个入口配置一个模块，模块带有一个名字和实现这个模块功能的DLL。

<globalModules>

<!--several modules omitted -->

<add name="BasicAuthenticationModule" image="…\authbas.dll" />

<add name="WindowsAuthenticationModule" image="…\authsspi.dll" />

</globalModules>

进入到<system.webServer>/<modules>节点。这个节点，既可以在服务级别下配置，也可以在应用级别下配置，定义了<globalModules>节点下加载的模块中哪些可以由应用使用。大部分时候，我们看到这个节点列出的模块的名称，适用于所有应用。

注意：在列举的最后，有一些额外的项目，它们是由ASP.NET扩展模式开发的托管模块。可以通过这个URL学习如何构造托管模块：<http://www.iis.net/default.aspx?tabid=2&subtabid=25&i=942>

进入到<system.webServer>/<handlers>节点。这个节点，可以在服务级别、应用级别、URL级别进行配置，它定义了请求是怎么被处理的。模块参与每一个请求，而句柄只处理特定URL的请求。

压缩模块（compression module）是一个说明模块的和好的例子。压缩模块关注每一个响应，并且根据需要压缩它。ASP.NET页面句柄是一个比较好的例子，它只接收映射到它的请求，比如带有.aspx扩展名的请求，<handlers>节点定义了基于URL及动词的请求，和用来处理这个请求的句柄模块之间的映射关系。当然还有一些额外的信息来配置每一个映射，这方面本文不作介绍。

<handlers>

<!-- certain details omitted -->

<add name="CGI-exe" path="\*.exe" verb="\*" modules="CgiModule" ... />

<add name="ISAPI-dll" path="\*.dll" verb="\*" modules="IsapiModule" ... />

<add name="ASPClassic" path="\*.asp" verb="GET,HEAD,POST" modules="IsapiModule" ... />

</handlers>

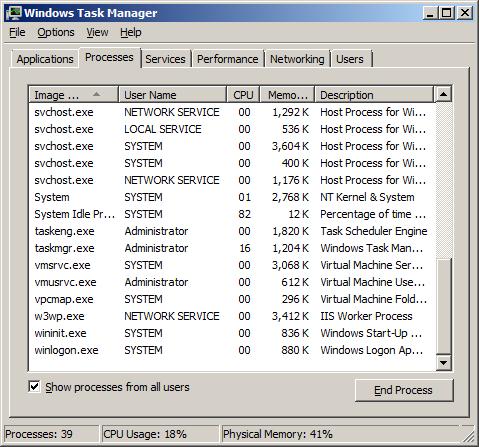
检查Server FootPrint

1. 打开浏览器，访问以下链接

<http://localhost/iisstart.htm>

这个启用应用服务程序池，并服务于iisstart.htm页面

1. 打开任务管理器，进入到“进程”属性页。因为IIS工作进程在不同的用户账号下运行，所以需要将“显示所有用户进程”勾上。注意w3wp.exe服务工作进程的大小。



1. 然后运行以下命令行命令

TASKLIST /fi "imagename eq w3wp.exe" /m

我们可以看到，有超过90个DLL被工作进程加载。它们中的大部分在…\intersrv\目录下，它们中的大部分都是我们刚才在<globalModules>节点下看到的模块DLLs，还有少数是用来支持.Net Framework以及服务本身所需要的。

剥光服务

在刚才的任务中，我们检查了由服务加载的默认的组件列表，大概有35个模块，提供从认证到静态文件服务在内的多种多样的服务。每个被服务加在的模块都对server footprint、受攻击范围、运行时效率、可用的特性集，产生影响。

在我们在下一个任务中构建自己的只包含所需的功能的服务之前，我们通过移除所有模块，运行一个空的服务，来构建一个速度快、体积小的，并且是安全的服务。

如果你的applicationHost.config有被改编过，你可以通过运行下述命令行命令来让它恢复到初始状态：%windir%\system32\inetsrv\appcmd restore backup initial。

接下来我们剥光服务：

1. 用文本编辑器打开%windir%\system32\inetsrv\config\applicationHost.config文件。
2. 找到<system.webServer>/<globalModules>节点。
3. 移除所有模块，使它成为一个空的节点。

<globalModules>

<!—Remove Everything -->

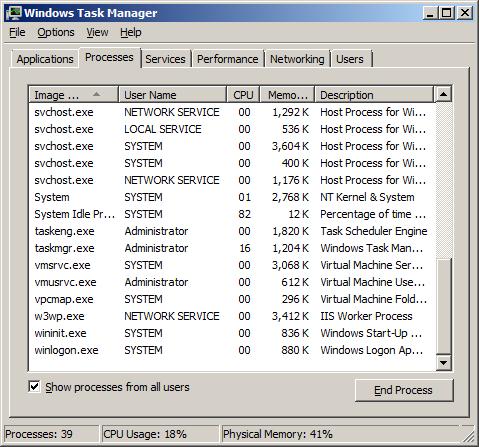
</globalModules>

1. 同样道理，移除<system.webServer>/<modules>下面的所有项目。
2. 同样道理，移除<system.webServer>/<handlers>下面的所有项目。

检查剥光了的服务的footprint

到这儿为止，我们已经构建了一个剥光的服务。让我们再来重复刚才的步骤来检查server footprint。

1. 打开任务管理器，进入到“进程”属性页。因为IIS工作进程在不同的用户账号下运行，所以需要将“显示所有用户进程”勾上。注意w3wp.exe服务工作进程的大小。



1. 然后运行以下命令行命令

TASKLIST /fi "imagename eq w3wp.exe" /m

与刚才的90个DLLs相比，现在只加载了大约50个DLLs。 – this indicates that the server did not load any of the module DLLs, which directly and indirectly accounted for the DLL count difference. Not only are the services disabled on the server, but also no code for these features is even loaded in the process. After optimization, the DLL count of the empty server will be significantly lower.

建立定制的服务

在上面的任务中，我们通过剥光服务使之达到最小化，仅保留了核心服务引擎，未加载任何额外的模块。现在，我们来构建自己的服务，用来作为网络文件服务。我们允许服务仅提供以下服务：

* 提供静态文件服务
* 提供文件夹目录列表服务
* 提供basic authentication和URL-based authorization规则的内容保护

提供静态文件服务

为了实现这个任务，我们假定已经按照上述步骤剥光了服务，移除了所有模块。在这个状态下，对于所有请求，服务总是返回401错误响应，因为没有任何模块被加载来提供任何形式的请求处理。

1. 用文件编辑器打开%windir%\system32\inetsrv\config\applicationHost.config
2. 找到<system.webServer>/<globalModules>节点。增加两个模块，一个是提供静态文件服务的，一个是匿名认证模块，它为请求提供默认的认证令牌（token）。

<globalModules>

<add name="StaticFileModule" image="%windir%\System32\inetsrv\static.dll" />

<add name="AnonymousAuthenticationModule" image="%windir%\System32\inetsrv\authanon.dll" />

</globalModules>

1. 找到<system.webServer>/<modules>节点。激活静态文件处理句柄模块，和匿名认证模块。

<modules>

<add name="AnonymousAuthenticationModule" />

<add name="StaticFileModule" />

</modules>

1. 找到<system.webServer>/<handlers>节点，将静态文件处理句柄映射到所有文件请求上。

<handlers>

<add name="StaticFile" path="\*" verb="GET,HEAD" modules="StaticFileModule" resourceType="Either" requireAccess="Read"/>

</handlers>

1. 保存applicationHost.config文件。
2. 打开浏览器，访问以下URL：<http://localhost/iisstart.htm>

此时可以发现，服务器响应了这个文件。

1. 接下来，通过访问以下URL来请求目录列表：<http://localhost>

我们得到了一个空白的画面，因为当前没有句柄模块被加载，并且被映射来处理这个请求，所以服务器发送了空的响应（200 OK）。接下来我们将增加这个处理模块。

提供文件夹目录服务

为了实现这个任务，我们假定已经按照上述步骤剥光了服务，移除了所有模块，然后增加了静态文件处理能力。

1. 用文件编辑器打开%windir%\system32\inetsrv\config\applicationHost.config
2. 找到<system.webServer>/<globalModules>节点，增加一个DirectoryListingModule以便增加浏览目录的能力。然后激活模块。

<globalModules>

<add name="AnonymousAuthenticationModule" image="%windir%\system32\inetsrv\authanon.dll" />

<add name="StaticFileModule" image="%windir%\system32\inetsrv\static.dll" />

<add name="DirectoryListingModule" image="%windir%\System32\inetsrv\dirlist.dll" />

</globalModules>

<modules>

<add name="AnonymousAuthenticationModule" />

<add name="StaticFileModule" />

<add name="DirectoryListingModule" />

</modules>

<handlers>

<add name="StaticFile" path="\*" verb="GET,HEAD" modules="StaticFileModule,DirectoryListingModule" resourceType="Either" requireAccess="Read" />

</handlers>

此时，我们已经添加了目录浏览服务能力。然而，处于从安全性方面考虑的原因，这个特性需要额外的配置，来控制是否允许浏览目录列表。这个配置被定义在<system.webServer>/<directoryBrowse>节点下，

1. 更改这个入口为：<directoryBrowse enabled="true" />
2. 保存applicationHost.config文件。
3. 打开浏览器，访问以下URL：<http://localhost/iisstart.htm>

可以发现，服务成功地列出了请求的目录。

使用URL认证保护资源

为了实现这个任务，我们假定已经按照上述步骤剥光了服务，移除了所有模块，然后增加了静态文件处理功能和目录浏览功能。

1. 用文件编辑器打开%windir%\system32\inetsrv\config\applicationHost.config。
2. 这次我们增加两个模块

* 基本认证模块，它支持基本认证方案，依靠http1.1和windows证书。
* URL认证模块，它支持用户与角色规则，基于访问控制。

1. 像以前一样，我们往<system.webServer>/<globalModules>节点和<system.webServer>/<modules>节点中增加相应的内容。

注意：这次我们不向<system.webServer>/<handlers>节点中增加任何内容，因为这些模块不提供请求句柄处理，他们只提供面向所有请求的请求服务。

<globalModules>  
 <add name="AnonymousAuthenticationModule" image="%windir%\system32\inetsrv\authanon.dll" />   
 <add name="StaticFileModule" image="%windir%\system32\inetsrv\static.dll" />   
 <add name="DirectoryListingModule" image="%windir%\system32\inetsrv\dirlist.dll" />   
 **<add name="UrlAuthorizationModule" image="%windir%\System32\inetsrv\urlauthz.dll" />**

**<add name="BasicAuthenticationModule" image="%windir%\System32\inetsrv\authbas.dll" />**   
</globalModules>   
   
  
<modules>   
 <add name="AnonymousAuthenticationModule" />   
 <add name="StaticFileModule" />   
 <add name="DirectoryListingModule" />   
 <add name="BasicAuthenticationModule" />   
 <add name="UrlAuthorizationModule" />   
</modules>

1. 允许Basic Authentication服务。

找到<basicAuthentication>节点，并且将属性设置为true。

<basicAuthentication enabled="true" />

1. 无效化匿名访问。找到<anonymousAuthentication>节点，设置属性为false。