A NGSSoftware Insight Security Research Publication

# Assessing IIS Configuration Remotely

(Low Level IIS Application Assessment)

**David Litchfield**
**(david@ngssoftware.com)**
**28th February 2002**
**www.ngssoftware.com**

**Contents**



## Introduction

A good application security assessment should probe all levels of the environment as well as the custom application itself. In terms of what can be exploited to leverage the greatest access it is of course the application itself, whether through SQL Injection or arbitrary command execution or file access, but defence in depth is by far the best stance to take. As such, this document will look at the relatively unsung skill of assessing the in-depth configuration of a Microsoft IIS web server remotely, showing how to "read" server responses to do this. Hopefully this paper will show how to use two seemingly disparate pieces of information to help determine an attack and therefore assess the associated risk. We will examine ways to determine what permissions have been set on virtual directories used by the application, what authentication options have been left enabled or disabled, what default application extension mappings have been left in place, how to gather information about the server, in terms of physical directory structure, internal IP addresses, computer and Windows NT domain name, anonymous Internet account names and more. Some of this information is relatively old, some fresh, straight out of the NGSSoftware research labs and when combined with a good security analysis of the application code or logic will be able to produce a killer app assessment.

This paper assumes a working knowledge of Microsoft's IIS and HTTP.

## Directory Permissions

As far as IIS is concerned directory permissions, speaks not about NTFS security, but what actions a given resource can perform on the web server and what a web client may do with said resource.

The IIS Directory Permissions are as follows

| | |
|---|---|
| Read | Users may read static resources |
| Write | Users may create and delete resources |
| Execute | Resources (e.g. DLLs and EXEs) may execute |
| Script | Application scripts (e.g. ASP) may execute |
| Directory Browsing | Users may browse for files |

This section will describe how to determine if a given permission is enabled or disabled. Some of this may seem self-explanatory but when applied to a specific application you will be able to determine if the current permissions are too lax and advise accordingly.

**Execute**
Request http://iis-server/dir/no-such-file.dll. If the server responds with a 500 Internal Server error response with a message that it could not find the specified module then the execute permission is enabled. Questions one should be asking is does this virtual directory need execute permission? If not then it should be removed because it presents an attacker with another possible location to attempt arbitrary command execution attacks with directory traversal such as the double-url decoding and UNICODE attacks. If the server responds with a 404 File not found (*) then execute permission is not enabled. (* See Read permissions).

**Write**
Being able to write files out to a virtual directory on the web server is most definitely not a good thing. To test if write permission is enabled for anonymous web clients telnet into the web server port usually TCP port 80 and make the following request:

```
PUT /dir/my_file.txt HTTP/1.1
Host: iis-server
Content-Length: 10 <enter><enter>
```

At this stage the server should respond with a 100 Continue message.

```
HTTP/1.1 100 Continue
Server: Microsoft-IIS/5.0
Date: Thu, 28 Feb 2002 15:56:00 GMT
```

On receiving this type 10 letters

```
AAAAAAAAAA

HTTP/1.1 201 Created
Server: Microsoft-IIS/5.0
Date: Thu, 28 Feb 2002 15:56:08 GMT
Location: http://iis-server/dir/my_file.txt
Content-Length: 0
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, COPY, MOVE, PROPFIND,
PROPPATCH, SEARCH, LOCK, UNLOCK
```

If the server responds with this 201 Created response then the write permission is enabled. If you're asked to authenticate (401 Access Denied) then the directory still has write permissions but the anonymous internet account doesn't have the rights to create files as far as NTFS security goes. If the server were to respond with a 403 Forbidden then the write permission is not enabled.

People may wonder how likely it is to find a production server with the write permission enabled. In the authors experience more often than one would expect (or like.) Often sites allow uploads of files such as images, resumes, etc. This is normally handled by the application but misunderstanding how IIS permissions work, developers or administrators wrongly assume that if you're uploading to a directory then IIS write permission must be enabled and do so. If the application handles the upload process then write permission does not need to be enabled. If you must allow users to write to a virtual directory using the PUT HTTP request method then remove scripting and execute permissions from the directory. Doing this will prevent malicious users from uploading nefarious scripts or executables. If the uploaded information is of a personal or private

nature then also consider removing the read permission, too. This way users may upload but not have others access what they have just deposited.

**Script**
Too often directories are given the script permission when it plainly isn't needed. Why would the /images virtual directory with only static gifs and jpegs ever need scripting permissions? Better to work to the least privilege model. To test a directory to see if the script permission is enabled request

```
http://iis-server/dir/no-such-file.asp
```

If the server responds with a 404 File not found message it is enabled. If a 403 Forbidden is returned it is not. If the directory doesn't need it disable it.

**Directory Browsing.**
This permission can be one of the more tricky to check to see if it has been enabled for a directory. That said it can also be one of the most easy to test for and it all depends upon whether the default served file exists (e.g. default.asp). If it doesn't, just request the directory name.

```
http://iis-server/dir/
```

If a 200 response is returned and an index of the files then directory browsing is allowed. If, however, default.asp existed in the directory, even though directory browsing is enabled this page and not the index is returned. So how does one by-pass this then? To answer that, we need to look to WebDAV. WebDAV is Web distributed authoring and versioning use to turn HTTP and web server's into a file system.

One of the request methods that came with WebDAV is PROPFIND. PROPFIND allows you to search for specific properties about resources on the web server such as file name, creation date, last modified etc. Using PROPFIND bypasses this default.asp issue if directory browsing is enabled.

Telnet into the web server and request

```
PROPFIND /dir/ HTTP/1.1
Host: iis-server
Content-Length: 0
```

Note the Content-Length header is required even though there is no content and hence the length is 0.

If the Directory Browsing permission has been enabled a list of all resources and their properties under this directory will be returned with a 207 Multi Status response. If Directory Browsing is not enabled then the same 207 response is returned but with considerably less information.

Allowing Directory Browsing presents what is generally considered as a low risk vulnerability but in the author's opinion the risk needs to be assessed on a basis-by-basis fashion. If the directory contained nothing but images then, sure, this is low risk, perhaps not even a risk. If however an attacker were able to get a directory listing that reveal secret_admin_page.asp or dbconnectinfo.inc then the risk is considerably greater. (By the way, if you or your developers insist in using the .inc extension either place these files in a directory with read permission disabled or associate the .inc extension with asp.dll - or do both. Doing this prevents attackers gaining access to the business/application logic stored in these include files.)

**Read**
Simply request a file with a .txt file extension

```
http://iis-server/dir/no-such-file.txt
```

If the server responds with a 404 File Not Found then the Read permission is enabled. Why should this be a problem? For those that remember the ::$DATA or the variations of the period append attacks, disallowing read access would have prevented attackers gaining access to ASP source code. Most applications running on IIS use ASP and the only permission ASP needs is the script permission. For sites with both an ASP application and static content (and this includes images) it is best to separate scripts and static resources into different directories. Enable only the script permission for the directory holding the ASP content and enable on the read permission for those directories holding static content.

Going back to the request the server would respond with a 403 Forbidden message with the more verbose response of "Read permission disallowed".

## IIS Authentication

One of the little known facts about IIS is that if a web client makes a request to the server and presents authentication credentials, even without having been asked for them, the server will go ahead and attempt to authenticate them. If unsuccessful, the server will respond for the user to reattempt the auth process. This means that even if the resource can be accessed anonymously the server can still be forced to perform authentication. There are two things with this. One it opens up the server to brute force attacks and the other is that when remotely assessing the IIS server's configuration this can be used to work out what types of auth processes have been enabled or disabled on the server. As we shall see, if IIS authentication is enabled then it can be used to gather useful information about the server.

By default IIS servers allow Anonymous access, and authentication using Basic Auth and NTLM (integrated Windows Authentication). To work out if these defaults have been changed make the following requests.

Telnet into port 80 and request

```
GET /default.asp HTTP/1.1
Host: iis-server
Authorization: Basic c3lzdGVtOm1hbmFnZXIA
```

This makes a request and presents a base 64 encoded user ID and password as credentials. This is known as Basic Auth. (For those interested the base 64 encoded string decodes to system:manager.) On receipt of this request there's one of two things that can happen. The server returns a 401 Access Denied response which means that Basic Auth is enabled or the server returns a 200 response (assuming of course anonymous access is allowed to default.asp and it exists!). This 200 response means either that there is a user account on the web server called "system" and it has a password of "manager" *or* the server is configured not to use Basic Auth. The latter is much more likely and this is how to tell if Basic Auth is allowable.

To work out if NTLM authentication is enabled make the following request.

```
GET / HTTP/1.1
Host: iis-server
Authorization: NegotiateTlRMTVNTUAABAAAAB4IAoAAAAAAAAAAAAAAAAAAAAA=
```

Again the same logic can be applied. A 200 response tells us that NTLM is not allowed but a 401 tells us that it is. If both requests elicit a 200 response then the server does not support any authentication methods and simply allows anonymous access. For most public websites this is the best (i.e. most secure) option - turn off both Basic and NTLM . As we'll shortly see we can use these auth procedures to have the server leak sensitive "inside" information.

**Information Leakage**
We've just looked at how to work out if authentication is enabled on a server and how to force the authentication process even if anonymous access is allowed to the resource. Continuing on the same thread we can use forced authentication to start gathering useful information.

**Obtaining the IP address**
For IIS servers behind a firewall that uses Network Address Translation (NAT) often a private internal IP addressing scheme may be used such as 10.x.x.x.

If the server allows Basic Auth then we can discover this internal IP address. Simply present an authentication attempt with a blank host header field.

```
GET / HTTP/1.1
Host:
Authorization: Basic c3lzdGVtOm1hbmFnZXIA
```

The server will respond with a 401 Access Denied message but look at the authentication "Realm":

```
HTTP/1.1 401 Access Denied
Server: Microsoft-IIS/5.0
Date: Fri, 01 Mar 2002 15:45:32 GMT
WWW-Authenticate: Basic realm="10.1.1.2"
Connection: close
Content-Length: 3245
Content-Type: text/html
```

Normally the Realm is set to whatever the client supplies as a Host header but as it is blank in this case the IIS machine needs to set a realm so it chooses the machine's IP address. As it turns out we can also use the PROPFIND, WRITE, MKCOL methods and others to do the same thing:

```
PROPFIND / HTTP/1.1
Host:
Content-Length: 0
```

If the server has been configured to use its hostname then the IP address won't be returned but rather its NetBIOS name. The NetBIOS name itself is of course useful information.

But can we get more than just the NetBIOS name of the server – for example the Domain name? We can – using authentication again but this time NTLM. Force authentication with the following request

```
GET / HTTP/1.1
Host: iis-server
Authorization: NegotiateTlRMTVNTUAABAAAAB4IAoAAAAAAAAAAAAAAAAAAAAA
```

The server will reply with

```
HTTP/1.1 401 Access Denied
Server: Microsoft-IIS/5.0
Date: Fri, 01 Mar 2002 16:24:58 GMT
WWW-Authenticate: Negotiate TlRMTVNTUAACAAAADAAMADAAAAAFgoKgeGvyVuvy67U
AAAAAAAAAAEQARAA8AAAAUwBDAFkATABMAEEAAgAMAFMAQwBZAEwATABBAAEA
DABTAEMAWQBMAEwAQQAEAAwAUwBDAFkATABMAEEAAwAMAFMAQwBZAEwATABB
AAAAAAA=
Content-Length: 3245
Content-Type: text/html
```

The text under the WWW-Authenticate header is base 64 encoded text and contains the machine name and Windows NT domain name.

**Default Extension Mappings**

When installed IIS has many application environments available such as ASP and Internet Data Queries. Best practices dictate that all unnecessary application mappings be disabled. If this practice had been followed by the majority of system administrators then the success of worms such as Code Red would have been drastically hampered. As part of the low level app assessment it needs to be determined if any of the defaults not being used have been left intact. What follows is a list of extension mappings and the expected response when requesting a non-existent file and the application extension mapping still exists.

| | |
|---|---|
| Extension | .printer |
| Request | http://iis-server/foo.printer |
| Response code | 500 Internal Server Error (13) |
| Text | Error in web printer install |

| | |
|---|---|
| Extension | .idc |
| Request | http://iis-server/foo.idc |
| Response code | 500 Internal Server Error |
| Text | Error performing query |

| | |
|---|---|
| Extension | .idq |
| Request | http://iis-server/foo.idq |
| Response code | 200 OK |
| Text | The IDQ file foo.idq could not be found |

| | |
|---|---|
| Extension | .ida |
| Request | http://iis-server/foo.ida |
| Response code | 200 OK |
| Text | The IDQ file foo.idq could not be found |

| | |
|---|---|
| Extension | .htr |
| Request | http://iis-server/foo.htr |
| Response code | 404 or Reset |
| Text | <html>Error: The requested file could not be found.</html> |

| | |
|---|---|
| Extension | .htw |
| Request | http://iis-server/foo.htw |
| Response code | 200 OK |
| Text | The format of QUERY_STRING is invalid. |

| | |
|---|---|
| Extension | .stm |
| Request | http://iis-server/foo.stm |
| Response code | 404 |
| Text | <body><h1>404 Object Not Found</h1></body> |

| | |
|---|---|
| Extension | .shtm |
| Request | http://iis-server/foo.shtm |
| Response code | 404 |
| Text | <body><h1>404 Object Not Found</h1></body> |

| | |
|---|---|
| Extension | .shtml |
| Request | http://iis-server/foo.ida |
| Response code | 404 |
| Text | <body><h1>404 Object Not Found</h1></body> |

Not that some return a 404 response, namely .htr, .stm, .shtm and .shtml. One would think that this makes it more difficult to ascertain if the extension is still mapped. You can tell simply because the 404 response is different from the standard 404 response. This way you can work out that there must be a different bit of code handling the response than the norm and from that deduce the extension is still mapped. Notice the difference between a request for a .htr resource and a .stm resource. Both return 404, file not found, responses but they differ in terms of text. The reason the .stm, .shtm and .shtml all return the same text is that the all are mapped to ssinc.dll hence it is the same bit of common code handling the response.

**When a 200 response doesn't quite mean 200**

According to the HTTP rfc web servers should only return a 200 response when the request has been satisfied properly. Often though many custom applications will return a 200 response to requests that really should have generated a 404 file not found or 500 internal server error. This is one of the great banes of security assessment scanners. For every request made a 200 response is returned so a long list of false positives is returned and the *real* results hidden in amongst them. Is there anything that can be done to help limit these false positives? Using methods like PROPFIND you can prevent these false positives.

**IIS on Server or Professional/Workstation**

Whilst it's not common to find a public web server running on Windows 2000 Professional or Windows NT Workstation they are often found in supporting roles. It can be useful to work out which you're dealing with – i.e. Server or Workstation. This is simple – IIS running on a Workstation allows a maximum of 10 given connections at any one time. Open up 11 connections making a HTTP 1.1 request to hold open the connection. If the 11$^{th}$ returns a 403 Access Forbidden response then the chances are the server is running on workstation or professional. This number may be less if there are already active connections.

**Conclusion**

Hopefully this document has shown how to take simple information and infer details about the remote system. Whilst this represents only a small section of the application assessment spectrum it does form an important part in terms of producing a holistic report.