

# Forensic Log Parsing with Microsoft's LogParser

Mark Burnett 2003-07-18

Investigating a web-based intrusion can be a daunting task, especially when you have no information other than knowing it was web-based. It is easy to waste precious time digging through megabytes, perhaps even gigabytes, of log files trying to locate suspicious activity. Often this search turns up little useful evidence.

Consider this scenario: an e-commerce site receives several reports from customers about unauthorized orders on their accounts. They suspect that someone has compromised their web-based ordering system so they gather the log files from several different IIS web servers. They have the dates and times of the orders, but the corresponding IP addresses in the log files turn out to be anonymous proxies used by the suspect. Searching for activity from those IP addresses in the log files turns up nothing. Browsing through the raw log files for those dates also turns up nothing. Somehow, someone found a flaw in the ordering system but he or she could have discovered the flaw months before exploiting it. Tracking down the flaw and IP addresses used by the suspect seems impossible. But there are techniques that can facilitate log file forensics. The purpose of this article is to demonstrate log file forensics of IIS logs using SQL queries with Microsoft's LogParser tool.

## IIS Log Fields

The first step is to prepare for security incidents by logging as much information as possible. IIS can log a significant amount of information about each web request, but many of the available log fields are not enabled by default. To enable full logging, open the Internet Services Manager and edit the Extended Logging Properties to include all available log fields. Much of this information has some forensics value as shown in Table 1.

**Table 1: IIS Log Fields**

Field Name	Description	Uses
Date (date)	The date of the request.	Event correlation.
Time (time)	The UTC time of the request.	Event correlation, determine time zone, identify scanning scripts.
Client IP Address (c-ip)	The IP address of the client or proxy that sent the request.	Identify user or proxy server.
User Name (cs-username)	The user name used to authenticate to the resource.	Identify compromised user passwords.

Service Name (s-sitename)	The W3SVC instance number of the site accessed.	Can verify the site accessed if the log files are later moved from the system.
Server Name (s-computername)	The Windows host name assigned to the system that generated the log entry.	Can verify the server accessed if the log files are later moved from the system.
Server IP Address (s-ip)	The IP address that received the request.	Can verify the IP address accessed if the log files are later moved from the system or if the server is moved to a new location.
Server Port (s-port)	The TCP port that received the request.	To verify the port when correlating with other types of log files.
Method (cs-method)	The HTTP method used by the client.	Can help track down abuse of scripts or executables.
URI Stem (cs-uri-stem)	The resource accessed on the server.	Can identify attack vectors.
URI Query (cs-uri-query)	The contents of the query string portion of the URI.	Can identify injection of malicious data.
Protocol Status (sc-status)	The result code sent to the client.	Can identify CGI scans, SQL injection and other intrusions.
Win32 Status (sc-win32-status)	The Win32 error code produced by the request.	Can help identify script abuse.
Bytes Sent (sc-bytes)	The number of bytes sent to the client.	Can help identify unusual traffic from a single script.
Bytes Received (cs-bytes)	The number of bytes received from the client.	Can help identify unusual traffic to a single script.
Time Taken (time-taken)	The amount of server time, in milliseconds, taken to process the request.	Can identify unusual activity from a single script.
Protocol Version (cs-version)	The HTTP protocol version supplied by the client.	Can help identify older scripts or browsers.

Host (cs-host)	The contents of the HTTP Host header sent by the client.	Can determine if the user browsed to the site by IP address or host name.
User Agent (cs(User-Agent))	The contents of the HTTP User-Agent header sent by the client.	Can help uniquely identify users or attack scripts.
Cookie (cs(Cookie))	The contents of the HTTP Cookie header sent by the client.	Can help uniquely identify users.
Referer (cs(Referer))	The contents of the HTTP Referer header sent by the client.	Can help identify the source of an attack or see if an attacker is using search engines to find vulnerable sites.

While I normally recommend logging all fields, the actual fields you choose to log should be based on a balance between forensics capabilities and disk space.

### Custom Logging

IIS does provide many log fields, but there may be other fields you wish to record. For example, if the request comes from a proxy server, you may want to see if the proxy server sends the client's real IP address through other HTTP headers. For example, some proxy servers add the "X-Forwarded-For" header containing the client's real IP address.

IIS has a limited capability to log custom fields through the `Response.AppendToLog` method. The limitation, however, is that a new field is not created in the log files, but this data is appended to the URI Query field. To distinguish the two values, you can separate them with a character such as the pipe ("|"). Below is example ASP code to log additional proxy headers:

```
<%
sHeader= Request.ServerVariables("X-Forwarded-For")
If Len(sHeader) Then Response.AppendToLog "|" & sHeader
%>
```

Note that other common proxy headers are `Forwarded`, `Client_IP`, `Remote_Addr`, `Remote_Host`, `Forwarded`, `VIA`, `HTTP_From`, `Remote_Host_Wp`, `Xonnection`, `Xroxy_Connection`, and `X_Locking`.

### Microsoft's LogParser Tool

Digging through logs requires that you have some common interface to perform queries across hundreds of individual log files. One method is to dump all the logs into an SQL database. Another solution is Microsoft's LogParser tool. This robust tool provides an SQL interface to a variety of log file formats and is fast enough for log file analysis of most web sites. I won't go into detail here about how to use LogParser, but the document included with the package is very helpful to get started. Because LogParser is a command-line tool, I have found it useful to either to copy the file to the C:\WINNT directory or to add the LogParser directory to your PATH variable.

You can download Microsoft's LogParser 2.0 [here](#), but the [IIS 6 Resource Kit](#) includes LogParser 2.1, which has some new features. Although LogParser 2.1 runs fine on a Win2k system, you cannot install the IIS 6 resource kit on Win2k. However, you can manually extract the resource kit files using the command: `iis6Orkt.exe /V/a`.

It is important to note that when doing any log file processing, be sure to work on copies of the logs to help preserve the integrity of the original files (see [Maintaining Credible Logfiles](#)). I also find it helpful to only copy those logs for the time period I want to analyze to reduce the size of the query results.

This article will demonstrate many of the forensic capabilities of LogParser. Keep in mind that I wrote each of these example queries for a typical configuration, therefore you may need to adjust them for your particular site. Not all queries listed here will be effective for you, depending on your site configuration and traffic level.

## Finding the Intrusion

If you do not know anything about the intruder or the nature of the intrusion, you must first do some high-level queries to know where to start your hunt. Most attacks leave some kind of trail or have some side-effect on your server. The trick is finding them.

## Trojan Files

Before we dig in to the actual log files, it may be useful to do a quick check of the newest files on the web site. If the intruder was able to create or modify files within the web content directories, he or she may have uploaded Trojan ASP scripts or executables. You might just get lucky and find these files. The following query lists the 20 newest files on the web site:

```
C:\>logparser -i:FS "SELECT TOP 20 Path, CreationTime from c:\inetpub\wwwroot\*. * ORDER BY
CreationTime DESC" -rtp:-1
```

Path	CreationTime
c:\inetpub\wwwroot\Default.asp	6/22/2003 6:00:01
c:\inetpub\wwwroot>About.asp	6/22/2003 6:00:00
c:\inetpub\wwwroot\global.asa	6/22/2003 6:00:00
c:\inetpub\wwwroot\Products.asp	6/22/2003 6:00:00

...

And this query lists the 20 most recently modified files:

```
C:\>logparser -i:FS "SELECT TOP 20 Path, LastWriteTime from c:\inetpub\wwwroot\*. * ORDER BY
LastWriteTime DESC" -rtp:-1
```

Path	LastWriteTime
c:\inetpub\wwwroot\Default.asp	6/22/2003 14:00:01
c:\inetpub\wwwroot>About.asp	6/22/2003 14:00:00
c:\inetpub\wwwroot\global.asa	6/22/2003 6:00:00
c:\inetpub\wwwroot\Products.asp	6/22/2003 6:00:00

...

But suppose the attacker was careful and deleted all Trojan files when finished. In that case, the files will not be exist but there will be log entries showing successful requests for those files. To identify these log entries you must make a list of all files on your site that have resulted in 200 HTTP status codes. From your log files directory, execute the following query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT TO_LOWERCASE(cs-uri-stem) AS URL,
Count(*) AS Hits FROM ex*.log WHERE sc-status=200 GROUP BY URL ORDER BY URL" -rtp:-1
```

URL	Hits
/About.asp	122
/Default.asp	9823
/downloads/setup.exe	701
/files.zip	1
/Products.asp	8341
/robots.txt	2830

...

Carefully review this list and make sure that each item listed is part of your web application. In particular, watch for files such as nc.exe, tini.exe, root.exe, cmd.exe, upload.asp, aspexec.asp, etc.

## Script Abuse

If searching for new or modified files turns up nothing, it is time to check out your scripts and executables. Any script or executable that accepts user input is a potential attack vector. Before starting, you should identify which executable file extensions you use in your web content areas. The following query will give you a report of all file extensions that exist within your web content (adjust the path names as necessary):

```
C:\>logparser -i:fs "SELECT TO_LOWERCASE(SUBSTR(Name, LAST_INDEX_OF(Name, '.'), STRLEN(Name)))
AS Extension, Count(*) as Files from c:\inetpub\wwwroot\*.*, c:\inetpub\scripts\*.* WHERE
Attributes NOT LIKE 'D%' GROUP BY Extension ORDER BY Files DESC" -rtp:-1
```

#### Extension Files

```
-----
.gif          704
.asp          180
.jpg          44
.css          43
.htm          28
.txt          21
.html         6
.dll          5
.zip          4
```

According to this list, the site contains several file extensions that may be of concern to us: .asp and .dll. Therefore, all the example queries from this point on will specifically look for ASP and DLL files. You will likely need to adjust this depending on which executable extensions you use on your web site.

One way to detect script abuse is to see if any one script has an unusually high number of hits. Since web-based attacks often require some trial and error, you should expect to see noticeable statistical variances, unless of course your web site gets millions of hits a day. Nevertheless, it is sometimes useful to see if any single day produced unusually high traffic.

The following query will show the number of hits for each day for each ASP and DLL file. From your log files directory, type the following:

```
C:\WINNT\System32\LogFiles\W3SVC1>LogParser "SELECT TO_STRING(TO_TIMESTAMP(date, time), 'yyyy-
MM-dd') AS Day, cs-uri-stem, COUNT(*) AS Total FROM ex*.log WHERE (sc-status<400 or sc-
status>=500) AND (TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' OR TO_LOWERCASE(cs-uri-stem) LIKE '%.
exe%') GROUP BY Day, cs-uri-stem ORDER BY cs-uri-stem, Day" -rtp:-1
```

Day	cs-uri-stem	Total
2003-04-01	/Default.asp	127
2003-04-02	/Default.asp	121
2003-04-03	/Default.asp	132
2003-04-04	/Default.asp	116
2003-04-05	/Default.asp	107
2003-04-06	/Default.asp	144

```

2003-04-07 /Default.asp          466
2003-04-08 /Default.asp          174
2003-04-09 /Default.asp          118
...

```

In the sample results above the number of hits on 2004-04-07 is suspiciously high and should be investigated further.

Another good attack indicator is the number of errors per hour. The following script returns the dates and hours that had more than 25 error codes returned. This value will likely need adjusting depending on how much traffic your site receives:

```

C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT date, QUANTIZE(time, 3600) AS hour, sc-
status, Count(*) AS Errors FROM ex03*.log WHERE sc-status>=400 GROUP BY date, hour, sc-status
HAVING Errors>25 ORDER BY Errors DESC" -rtp:-1

```

```

date          hour          sc-status  Errors
-----
2003-06-22  22:00:00  404        110
2003-04-21  13:00:00  404         36
2003-04-19  23:00:00  404         36
2003-04-19  13:00:00  404         27
...

```

Further investigation of the dates listed above may show that the high number of 404 errors are CGI scans looking for vulnerable scripts on your site. The 404 errors themselves are not as much of a concern as are the 200 results during that same time that may indicate a successful attack. This query will return all valid requests from any IP address that also had a 404 error on 2003-06-22:

```

C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT c-ip, cs-uri-stem, Count(*) as Hits FROM
ex*.log WHERE TO_LOWERCASE(cs-uri-stem) NOT LIKE '%.gif' AND TO_LOWERCASE(cs-uri-stem) NOT
LIKE '%.jpg%' AND c-ip IN (SELECT c-ip FROM ex030622.log WHERE sc-status=404) AND sc-
status=200 GROUP BY c-ip, cs-uri-stem" -rtp:-1

```

```

c-ip          cs-uri-stem          Hits
-----
199.154.189.199 /Default.asp          3
199.154.189.199 /main.css              3
199.154.189.199 /Products.asp          7
199.154.189.199 /About.asp             1
63.54.202.2    /Products.asp          18
63.54.202.2    /main.css              1

```

```
81.112.9.62      /Default.asp      1
```

```
...
```

Looking at these results, you can see two IP addresses that had an unusual number of hits on Products.asp. It could be that these were two different attackers or the same attacker who used two different proxies to conceal his or her IP address. One way to find out if they are likely the same person is to check the User-Agent header for the two different IP addresses:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT c-ip, cs(User-Agent) FROM
ex030622.log WHERE c-ip='198.54.202.2' or c-ip='62.135.71.223'" -rtp:-1
```

```
c-ip              cs(User-Agent)
-----
63.54.202.2      Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+Q312461;+.NET+CLR+1.0.3705
199.154.189.199 Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+Q312461;+.NET+CLR+1.0.3705)
```

This proves that those two IP addresses are either the same user or two different users with the exact same OS, browser, service pack level, installed hotfixes, and .NET installation. It is not a perfect indicator but it is significant. To support this evidence, you could go through your logs and discover when each of the IP addresses first hit your web site. When a user visits a web site for the first time, the browser downloads the page and any graphics and stores it all in the browser's temporary cache. This is so that subsequent visits to the page will not require downloading all the graphics again. However, the browser does check to see if the graphics have been modified before using the cached versions. If the graphic has not been modified, the server will return a 304 HTTP status code. Therefore, if you create a query for a specific IP address with a status code of 200 for any particular graphic, that log entry will be the user's first visit, providing they have not cleared their cache. So if a user switches to a different proxy server, the file will still be cached and therefore there will never be a first visit from one of the IP addresses. If one of the two IP addresses mentioned above turns up not having a first visit, chances are that they first visited the site from the other IP address. If neither IP address shows a 200 result, then there are more IP addresses left to discover.

## SQL Injection

If you read [this paper](#) (PDF) from [NGSSoftware](#) you will see that attacks such as SQL injection are based on sending faulty requests to a server and interpreting the error messages. Some of the indicators of this type of attack are:

- Numerous sequential hits from the same IP address to the same URL;
- High numbers of 500 HTTP status codes or other errors;
- GET requests to ASP pages that normally only receive POST requests; and
- Other clusters of anomalous web site activity.

It may also be useful to see an unusually high number of hits on a single page from a single IP address. The

following query shows any IP address that hit the same page more than 50 times in a single day:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT date, cs-uri-stem, c-ip, Count(*)
AS Hits FROM ex*.log GROUP BY date, c-ip, cs-uri-stem HAVING Hits>50 ORDER BY Hits Desc" -rtp:-
1
```

date	cs-uri-stem	c-ip	Hits
2003-05-19	/Products.asp	203.195.18.24	281
2003-06-22	/Products.asp	210.230.200.54	98
2003-06-05	/Products.asp	203.195.18.24	91
2003-05-07	/Default.asp	198.132.116.174	74
...			

Looking at these results, it is immediately obvious that one IP address hit the same page 281 times one day and 91 times another day, which is obviously suspicious.

Another useful technique is to view exactly what ASP errors IIS encountered while serving requests. Most attempts at breaking into a web site will inevitably result in some kind of error. The following query will return a list of every ASP error recorded in the log files:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-query, Count(*) AS Total FROM ex*.
log WHERE sc-status>=500 GROUP BY cs-uri-query ORDER BY Total DESC " -rtp:-1
```

cs-uri-query	Total
Out-of-process+ISAPI+extension+request+failed.	18
55 8000ffff Catastrophic_failure__	8
49 8000ffff Catastrophic_failure__	6
74 800a01c2 Wrong_number_of_arguments_or_invalid_property_assignment	1
...	

If you find any errors that are interesting, you could write another query to drill down to the specific error. In particular, you want to watch for ODBC and ADO errors, indicating a possible attempt at SQL injection.

Another way to identify errors is to look at the status codes returned by the server. If you want to see a detail of what status codes IIS returned for each page, try the following query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, sc-status, Count(*) AS Total
FROM ex*.log WHERE TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE
'%.exe%' GROUP BY cs-uri-stem, sc-status ORDER BY cs-uri-stem, sc-status" -rtp:-1
```

cs-uri-stem	sc-status	Total
/Default.asp	200	9258
/Default.asp	500	3
/MSOffice/cltreq.asp	404	12
/MailResult.asp	404	1
/asp/aspmail.asp	302	86
/asp/aspmail.asp	500	28
/autocomplete.asp	404	2
/awards.asp	404	4
...		

Also of interest are the Win32 Status codes, which may be attack indicators:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, WIN32_ERROR_DESCRIPTION(sc-win32-status) as Error, Count(*) AS Total FROM ex*.log WHERE sc-win32-status>0 and (TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%') GROUP BY cs-uri-stem, Error ORDER BY cs-uri-stem, Error" -rtp:-1
```

cs-uri-stem	Error	Total
/Default.asp	The RPC server is unavailable.	2
/Default.asp	The remote procedure call failed	1
/asp/aspmail.asp	The RPC server is unavailable.	12
/download/Default.asp	The RPC server is unavailable.	3
...		

Some ASP pages should only accept form input from previous pages. If, for example, you may have a page such as checkout1.asp that sends a POST request to checkout2.asp, then anything other than a POST request to checkout2.asp may be suspicious. This query will show what HTTP methods were sent to each page:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, cs-method, Count(*) AS Total FROM ex*.log WHERE (sc-status<400 or sc-status>=500) AND (TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%') GROUP BY cs-uri-stem, cs-method ORDER BY cs-uri-stem, cs-method" -rtp:-1
```

cs-uri-stem	cs-method	Total
/Default.asp	GET	9136
/Default.asp	HEAD	125
/asp/aspmail.asp	GET	3
/asp/aspmail.asp	POST	111

```

/awards/Default.asp          GET          269
/compare/Default.asp         GET          437
/compare/Default.asp         HEAD         3
/download/Default.asp        GET          5018
/download/Default.asp        HEAD         436
/download/default.asp        GET          727
/download/default.asp        HEAD         1
/orders/Default.asp          GET          1420
/orders/Default.asp          POST         3
...

```

You may also want to write a query that checks the HTTP referer header to make sure the traffic is coming from where you expect it to be coming from.

## Digging Deeper

At this point, you should begin to see patterns emerge. You should be able to narrow down the attack to specific dates and URL's. If you still have not found any apparent patterns, you may need to dig deeper. Sometimes an attack will involve sending a large amount of information back to the attacker. The following query will report some statistics for the number of bytes sent to the client

```

C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, Count(*) as Hits, AVG(sc-
bytes) AS Avg, Max(sc-bytes) AS Max, Min(sc-bytes) AS Min, Sum(sc-bytes) AS Total FROM ex*.log
WHERE TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%' GROUP
BY cs-uri-stem ORDER BY cs-uri-stem" -rtp:-1

```

cs-uri-stem	Hits	Avg	Max	Min	Total
/Default.asp	9261	18321	19920	145	16967359
/MSOffice/cltreq.asp	12	227	269	221	2724
/MailResult.asp	1	221	221	221	221
/asp/aspmail.asp	114	545	704	218	62232
/complete.asp	2	230	240	221	461
/orders/Default.asp	269	6998	7625	6692	1882463
...					

And this one will report on bytes sent from the client:

```

C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, Count(*) as Hits, AVG(cs-
bytes) AS Avg, Max(cs-bytes) AS Max, Min(cs-bytes) AS Min, Sum(cs-bytes) AS Total FROM ex*.log
WHERE TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%' GROUP
BY cs-uri-stem ORDER BY cs-uri-stem" -rtp:-1

```

cs-uri-stem	Hits	Avg	Max	Min	Total
/Default.asp	9261	435	1544	49	4037788
/MSOffice/cltreq.asp	12	369	482	276	4430
/MailResult.asp	1	313	313	313	313
/asp/aspmail.asp	114	1418	2383	153	161685
/complete.asp	2	172	191	154	345
/orders/Default.asp	269	441	1062	118	118766
...					

Another indicator may be how much time the server spent processing the request. It is not uncommon for exploits to take an unusually large amount of time or even timeout completely. The following query reports on time taken:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, Count(*) as Hits, AVG(time-taken) AS Avg, Max(time-taken) AS Max, Min(time-taken) AS Min, Sum(time-taken) AS Total FROM ex*.log WHERE TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%' GROUP BY cs-uri-stem ORDER BY cs-uri-stem" -rtp:-1
```

cs-uri-stem	Hits	Avg	Max	Min	Total
/Default.asp	9261	8	312	0	75228
/MSOffice/cltreq.asp	12	4	16	0	48
/MailResult.asp	1	0	0	0	0
/asp/aspmail.asp	114	699	31719	0	79765
/complete.asp	2	7	15	0	15
/orders/Default.asp	269	4	32	0	1206
...					

## User Logins

If your site is mostly unauthenticated anonymous access, then any user login may be suspicious. To see what users have authenticated to the site, try the following query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-username, Count(*) AS Hits from ex*.log WHERE cs-username IS NOT NULL GROUP BY cs-username ORDER BY Hits Desc" -rtp:-1
```

## User-Agents

Sometimes it is possible to identify an attack script by looking at the HTTP User-Agent header sent by the client. You can get a list of non-standard User-Agent strings with this query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT cs(User-Agent) FROM ex*.log WHERE TO_LOWERCASE(cs(User-Agent)) NOT LIKE '%mozilla%' AND TO_LOWERCASE(cs(User-Agent)) NOT LIKE '%opera%' ORDER BY cs(User-Agent)" -rtp:-1
```

## Closing In

Following these same patterns, you will eventually close in on the source of the intrusion or identify unknown intrusions. With each query, try to add more criteria and more detail to identify the specific log evidence to identify the attacker or type of attack. LogParser is a very powerful tool, but the real power comes when you learn how to use these and other queries to quickly bring information to your fingertips.

---

### About the Author

[Mark Burnett](#) is an independent security consultant and author who specializes in securing Windows-based servers. He is co-author of the best-selling book *Stealing the Network* (Syngress Publishing, ISBN: 1-931836-87-6). He has also co-authored or contributed to several other books, including *Special OPS: Host and Network Security for Microsoft, UNIX, and Oracle* (Syngress Publishing, ISBN: 1-931836-69-8); *Maximum Windows Security* (SAMS Publishing, ISBN: 0-672-31965-9); and *Dr. Tom Shinder's ISA Server and Beyond* (Syngress Publishing, ISBN: 1-931836-66-3). Mark is a regular contributor to many security-related magazines, newsletters, and web sites.

[more articles by Mark Burnett](#) on SecurityFocus

[Privacy Statement](#)

Copyright 2006, SecurityFocus